Online Passive-Aggressive Active Learning for Trapezoidal Data Streams

Yanfang Liu, Xiaocong Fan, *Senior Member, IEEE*, Wenbin Li, and Yang Gao, *Senior Member, IEEE*

***Abstract*— The idea of combining the active query strategy and the passive-aggressive (PA) update strategy in online learning can be credited to the PA active (PAA) algorithm, which has proven to be effective in learning linear classifiers from datasets with a fixed feature space. We propose a novel family of online active learning algorithms, named PAA learning for trapezoidal data streams ($\mathrm{PAA_{TS}}$) and multiclass $\mathrm{PAA_{TS}}$ ($\mathrm{MPAA_{TS}}$) (and their variants), for binary and multiclass online classification tasks on trapezoidal data streams where the feature space may expand over time. Under the context of an ever-changing feature space, we provide the theoretical analysis of the mistake bounds for both $\mathrm{PAA_{TS}}$ and $\mathrm{MPAA_{TS}}$. Our experiments on a wide variety of benchmark datasets have confirm that the combination of the instance-regulated active query strategy and the PA update strategy is much more effective in learning from trapezoidal data streams. We have also compared $\mathrm{PAA_{TS}}$ with online learning with streaming features ($\mathrm{OL_{SF}}$)—the state-of-the-art approach in learning linear classifiers from trapezoidal data streams. $\mathrm{PAA_{TS}}$ could achieve much better classification accuracy, especially for large-scale real-world data streams.**

***Index Terms*— Multiclass classification, online active learning, online learning, passive-aggressive (PA) learning, trapezoidal data streams.**

I. Introduction

ONLINE learning is an incremental learning approach where a predictive model is updated sequentially. It has been extensively studied [1]–[7], and applied when training data become available only gradually over time or it is computationally infeasible to train over the entire dataset.

Traditional online learning *always* assumes that the ground truth (e.g., the class labels in classification tasks) can be available to the learner at the end of each iteration. However, in many real-life applications, it would be too costly, if not possible, to manually label a large amount of training instances. In response to such a challenge, researchers have started to investigate online active learning [8]–[10], searching for effective query strategies that only need to reveal the labels of a small subset of instances.

On the other hand, most online learning methods, including online active learning algorithms in the literature, assume that the feature space they learn from remains constant. In particular, passive-aggressive active (PAA) [8] is an online active learning approach, which has proven to be effective in learning linear classifiers from datasets with a fixed feature space. Only a few recent studies have paid attention to learning from data streams with a dynamic feature space, such as trapezoidal data streams [11], [12] with an increasing feature space, evolving streams [13] with some features vanished and some other features being augmented, feature evolvable streams [14]–[18] with old features vanished and new features occurred, and capricious data streams [19], [20] with an arbitrarily varying feature space.

In this study, we focus on trapezoidal data streams where the feature space expands over time. Building upon the success of PAA [8] on data streams with a fixed feature space, we propose a novel family of online active learning algorithms, i.e., *PAA learning for trapezoidal data streams* ($\mathrm{PAA_{TS}}$) for online binary classification tasks, and multiclass $\mathrm{PAA_{TS}}$ ($\mathrm{MPAA_{TS}}$) for online multiclass classification tasks. The main contributions of this article are summarized as follows.

1) The PAA algorithm [8] introduces the idea of combining the active query strategy and the PA update strategy in online learning. We successfully extend this idea to handle both binary and multiclass classification tasks on trapezoidal data streams where the feature space may expand over time. Such an extension is nontrivial, because in the theoretical analysis of the mistake bounds of $\mathrm{PAA_{TS}}$, $\mathrm{MPAA_{TS}}$ and their variants, we have to carefully deal with the complexity due to the introduction of the ever-changing feature space.
2) The online learning with streaming features ($\mathrm{OL_{SF}}$) algorithm [11], [12] represents the state-of-the-art in learning linear classifiers from trapezoidal data streams. $\mathrm{OL_{SF}}$ uses the query-all strategy and the PA update strategy. We have experimentally shown that $\mathrm{PAA_{TS}}$ outperforms $\mathrm{OL_{SF}}$ consistently, confirming that the combination of the active query strategy and the PA update strategy is much more effective in learning from trapezoidal data streams.

Manuscript received November 22, 2021; revised April 15, 2022; accepted May 11, 2022. This work was supported in part by the Science and Technology Innovation 2030 New Generation Artificial Intelligence Major Project under Grant 2018AAA0100905, in part by the National Natural Science Foundation of China under Grant 62192783 and Grant 62106100, and in part by the Primary Research and Development Plan of Jiangsu Province under Grant BE2021028. *(Corresponding authors: Xiaocong Fan; Yang Gao.)*

Yanfang Liu is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China, and also with the College of Mathematics and Information Engineering, Longyan University, Longyan 364012, China (e-mail: liuyanfang003@163.com).

Xiaocong Fan is with the College of Science, Technology, Engineering and Mathematics, California State University, San Marcos, CA 92096 USA (e-mail: sfan@csusm.edu).

Wenbin Li and Yang Gao are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: liwenbin@nju.edu.cn; gaoy@nju.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3178880.

Digital Object Identifier 10.1109/TNNLS.2022.3178880

The remainder of this article is organized as follows. Section II introduces the related work in online learning for trapezoidal data streams and online active learning. In Section III, we present the $\mathrm{PAA_{TS}}$ algorithm and its variants for online binary classification tasks on trapezoidal data streams, and analyze the mistake bounds of the proposed algorithms. In Section IV, we extend the $\mathrm{PAA_{TS}}$ algorithms to a family of $\mathrm{MPAA_{TS}}$ algorithms for online multiclass classification tasks, as well as their mistake bounds. In Section V, we discuss experiment design and result analysis for both binary and multiclass online classification tasks. Section VI concludes this article.

II. Related Work

Our work is closely related to online learning for trapezoidal data streams and online active learning.

A. Online Learning for Trapezoidal Data Streams

The term “trapezoidal data stream” is first used in [12] to refer to doubly streaming data where both the data volume and feature dimension increase over time. In other words, the number of instances and the number of features in a trapezoidal data stream can grow simultaneously and steadily. For example, in text mining [21], [22], especially the infinite vocabulary topic model under online settings [23], both the number of documents and text vocabulary increase over time.

Integrating the online learning technique [24], [25] and streaming feature selection [26], [27], the $\mathrm{OL_{SF}}$ algorithm [11], [12] uses the PA principle [2] to learn from trapezoidal data streams. At each round, $\mathrm{OL_{SF}}$ receives an instance and makes a prediction by applying the classifier learned so far. After the prediction is made, the true label is disclosed and $\mathrm{OL_{SF}}$ computes an instantaneous loss which reflects the degree of its wrong prediction. The classifier is unchanged if the instantaneous loss is zero, otherwise $\mathrm{OL_{SF}}$ updates the weights of the learner by minimizing the loss and making them close to the previous classifier weights at the same time. In doing so, if the current instance carries new features, $\mathrm{OL_{SF}}$ also learns additional weights for the new features. $\mathrm{OL_{SF}}$ has two variants, $\mathrm{OL_{SF}}$-I and $\mathrm{OL_{SF}}$-II, which use the soft-margin technique where a slack variable is introduced into the optimization problem to reduce the overfitting issue in $\mathrm{OL_{SF}}$. $\mathrm{OL_{SF}}$ also uses a sparsity step to control the model’s complexity.

Since the $\mathrm{OL_{SF}}$ algorithms are limited to learn linear decision boundaries, adaptive single hidden layer feedforward neural network with shortcut connections (SLFN-S) [28] is proposed which provides growing and pruning capabilities to learn from trapezoidal data streams. Additionally, the $\mathrm{OL_{SF}}$ algorithms may suffer from poor convergence and be unable to rescale different features under streaming conditions. To tackle this issue, scale invariant learning with trapezoidal data streams (SILT) and its variant (SILT-I) [29] are proposed that can maintain feature scale-invariants even with an arbitrary scaling of features. To address the challenges that arise from trapezoidal data streams, the method of restructuring of Hoeffding trees is employed and the dynamic fast decision tree (DFDT) [30] is presented. The main drawback of the $\mathrm{OL_{SF}}$ algorithm (and its extensions) is that it needs to query the true label of every instance. And they only address online binary classification tasks.

B. Online Active Learning

Active learning is an iterative supervised learning approach where the learner may selectively learn from informative instances in situations where unlabeled data are abundant but manual labeling is expensive. A wide variety of query strategies have been examined to reduce the number of instances that is necessary to train a well-performed classification model.

Online active learning processes a stream of data in a sequential order. At each round, a learner is presented with a new instance, makes a prediction based on its current model, and then decides whether to query the true label. Once the label is revealed, the learner can use the feedback to update its prediction model for improved performance. Generally, online active learning algorithms fall under two categories [31], i.e., first-order algorithms and second-order algorithms.

The first-order online active learning algorithms exploit the first-order information for model update. The perceptron-based active (PEA) learning [9] and the PAA learning [8] are two well-known first-order online active learning methods. According to the principle of the Perceptron algorithm, the PEA learner does not update its model when it can correctly classify an instance. In so doing, the learner obviously will not benefit from the effort of label querying. The PAA learning approach uses the PA [2] update strategy, which allows the learner to fully exploit the potential of every queried instance for updating its classification model. It is also extended to handle online multiclass classification tasks. There are some other first-order online active learning algorithms, such as active online multitask relative similarity learning (MTRSL-Active) [32] for online multiclass classification tasks, double ramp loss active learning (DRAL), and double sigmoid loss active learning (DSAL) [10] for online binary classification tasks.

First-order online active learning algorithms tend to obtain suboptimal solutions [31]. *The second-order online learning* algorithms attempt to explore and exploit the underlying structural information [5], assuming that the weight vector (i.e., the linear classifier) follows a Gaussian distribution with a mean vector and a covariance matrix, both of which are updated at each round of online learning. One example is the selective sampling second-order perceptron algorithm (SEL-2nd) [33], [34]. It suffers from a serious limitation that the effort of querying the label of a correctly classified instance is wasted due to the use of the perceptron-based update strategy. Another approach is second-order online active learning (SOAL) [31], which is proposed to exploit both the first-order and second-order information.

All the existing online active learning methods assume a fixed feature space, hence they are not directly applicable to problems with a dynamically changing feature space. The objective of this study is to extend the PAA approach to advance the state-of-the-art in online learning from trapezoidal data streams.

III. Passive-Aggressive Active Learning for Trapezoidal Data Streams

In this section, we consider the problem setting of the online binary classification task on trapezoidal data streams.

A. Problem Setting and Background

We start with a typical online binary classification task on trapezoidal data streams where data dynamically change in both volume and feature dimension. Following $\mathrm{OL}_{\mathrm{SF}}$, the state-of-the-art online learning for trapezoidal data streams [11], we use $\{(\mathbf{x}_t, y_t)|t = 1, 2, \ldots, T\}$ to denote a sequence of input instances. Each instance $\mathbf{x}_t \in \mathbb{R}^{d_t}$ received at the tth round is a vector of d_t dimensions where $d_{t-1} \leq d_t$ and $y_t \in \{-1, +1\}$ are its true class label. The goal is to learn a linear classifier $\mathbf{w}_t \in \mathbb{R}^{d_{t-1}}$, which has the same dimension as the instance $\mathbf{x}_{t-1}$, and has either the same or smaller dimension as the current instance $\mathbf{x}_t$, for all $t = 2, 3, \ldots, T$. In the first round, $\mathbf{w}_1$ is initialized as a zero vector with the same dimension as $\mathbf{x}_1$.

Since the feature dimension may be increasing as the learning proceeds, we decompose $\mathbf{w}_{t+1} \in \mathbb{R}^{d_t}$ as $\mathbf{w}_{t+1} = [\mathbf{w}^e_{t+1}; \mathbf{w}^n_{t+1}]$, where the following holds.

1) $\mathbf{w}^e_{t+1} = \Pi_{\mathbf{w}_t}\mathbf{w}_{t+1} \in \mathbb{R}^{d_{t-1}}$ represents a projection of the feature space from dimension d_t to dimension d_{t-1}, and it is a vector consisting of elements of $\mathbf{w}_{t+1}$ which are in the same feature space of $\mathbf{w}_t$.
2) $\mathbf{w}^n_{t+1} = \Pi_{\mathbf{w}_{t+1}/\mathbf{w}_t}\mathbf{w}_{t+1} \in \mathbb{R}^{d_t - d_{t-1}}$ is a vector consisting of elements of $\mathbf{w}_{t+1}$ which are not in the feature space of $\mathbf{w}_t$.

By the same notations, we can decompose $\mathbf{x}_t \in \mathbb{R}^{d_t}$ as $\mathbf{x}_t = [\mathbf{x}^e_t; \mathbf{x}^n_t]$, where $\mathbf{x}^e_t = \Pi_{\mathbf{w}_t}\mathbf{x}_t$ and $\mathbf{x}^n_t = \Pi_{\mathbf{x}_t/\mathbf{w}_t}\mathbf{x}_t$.

At each round, the learner uses the current linear classifier to predict the label of the current instance by $\hat{y}_t = \mathrm{sign}(\mathbf{w}_t \cdot \mathbf{x}^e_t)$.

At the end of each round, a learner may employ a certain strategy to decide whether to reveal the true label of the current instance. Once the true label is revealed, the learner may suffer an instantaneous loss which reflects the degree of its wrong prediction, and the linear classifier can be improved for the upcoming round [2]. The loss function used in the PA algorithms [25] is called the hinge loss, i.e., $\ell(\mathbf{w}_t, (\mathbf{x}_t, y_t)) = max(0, 1 - y_t(\mathbf{w}_t \cdot \Pi_{\mathbf{w}_t}\mathbf{x}_t))$.

The $\mathrm{OL}_{\mathrm{SF}}$ algorithms [11], which extend the PA algorithms to learn from trapezoidal data streams, update the model $\mathbf{w}_{t+1}$ by solving three variants of the following optimization task:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \left[\mathbf{w}^e_{t+1}; \mathbf{w}^n_{t+1}\right] \\
&= \begin{cases}
\underset{\substack{\mathbf{w}=[\mathbf{w}^e;\mathbf{w}^n] \\ \ell_t = 0}}{\arg\min} \frac{1}{2}\|\mathbf{w}^e - \mathbf{w}_t\|^2 + \frac{1}{2}\|\mathbf{w}^n\|^2, \\
\quad (\mathrm{OL}_{\mathrm{SF}}) \\
\underset{\substack{\mathbf{w}=[\mathbf{w}^e;\mathbf{w}^n] \\ \ell_t \leq \xi; \xi \geq 0}}{\arg\min} \frac{1}{2}\|\mathbf{w}^e - \mathbf{w}_t\|^2 + \frac{1}{2}\|\mathbf{w}^n\|^2 + C\xi, \\
\quad (\mathrm{OL}_{\mathrm{SF}} - \mathrm{I}) \\
\underset{\substack{\mathbf{w}=[\mathbf{w}^e;\mathbf{w}^n] \\ \ell_t \leq \xi}}{\arg\min} \frac{1}{2}\|\mathbf{w}^e - \mathbf{w}_t\|^2 + \frac{1}{2}\|\mathbf{w}^n\|^2 + C\xi^2, \\
\quad (\mathrm{OL}_{\mathrm{SF}} - \mathrm{II})
\end{cases}
\end{aligned} \tag{1}$$

where $C > 0$ is a penalty parameter, and $\ell_t = \ell(\mathbf{w}, (\mathbf{x}_t, y_t)) = \max(0, 1 - y_t(\mathbf{w}^e \cdot \mathbf{x}^e_t) - y_t(\mathbf{w}^n \cdot \mathbf{x}^n_t))$ is the loss at round t.

The above optimization task has closed-form solutions, i.e., $\mathbf{w}_{t+1} = [\mathbf{w}_t + \tau_t y_t \mathbf{x}^e_t, \tau_t y_t \mathbf{x}^n_t]$, where τ_t is, respectively, computed according to the following equations:

$$\tau_t = \begin{cases}
\frac{\ell_t}{\|\mathbf{x}_t\|^2}, & (\mathrm{OL}_{\mathrm{SF}}) \\
\min\left(C, \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right), & (\mathrm{OL}_{\mathrm{SF}} - \mathrm{I}) \\
\frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}, & (\mathrm{OL}_{\mathrm{SF}} - \mathrm{II}).
\end{cases} \tag{2}$$

B. Passive-Aggressive Active Learning Algorithms for Trapezoidal Data Streams

Recent advance in online active learning centers around innovative strategies for when to query the true labels and how to update the model in the process. Three query strategies have been investigated in the literature: 1) the learner tries to query the true label of every instance; 2) the learner uses an *instance-irrelevant* random query strategy (i.e., all Bernoulli trials with the same success probability p) to decide whether to reveal the label of an incoming instance; and 3) the learner uses an *instance-regulated* random query strategy (i.e., a sequence of Bernoulli trials with instance-dependent success probabilities) to decide whether to reveal labels. On the other hand, there are two well-known update strategies: perceptron-based and PA-based.

In particular, the online PAA learning approach [8] uses an instance-regulated strategy to make query decisions and adopts the PA principle [2] to exploit every queried instance for updating the binary classification model. Inspired by this work, we here attempt to extend the PAA approach to improve the performance of learning from trapezoidal data streams. We use $\mathrm{PAA}_{\mathrm{TS}}$ below to distinguish it from the original PAA approach.

Algorithm 1 summarizes the details of our proposed $\mathrm{PAA}_{\mathrm{TS}}$ algorithm and its variants $\mathrm{PAA}_{\mathrm{TS}}$-I and $\mathrm{PAA}_{\mathrm{TS}}$-II.

In Lines (1)–(5), given an incoming instance $\mathbf{x}_t$ at the tth round, the prediction margin value $|f_t|$ is computed first, where $f_t = \mathbf{w}_t \cdot \Pi_{\mathbf{w}_t}\mathbf{x}_t$. The margin value can be interpreted as the degree of confidence in this prediction, which represents how far the current instance is away from the current classifier's hyperplane $\mathbf{w}_t$. Similar to the PAA approach [8], we employ the instance-regulated random query strategy to decide whether the label of an instance $\mathbf{x}_t$ should be queried or not

$$Pr(Z_t = 1) = \frac{\delta}{\delta + 1 + |f_t|} \tag{3}$$

where $Z_t \in \{0, 1\}$ is a Bernoulli random variable with respect to the instance $\mathbf{x}_t$, and $\delta > 0$ is a smoothing parameter. Note here that the Bernoulli probability varies from instance to instance as $|f_t|$ changes, and it is inversely proportional to the prediction confidence $|f_t|$. The smaller the value of $|f_t|$ is, the more uncertain the prediction of the classifier on the instance $\mathbf{x}_t$, so the instance should have a higher chance of being queried to obtain its true label.

In Lines (6)–(19), the learner updates the classifier conditionally. If the outcome $Z_t = 0$, the label of the instance $\mathbf{x}_t$ will

Algorithm 1 PAA$_{TS}$ Algorithm and Its Variants PAA$_{TS}$-I and PAA$_{TS}$-II

Input: penalty parameter $C > 0$ and smoothing parameter $\delta > 0$.
Initialize: $\mathbf{w}_1 = [0, \ldots, 0]^T \in \mathbb{R}^{d_1}$.

for $t = 1, 2, \cdots$ **do**
  receive instance: $\mathbf{x}_t \in \mathbb{R}^{d_t}$;
  compute: $f_t = \mathbf{w}_t^T \Pi_{\mathbf{w}_t} \mathbf{x}_t$;
  predict: $\hat{y}_t = sign(f_t)$;
  sample $Z_t \in \{0, 1\}$ with $Pr(Z_t = 1) = \frac{\delta}{\delta + 1 + |f_t|}$;
  if $Z_t = 1$ **then**
    query label: $y_t \in \{-1, +1\}$;
    suffer loss: $\ell_t = \max(0, 1 - y_t \mathbf{w}_t^T \Pi_{\mathbf{w}_t} \mathbf{x}_t)$;
    if $\ell_t > 0$ **then**
      set: $\tau_t = \begin{cases} \frac{\ell_t}{\|\mathbf{x}_t\|^2} & (\text{PAA}_{\text{TS}}) \\ \min(C, \frac{\ell_t}{\|\mathbf{x}_t\|^2}) & (\text{PAA}_{\text{TS}}\text{-I}) \\ \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} & (\text{PAA}_{\text{TS}}\text{-II}) \end{cases}$
      compute: $\mathbf{w}_{t+1}^e = \mathbf{w}_t + \tau_t y_t \Pi_{\mathbf{w}_t} \mathbf{x}_t$;
      compute: $\mathbf{w}_{t+1}^n = \tau_t y_t \Pi_{\mathbf{w}_{t+1}/\mathbf{w}_t} \mathbf{x}_t$;
      update: $\mathbf{w}_{t+1} = [\mathbf{w}_{t+1}^e; \mathbf{w}_{t+1}^n]$;
    else
      $\mathbf{w}_{t+1} = \mathbf{w}_t$;
    end if
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$;
  end if
end for

not be queried and the learner will not be updated; otherwise, the true label y_t of the instance $\mathbf{x}_t$ is queried and disclosed to the PAA$_{TS}$ learner. Then the PAA$_{TS}$ algorithm will use the true label y_t to compute the instantaneous loss, and update the linear classification model $\mathbf{w}_{t+1}$ according to (1).

It is worth noting that PAA$_{TS}$ differs from the OL$_{SF}$ algorithm in two aspects. First, OL$_{SF}$ queries the label of each incoming instance whereas PAA$_{TS}$ queries based on the outcome of a Bernoulli probability that is determined by the incoming instance. Second, the OL$_{SF}$ algorithm has one additional step to address "feature sparsity," where a projection and a truncation are introduced to prune redundant features. We purposely skip the sparsity step in PAA$_{TS}$ so that we can clearly see how the query ratio (the fraction of instances queried) may affect the learner's performance when all features are considered.

We next theoretically analyze the mistake bounds of the proposed PAA$_{TS}$ algorithms. It is worth noting that the theoretical analysis process of PAA$_{TS}$ is very similar to PAA [8], but this is nontrivial because as we extend PAA to handle trapezoidal data streams, we have to carefully consider the ever-changing feature space.

C. Analysis of Mistake Bounds for PAA$_{TS}$ Algorithms

We first introduce a lemma, which allows us to derive the mistake bounds for the three variants of PAA$_{TS}$ algorithm. For convenience, we introduce the following notation: $\mathcal{M} = \{t : t \in [T], \hat{y}_t \neq y_t\}$, and $\mathcal{L} = \{t : t \in [T], \hat{y}_t = y_t, \ell_t(\mathbf{w}_t) > 0\}$, where $[T]$ denotes $\{1, 2, \ldots, T\}$ and $\ell_t(\mathbf{w_t}) = \ell(\mathbf{w_t}; (\Pi_{\mathbf{w}_t} \mathbf{x}_t, y_t))$.

Lemma 1: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \le d_{t+1}$ and $y_t \in \{+1, -1\}$ for all $t \in [T]$. Let the learning rate $\tau_t \in \{(\ell_t/\|\mathbf{x}_t\|^2), \min(C, (\ell_t/\|\mathbf{x}_t\|^2)), (\ell_t/\|\mathbf{x}_t\|^2 + (1/2C))\}$ as given in Algorithm 1. Then, the following bound holds for any $\mathbf{u} \in \mathbb{R}^{d_T}$ and any $\alpha > 0$:

$$\sum_{t=1}^{T} 2 Z_t \tau_t [L_t(\alpha - |f_t|) + M_t(\alpha + |f_t|)] \le \alpha^2 \|\mathbf{u}\|^2 + \sum_{t=1}^{T} \tau_t^2 \|\mathbf{x}_t\|^2 + \sum_{t=1}^{T} 2\alpha \tau_t \ell_t(\mathbf{u})$$

where $\ell_t(\mathbf{u}) = \ell(\Pi_{\mathbf{x}_t} \mathbf{u}; (\mathbf{x}_t, y_t))$, $M_t = \mathbb{I}_{(t \in \mathcal{M})}$, $L_t = \mathbb{I}_{(t \in \mathcal{L})}$, $\mathbb{I}$ is an indicator function, that is,

$$M_t = \begin{cases} 1, & t \in \mathcal{M} \\ 0, & t \notin \mathcal{M} \end{cases} \qquad L_t = \begin{cases} 1, & t \in \mathcal{L} \\ 0, & t \notin \mathcal{L}. \end{cases}$$

The detailed proof of Lemma 1 can be found in "Appendix A" of the Supplemental Material. Based on Lemma 1, we first prove the expected mistake bound for the PAA$_{TS}$ algorithm in the linearly separable case. We assume that there exists a classifier $\mathbf{u} \in \mathbb{R}^{d_T}$ such that $y_t \Pi_{\mathbf{x}_t} \mathbf{u}^T \mathbf{x}_t \ge 1$ for all $t \in [T]$.

Theorem 1: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \le d_{t+1}$, $y_t \in \{+1, -1\}$, and $\|\mathbf{x}_t\| \le R$ for all t. Assume that there exists a classifier $\mathbf{u} \in \mathbb{R}^{d_T}$ such that $\ell_t(\mathbf{u}) = 0$ for all t. Assume a Bernoulli distribution $(\delta/\delta + 1 + |f_t|)$ is used for each query decision where $\delta > 0$, then the expected number of mistakes made by the PAA$_{TS}$ algorithm on this sequence is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \le \mathbb{E}\left[\sum_{t=1}^{T} M_t \ell_t(\mathbf{w}_t)\right] \le R^2 \left(\frac{\delta}{4} + \frac{1}{\delta} + 1\right) \|\mathbf{u}\|^2.$$

Proof: According to Lemma 1 and the fact $\ell_t(\mathbf{u}) = 0$ for all $t \in [T]$, we have

$$\sum_{t=1}^{T} 2 Z_t \tau_t [L_t(\alpha - |f_t|) + M_t(\alpha + |f_t|)] \le \alpha^2 \|\mathbf{u}\|^2 + \sum_{t=1}^{T} \tau_t^2 \|\mathbf{x}_t\|^2. \quad (4)$$

Note that $M_t = 1$ and $L_t = 1$ cannot hold at the same time, and we must have $\ell_t(\mathbf{w}_t) = 0$ (thus $\tau_t = 0$) when both $M_t = 0$ and $L_t = 0$ hold. Based on the learning rate $\tau_t = (\ell_t(\mathbf{w}_t)/\|\mathbf{x}_t\|^2)$ as given in the PAA$_{TS}$ algorithm, we can reformulate the inequality (4) and further simplify as follows:

$$\begin{aligned} \alpha^2 \|\mathbf{u}\|^2 &\ge \sum_{t=1}^{T} 2 Z_t \tau_t [L_t(\alpha - |f_t|) + M_t(\alpha + |f_t|)] - \sum_{t=1}^{T} \tau_t^2 \|\mathbf{x}_t\|^2 \\ &= \sum_{t=1}^{T} 2 Z_t \tau_t \Big[L_t \Big(\alpha - |f_t| - \frac{\tau_t}{2} \|\mathbf{x}_t\|^2\Big) + M_t \Big(\alpha + |f_t| - \frac{\tau_t}{2} \|\mathbf{x}_t\|^2\Big) \Big] \end{aligned}$$

$$
\begin{aligned}
&= \sum_{t=1}^{T} 2Z_t\tau_t\Bigg[L_t\left(\alpha - |f_t| - \frac{\ell_t(\mathbf{w}_t)}{2}\right)\\
&\qquad + M_t\left(\alpha + |f_t| - \frac{\ell_t(\mathbf{w}_t)}{2}\right)\Bigg]\\
&= \sum_{t=1}^{T} 2Z_t\tau_t\Bigg[L_t\left(\alpha - |f_t| - \frac{1 - y_t\mathbf{w}_t^T\Pi_{\mathbf{w}_t}\mathbf{x}_t}{2}\right)\\
&\qquad + M_t\left(\alpha + |f_t| - \frac{1 - y_t\mathbf{w}_t^T\Pi_{\mathbf{w}_t}\mathbf{x}_t}{2}\right)\Bigg]\\
&= \sum_{t=1}^{T} 2Z_t\tau_t\Bigg[L_t\left(\alpha - |f_t| - \frac{1-|f_t|}{2}\right)\\
&\qquad + M_t\left(\alpha + |f_t| - \frac{1+|f_t|}{2}\right)\Bigg]\\
&= \sum_{t=1}^{T} 2Z_t\tau_t\Bigg[L_t\left(\alpha - \frac{1+|f_t|}{2}\right) + M_t\left(\alpha - \frac{1-|f_t|}{2}\right)\Bigg].
\end{aligned} \tag{5}
$$

Suppose $\alpha = (\delta/2) + 1, \delta > 0$. The first item on the right-hand side of (5) is positive, because when $L_t = 1, |f_t| \in [0, 1)$, thus $\alpha - (1 + |f_t|/2) = (\delta + 1 - |f_t|/2) > 0$. Plugging $\alpha = (\delta/2) + 1$ into (5) results in

$$
\left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2 \geq \sum_{t=1}^{T} Z_t\tau_t M_t(\delta + 1 + |f_t|). \tag{6}
$$

Because $\tau_t = (\ell_t(\mathbf{w}_t)/\|\mathbf{x}_t\|^2) \geq (\ell_t(\mathbf{w}_t)/R^2)$, we can obtain

$$
\left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2 \geq \frac{1}{R^2}\sum_{t=1}^{T} Z_t M_t \ell_t(\mathbf{w}_t)(\delta + 1 + |f_t|). \tag{7}
$$

Given that the Bernoulli distribution has a probability of $(\delta/\delta + 1 + |f_t|)$, that is, $\mathbb{E}Z_t = (\delta/\delta + 1 + |f_t|)$, by taking expectation with the above inequality, we have

$$
\begin{aligned}
&\frac{1}{R^2}\mathbb{E}\left[\delta\sum_{t=1}^{T} M_t\ell_t(\mathbf{w}_t)\right]\\
&\quad = \frac{1}{R^2}\mathbb{E}\left[\sum_{t=1}^{T} M_t\ell_t(\mathbf{w}_t)(\delta + 1 + |f_t|)\mathbb{E}Z_t\right]\\
&\quad = \mathbb{E}\left[\frac{1}{R^2}\sum_{t=1}^{T} Z_t M_t\ell_t(\mathbf{w}_t)(\delta + 1 + |f_t|)\right]\\
&\quad \leq \left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2.
\end{aligned} \tag{8}
$$

After simplification, we can obtain

$$
\mathbb{E}\left[\sum_{t=1}^{T} M_t\ell_t(\mathbf{w}_t)\right] \leq R^2\left(\frac{\delta}{4} + \frac{1}{\delta} + 1\right)\|\mathbf{u}\|^2. \tag{9}
$$

■

Note that the above mistake bound indicates that the expected number of mistakes is proportional to the upper bound of the instance norm R and inversely proportional to the margin $(1/\|\mathbf{u}\|^2)$, which is consistent with the result for PAA [8] where the sequence of instances are in the same feature space.

We next present the expected mistake bounds for the PAA$_{TS}$-I and PAA$_{TS}$-II algorithms, which are more suitable for datasets that are not linearly separable.

Theorem 2: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \leq d_{t+1}$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Assume a Bernoulli distribution $(\delta/\delta + 1 + |f_t|)$ is used for each query decision where $\delta > 0$, then for any classifier $\mathbf{u} \in \mathbb{R}^{d_T}$, the expected number of mistakes made by the PAA$_{TS}$-I algorithm on this sequence is bounded by

$$
\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leq \beta\left\{\left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2 + (\delta + 2)C\sum_{t=1}^{T}\ell_t(\mathbf{u})\right\}
$$

where $\beta = (1/\delta)\max((1/C), R^2)$ and C is the penalty parameter for PAA$_{TS}$-I.

Proof: According to Lemma 1 and by following a similar derivation process as (5), we have:

$$
\begin{aligned}
&\alpha^2\|\mathbf{u}\|^2 + \sum_{t=1}^{T} 2\alpha\tau_t\ell_t(\mathbf{u})\\
&\quad \geq \sum_{t=1}^{T} 2Z_t\tau_t\left[L_t\left(\alpha - \frac{1+|f_t|}{2}\right) + M_t\left(\alpha - \frac{1-|f_t|}{2}\right)\right].
\end{aligned} \tag{10}
$$

Similarly, suppose $\alpha = (\delta/2) + 1, \delta > 0$, the first item on the right-hand side of (10) is positive, because when $L_t = 1, |f_t| \in [0, 1)$, thus $\alpha - (1 + |f_t|/2) > 0$. Then, (10) can be reformulated as

$$
\begin{aligned}
&\left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2 + \sum_{t=1}^{T}(\delta + 2)\tau_t\ell_t(\mathbf{u})\\
&\qquad\qquad \geq \sum_{t=1}^{T} Z_t\tau_t M_t(\delta + 1 + |f_t|).
\end{aligned} \tag{11}
$$

When $M_t = 1$, that is, $y_t\mathbf{w}_t^T\Pi_{\mathbf{w}_t}\mathbf{x}_t \leq 0$, we have $\ell_t(\mathbf{w}_t) \geq 1$. Using the assumption $\|\mathbf{x}_t\| \leq R$ and the learning rate $\tau_t = \min(C, (\ell_t(\mathbf{w}_t)/\|\mathbf{x}_t\|^2))$, we have $\tau_t \geq \min(C, (1/R^2))$. Thus, we can derive the following from (11):

$$
\begin{aligned}
&\left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2 + (\delta + 2)C\sum_{t=1}^{T}\ell_t(\mathbf{u})\\
&\quad \geq \left(\frac{\delta}{2} + 1\right)^2 \|\mathbf{u}\|^2 + \sum_{t=1}^{T}(\delta + 2)\tau_t\ell_t(\mathbf{u})\\
&\quad \geq \sum_{t=1}^{T} Z_t\tau_t M_t(\delta + 1 + |f_t|)\\
&\quad \geq \min\left(C, \frac{1}{R^2}\right)\sum_{t=1}^{T} Z_t M_t(\delta + 1 + |f_t|).
\end{aligned} \tag{12}
$$

Given that the Bernoulli distribution has a probability of $(\delta/\delta + 1 + |f_t|)$, we have

$$
\min\left(C, \frac{1}{R^2}\right)\mathbb{E}\left[\sum_{t=1}^{T} Z_t M_t(\delta + 1 + |f_t|)\right]
$$

$$= \min\left(C, \frac{1}{R^2}\right)\mathbb{E}\left[\sum_{t=1}^{T} M_t(\delta + 1 + |f_t|)\mathbb{E}Z_t\right]$$
$$= \delta \min\left(C, \frac{1}{R^2}\right)\mathbb{E}\left[\sum_{t=1}^{T} M_t\right]. \tag{13}$$

The theorem is proven by substituting (13) into (12)

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \le \beta\left\{\left(\frac{\delta}{2}+1\right)^2\|\mathbf{u}\|^2 + (\delta+2)C\sum_{t=1}^{T}\ell_t(\mathbf{u})\right\} \tag{14}$$

where $\beta = (1/\delta)\max((1/C), R^2)$. ■

Theorem 3: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \le d_{t+1}$, $y_t \in \{+1, -1\}$, and $\|\mathbf{x}_t\| \le R$ for all t. Assume a Bernoulli distribution $(\delta/\delta + 1 + |f_t|)$ is used for each query decision where $\delta > 0$, then for any classifier $\mathbf{u} \in \mathbb{R}^{d_T}$, the expected number of mistakes made by the PAA$_{\text{TS}}$-II algorithm on this sequence is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \le \beta\left\{\left(\frac{\delta}{2}+1\right)^2\|\mathbf{u}\|^2 + 2C\left(\frac{\delta}{2}+1\right)^2\sum_{t=1}^{T}\ell_t(\mathbf{u})^2\right\}$$

where $\beta = (1/\delta)(R^2+(1/2C))$ and C is the penalty parameter for PAA$_{\text{TS}}$-II.

Proof: Suppose

$$\mathcal{A} = \alpha^2\|\mathbf{u}\|^2 + \sum_{t=1}^{T}\tau_t^2\|\mathbf{x}_t\|^2 + \sum_{t=1}^{T}2\alpha\tau_t\ell_t(\mathbf{u})$$
$$\mathcal{B} = \sum_{t=1}^{T}\alpha\left\{\frac{\tau_t}{\sqrt{2C\alpha}} - \sqrt{2C\alpha}\ell_t(\mathbf{u})\right\}^2$$
$$\mathcal{C} = \alpha^2\|\mathbf{u}\|^2 + \sum_{t=1}^{T}\tau_t^2\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right) + \sum_{t=1}^{T}2C\alpha^2\ell_t(\mathbf{u})^2$$

then it is easy to prove that $\mathcal{A} \le \mathcal{A} + \mathcal{B} = \mathcal{C}$.

From Lemma 1, we have

$$\sum_{t=1}^{T}2Z_t\tau_t[L_t(\alpha - |f_t|) + M_t(\alpha + |f_t|)] \le \mathcal{A} \le \mathcal{C}. \tag{15}$$

Following a similar derivation process as (5), and given that the learning rate τ_t is set to $(\ell_t(\mathbf{w}_t)/\|\mathbf{x}_t\|^2 + (1/2C))$ in PAA$_{\text{TS}}$-II, (15) can be reformulated as follows:

$$\alpha^2\|\mathbf{u}\|^2 + \sum_{t=1}^{T}2C\alpha^2\ell_t(\mathbf{u})^2$$
$$\ge \sum_{t=1}^{T}\left\{2Z_t\tau_t[L_t(\alpha - |f_t|) + M_t(\alpha + |f_t|)] - \tau_t^2\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right)\right\}$$
$$= \sum_{t=1}^{T}\left\{2Z_t\tau_t\left[L_t\left(\alpha - |f_t| - \frac{\tau_t}{2}\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right)\right) + M_t\left(\alpha + |f_t| - \frac{\tau_t}{2}\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right)\right)\right]\right\}$$
$$= \sum_{t=1}^{T}2Z_t\tau_t\left[L_t\left(\alpha - |f_t| - \frac{\ell_t(\mathbf{w}_t)}{2}\right) + M_t\left(\alpha + |f_t| - \frac{\ell_t(\mathbf{w}_t)}{2}\right)\right]$$
$$= \sum_{t=1}^{T}2Z_t\tau_t\left[L_t\left(\alpha - \frac{1+|f_t|}{2}\right) + M_t\left(\alpha - \frac{1-|f_t|}{2}\right)\right]. \tag{16}$$

Similar to Theorems 1 and 2, suppose $\alpha = (\delta/2)+1, \delta > 0$, when $L_t = 1, |f_t| \in [0, 1)$, we have $(\alpha - (1 + |f_t|/2)) = (\delta + 1 - |f_t|/2) > 0$. Plugging $\alpha = (\delta/2) + 1$ into the above inequality results in

$$\left(\frac{\delta}{2}+1\right)^2\|\mathbf{u}\|^2 + 2C\left(\frac{\delta}{2}+1\right)^2\sum_{t=1}^{T}\ell_t(\mathbf{u})^2 \ge \sum_{t=1}^{T}Z_t\tau_t M_t(\delta + 1 + |f_t|). \tag{17}$$

The theorem is proven by applying the fact $\tau_t \ge (1/R^2 + (1/2C))$ and taking expectation on the above inequality. ■

IV. Extension to Multiclass Classification for Trapezoidal Data Streams

In this section, we extend the PAA$_{\text{TS}}$ algorithms to learn from streams of trapezoidal data with multiple class labels.

A. Problem Formulation

Let $\{(\mathbf{x}_t, y_t)|t = 1, 2, \ldots, T\}$ be a sequence of input instances. Each instance $\mathbf{x}_t \in \mathbb{R}^{d_t}$ received at the tth round is a vector of d_t dimensions where $d_t \ge d_{t-1}$, and it is associated with a unique class label $y_t \in Y = \{1, 2, \ldots, k\}$.

We adopt the multiprototype model in [2]. The classifier $\mathbf{W}$ consists of k weight vectors, where each weight vector $\mathbf{W}^r (r \in Y)$ corresponds to one class label. Since we are dealing with trapezoidal data streams, at the tth round (with $\mathbf{x}_t \in \mathbb{R}^{d_t}$ being the incoming instance), we denote the classifier as $\mathbf{W}_t \in \mathbb{R}^{d_{t-1}\times k}$, with $\mathbf{W}_t = [\tilde{\mathbf{W}}_t; \hat{\mathbf{W}}_t]$, where $\tilde{\mathbf{W}}_t = \Pi_{\mathbf{W}_{t-1}}\mathbf{W}_t \in \mathbb{R}^{d_{t-2}\times k}$ and $\hat{\mathbf{W}}_t = \Pi_{\mathbf{W}_t/\mathbf{W}_{t-1}}\mathbf{W}_t \in \mathbb{R}^{(d_{t-1}-d_{t-2})\times k}$, respectively, corresponding to the projection of $\mathbf{W}_t$ onto the feature space of $\mathbf{W}_{t-1}$ and onto the space with the newly introduced features by $\mathbf{x}_{t-1}$. Similarly, let $\mathbf{x}_t^e = \Pi_{\mathbf{W}_t}\mathbf{x}_t$ and $\mathbf{x}_t^n = \Pi_{\mathbf{x}_t/\mathbf{W}_t}\mathbf{x}_t$.

Then, a sequence of k prediction scores for all the class labels can be generated: $\mathbf{W}_t \cdot \mathbf{x}_t = [\mathbf{W}_t^1 \cdot \mathbf{x}_t^e, \ldots, \mathbf{W}_t^r \cdot \mathbf{x}_t^e, \ldots, \mathbf{W}_t^k \cdot \mathbf{x}_t^e]$. By comparing the above scores, the learner can choose the class label with the largest score as the prediction:

$$\hat{y}_t = \arg\max_{r\in Y}\mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t. \tag{18}$$

The margin is defined to be the gap between the prediction score of class y_t and the irrelevant class with the highest prediction score:

$$\gamma_t = \mathbf{W}_t^{y_t} \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t - \max_{r\neq y_t}\mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t. \tag{19}$$

Similar to the binary case, the hinge loss is computed by

$$\ell(\mathbf{W}_t, (\mathbf{x}_t, y_t)) = \max(0, 1 - \gamma_t). \quad (20)$$

B. Multiclass Passive-Aggressive Active Learning

In the multiclass setting, we need a different stochastic rule for deciding whether to query the label of a certain instance. The probability of querying a label should still be inversely proportional to the margin of the classifier on the current instance $\mathbf{x}_t$. However, the margin in (19) cannot be directly used because the true label y_t is not disclosed yet.

We adopt the same approach in [8] using a different confidence score, which is defined as the prediction score difference between the predicted label and the label with the second largest prediction score:

$$f_t = \mathbf{W}_t^{\hat{y}_t} \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t - \max_{r \neq \hat{y}_t} \mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t. \quad (21)$$

It is worth noting that $f_t \geq 0$ holds for all the instances. When a prediction is correct, i.e., $\hat{y}_t = y_t$, the confidence value f_t is equal to the margin γ_t; when a prediction is incorrect, i.e., $\hat{y}_t \neq y_t$, then it is easy to check $f_t \leq |\gamma_t|$. Based on this confidence score, the probability of querying a label in multiclass cases is set as $Pr(Z_t) = (\delta/\delta + 1 + f_t)$, where $\delta > 0$ is a smoothing parameter.

In case that the label of the current instance is revealed, we next need to decide how to update the classifier. At the tth round, given $\mathbf{W}_t \in \mathbb{R}^{d_{t-1} \times k}$ and $\mathbf{x}_t$, the new classifier $\mathbf{W}_{t+1} = [\tilde{\mathbf{W}}_{t+1}; \hat{\mathbf{W}}_{t+1}] \in \mathbb{R}^{d_t \times k}$ can be obtained by solving the following optimization problem:

$$\mathbf{W}_{t+1} = \mathop{\arg\min}_{\substack{\mathbf{W}=[\tilde{\mathbf{W}};\hat{\mathbf{W}}] \\ \ell_t(\mathbf{W})=0}} \frac{1}{2}\|\tilde{\mathbf{W}} - \mathbf{W}_t\|^2 + \frac{1}{2}\|\hat{\mathbf{W}}\|^2 \quad (22)$$

where $\ell_t(\mathbf{W}) = \ell(\mathbf{W}, (\mathbf{x}_t, y_t)) = \max(0, 1 - (\tilde{\mathbf{W}}^{y_t} \cdot \mathbf{x}_t^e + \hat{\mathbf{W}}^{y_t} \cdot \mathbf{x}_t^n - \max_{r \neq y_t}(\tilde{\mathbf{W}}^r \cdot \mathbf{x}_t^e + \hat{\mathbf{W}}^r \cdot \mathbf{x}_t^n))$ is the loss at round t.

Similar to the binary case, we can consider two variants of (22) for datasets that are not linearly separable:

$$\mathbf{W}_{t+1} = \mathop{\arg\min}_{\substack{\mathbf{W}=[\tilde{\mathbf{W}};\hat{\mathbf{W}}] \\ \ell_t(\mathbf{W})\leq\xi;\xi\geq 0}} \frac{1}{2}\|\tilde{\mathbf{W}} - \mathbf{W}_t\|^2 + \frac{1}{2}\|\hat{\mathbf{W}}\|^2 + C\xi \quad (23)$$

and

$$\mathbf{W}_{t+1} = \mathop{\arg\min}_{\substack{\mathbf{W}=[\tilde{\mathbf{W}};\hat{\mathbf{W}}] \\ \ell_t(\mathbf{W})\leq\xi}} \frac{1}{2}\|\tilde{\mathbf{W}} - \mathbf{W}_t\|^2 + \frac{1}{2}\|\hat{\mathbf{W}}\|^2 + C\xi^2. \quad (24)$$

The three optimization problems have closed-form solutions:

$$\begin{aligned}
\tilde{\mathbf{W}}_{t+1}^{y_t} &= \mathbf{W}_t^{y_t} + \tau_t \mathbf{x}_t^e \\
\tilde{\mathbf{W}}_{t+1}^{s_t} &= \mathbf{W}_t^{s_t} - \tau_t \mathbf{x}_t^e \\
\hat{\mathbf{W}}_{t+1}^{y_t} &= \tau_t \mathbf{x}_t^n \\
\hat{\mathbf{W}}_{t+1}^{s_t} &= -\tau_t \mathbf{x}_t^n
\end{aligned} \quad (25)$$

where $s_t = \arg\max_{r \neq y_t, r \in Y} \mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t$, and the stepsize τ_t is, respectively, computed as follows:

$$\tau_t = \begin{cases} \dfrac{\ell_t(\mathbf{W}_t)}{2\|\mathbf{x}_t\|^2}, & (\text{MPAA}_{\text{TS}}) \\ \min\left(C, \dfrac{\ell_t(\mathbf{W}_t)}{2\|\mathbf{x}_t\|^2}\right), & (\text{MPAA}_{\text{TS}} - \text{I}) \\ \dfrac{\ell_t(\mathbf{W}_t)}{2\|\mathbf{x}_t\|^2 + \frac{1}{2C}}, & (\text{MPAA}_{\text{TS}} - \text{II}). \end{cases} \quad (26)$$

The details of our proposed MPAA_{TS} algorithm (and its variants MPAA_{TS}-I and MPAA_{TS}-II) is given in Algorithm 2.

Algorithm 2 MPAA_{TS} algorithm and its variants MPAA_{TS}-I and MPAA_{TS}-II

Input: set of all labels $Y = \{1, 2, \ldots, k\}$, penalty parameter $C > 0$ and smoothing parameter $\delta > 0$.
Initialize: $\mathbf{W}_1 = zeros(d_1, k)$, where $\mathbf{x}_1 \in \mathbb{R}^{d_1}$.

for $t = 1, 2, \cdots$ **do**
  receive instance: $\mathbf{x}_t \in \mathbb{R}^{d_t}$;
  predict: $\hat{y}_t = \arg\max_{r \in Y} \mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t$;
  compute: $f_t = \mathbf{W}_t^{\hat{y}_t} \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t - \max_{r \neq \hat{y}_t} \mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t$;
  sample $Z_t \in \{0, 1\}$ with $Pr(Z_t = 1) = \frac{\delta}{\delta + 1 + f_t}$;
  if $Z_t = 1$ **then**
    query label: $y_t \in Y$;
    compute: $\gamma_t = \mathbf{W}_t^{y_t} \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t - \max_{r \neq y_t} \mathbf{W}_t^r \cdot \Pi_{\mathbf{W}_t}\mathbf{x}_t$;
    suffer loss: $\ell_t(\mathbf{W}_t) = \max(0, 1 - \gamma_t)$;
    if $\ell_t > 0$ **then**
      set: τ_t according to Eq. (26);
      compute: $\tilde{\mathbf{W}}_{t+1}$ and $\hat{\mathbf{W}}_{t+1}$ according to Eq. (25);
      update: $\mathbf{W}_{t+1} = [\tilde{\mathbf{W}}_{t+1}; \hat{\mathbf{W}}_{t+1}]$;
    end if
  else
    $\mathbf{W}_{t+1} = \mathbf{W}_t$;
  end if
end for

C. Analysis of Mistake Bounds for MPAA_{TS}

In this section, we aim to theoretically analyze the mistake bounds of the proposed MPAA_{TS} algorithms. To facilitate the understanding of Theorems 4 and 5, we first introduce Lemma 2.

Lemma 2: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \leq d_{t+1}$ and $y_t \in \{1, 2, \ldots, k\}$ for all $t \in [T]$. The confidence f_t and the learning rate τ_t are as given in (21) and (26). The following bound holds for any $\mathbf{U} = [\mathbf{U}^1, \mathbf{U}^2, \ldots, \mathbf{U}^k] \in \mathbb{R}^{d_T \times k}$:

$$\sum_{t=1}^{T} 2Z_t\tau_t[L_t(\alpha - f_t) + M_t(\alpha + f_t)] \leq \alpha^2 \sum_{r=1}^{k} \|\mathbf{U}^r\|^2 + \sum_{t=1}^{T} 2\tau_t^2\|\mathbf{x}_t\|^2 + \sum_{t=1}^{T} 2\alpha\tau_t\ell_t(\mathbf{U})$$

where $M_t = \mathbb{I}_{(t \in \mathcal{M})}, L_t = \mathbb{I}_{(t \in \mathcal{L})}$, $\mathbb{I}$ is an indicator function, $\alpha > 0$, and $\ell_t(\mathbf{U}) = \ell(\Pi_{\mathbf{x}_t}\mathbf{U}; (\mathbf{x}_t, y_t))$.

Theorem 4: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \leq d_{t+1}$, $y_t \in \{1, 2, \ldots, k\}$, and $\|\mathbf{x}_t\| \leq R$ for all t. Assume a Bernoulli distribution $(\delta/\delta + 1 + f_t)$ is used for each query decision where $\delta > 0$. Assume that there exists a classifier $\mathbf{U} \in \mathbb{R}^{d_T \times k}$ such that $\ell_t(\mathbf{U}) = 0$ for all t. The expected number of mistakes made by MPAA$_{TS}$ on this sequence is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leq \mathbb{E}\left[\sum_{t=1}^{T} M_t \ell_t(\mathbf{W}_t)\right] \leq 2R^2\left(\frac{\delta}{4} + \frac{1}{\delta} + 1\right)\sum_{r=1}^{k}\|\mathbf{U}^r\|^2.$$

The proof of Lemma 2 and Theorem 4 can be found in "Appendixes B and C" of the Supplemental Material. Similarly, we can prove Theorem 5 below. Because it is easy, we skip it for conciseness.

Theorem 5: Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of instances, where $\mathbf{x}_t \in \mathbb{R}^{d_t}, d_t \leq d_{t+1}$, $y_t \in \{1, 2, \ldots, k\}$, and $\|\mathbf{x}_t\| \leq R$ for all t. Assume a Bernoulli distribution $(\delta/\delta + 1 + f_t)$ is used for each query decision where $\delta > 0$. For any $\mathbf{W} = [\mathbf{W}^1, \mathbf{W}^2, \ldots, \mathbf{W}^k] \in \mathbb{R}^{d_T \times k}$, the expected number of mistakes made by the MPAA$_{TS}$-I is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leq \beta\left\{\left(\frac{\delta}{2}+1\right)^2\sum_{r=1}^{k}\|\mathbf{W}^r\|^2 + (\delta+2)C\sum_{t=1}^{T}\ell_t(\mathbf{W})\right\}$$

and the expected number of mistakes made by the MPAA$_{TS}$-II is bounded by

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leq \lambda\left\{\left(\frac{\delta}{2}+1\right)^2\sum_{r=1}^{k}\|\mathbf{W}^r\|^2 + 2C\left(\frac{\delta}{2}+1\right)^2\sum_{t=1}^{T}\ell_t(\mathbf{W})^2\right\}$$

where $\beta = (1/\delta)\max((1/C), 2R^2)$, $\lambda = (1/\delta)(2R^2 + (1/2C))$ and C is the penalty parameter.

V. Experimental Results

We evaluate our proposed algorithms, focusing on the PAA$_{TS}$ algorithms for binary classification tasks in Section V-A and the MPAA$_{TS}$ algorithms for multiclass classification tasks in Section V-B. All the algorithms are implemented in MATLAB R2019a, and all the experiments are conducted on a 64-bit PC with Intel Core i7-9700 CPU @3.00 GHz and 16-GB memory.

A. Evaluation of PAA$_{TS}$ Algorithms

To fully evaluate our proposed algorithms, we consider various combinations of query strategies and update strategies as shown in Fig. 1. All the online learning algorithms are implemented to handle trapezoidal data streams. In particular, the followings are given.

1) *RPE-TS:* It uses the perceptron-based update strategy $(\mathbf{w}_{t+1} = [\mathbf{w}_t + y_t\mathbf{x}_t^e, y_t\mathbf{x}_t^n]$ [11]) and the instance-irrelevant random query strategy [i.e., the same Bernoulli(p) for all instances].
2) *PEA-TS:* It uses the perceptron-based update strategy and the instance-regulated query strategy [see (3)].
3) *RPA$_{TS}$:* It uses the PA update strategy [see (1)] and the instance-irrelevant random query strategy. Because the update strategy can vary, RPA$_{TS}$ has two other variants RPA$_{TS}$-I and RPA$_{TS}$-II.
4) *PAA$_{TS}$:* It uses the PA update strategy and the instance-regulated query strategy. PAA$_{TS}$ also has two other variants PAA$_{TS}$-I and PAA$_{TS}$-II.
5) *OL$_{SF}$:* It is the state-of-the-art online learning approach that uses the PA update strategy and the query-all strategy. It also has two other variants OL$_{SF}$-I and OL$_{SF}$-II.

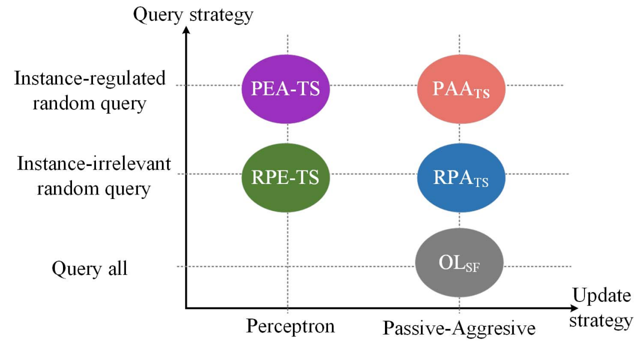

Fig. 1. Design rationale: query strategies and update strategies.

TABLE I
Datasets Description Used in the Experiments

DID	Dataset	# Instances	# Features
1	svmguide3	1243	21
2	HAPT	3266	561
3	gisette	7000	5000
4	mushrooms	8124	112
5	magic04	19020	10
6	a8a	32561	123
7	covtype	581012	54
8	HIGGS	11000000	28

We first compare PAA$_{TS}$ with RPE-TS, PEA-TS, and RPA$_{TS}$, and then in Section V-A6 compare PAA$_{TS}$ with OL$_{SF}$ using real-world data streams.

1) Experiment Settings: To examine the performance of the proposed algorithms, we conduct experiments on eight binary class datasets from machine learning repositories as listed in Table I. These datasets can be freely downloaded from UCI machine learning repository[1] and LIBSVM website.[2]

To simulate trapezoidal data streams, we follow the same method as used in [11] and [12], where each dataset is split into ten chunks, with each carrying only 10% of instances and a different number of features. More specifically, the first data chunk contains the first 10% of instances with the first 10%

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

of features; the second data chunk contains the second 10% of instances with an additional 10% of features (i.e., 20% of features), and so on. All the algorithms try to learn a linear classifier from the incoming trapezoidal data streams.

Performance is measured in terms of classification accuracy. All the experiments are repeated 20 times with random permutations on each dataset. If not noted otherwise, all the results reported here are averages over the 20 repeats. Also, the smoothing parameter δ is adjusted in the range $2^{[-10:10]}$ to set different query ratios for a learner.

2) Evaluation Under Fixed Query Ratios: We first compare our proposed $\mathrm{PAA_{TS}}$ algorithms and the other algorithms when they perform with the same query ratio. By adjusting the parameter δ (and p for the instance-irrelevant query strategy), we control the percentage of queried instances to be near 10% and 20%. For $\mathrm{PAA_{TS}}$-I, $\mathrm{PAA_{TS}}$-II, $\mathrm{RPA_{TS}}$-I, and $\mathrm{RPA_{TS}}$-II, because the value of the penalty parameter C under which one algorithm produces its best average performance can be different from another algorithm, C is searched in the range $10^{[-4:4]}$ through cross validation for all the datasets. Given in Table II are the best average performances for each method.

First, as shown in Table II, the algorithms employing the instance-regulated query strategy (PEA-TS and $\mathrm{PAA_{TS}}$) in most cases outperform their counterparts that employ the instance-irrelevant strategy (RPE-TS and $\mathrm{RPA_{TS}}$). Why is the instance-regulated query strategy more effective? Although all the algorithms query almost the same number of instances, the collection of instances to be queried is certainly different when a different query strategy is adopted. The key here is to query those instances that are the most significant in refining the model being constructed. The instance-regulated query strategy obviously helps in this regard. Indeed, when the degree of confidence f_t of an instance $\mathbf{x}_t$ is close to 0, it is more beneficial to reveal its true label because the existing model fails to classify $\mathbf{x}_t$ with certainty. According to (3), the probability to query an instance becomes higher when f_t moves closer to 0, so the learner has a bias toward querying uncertain instances. In contrast, the instance-irrelevant query strategy treats all instances equally.

Second, let's focus on PEA-TS and $\mathrm{PAA_{TS}}$, the algorithms employing the instance-regulated query strategy. The $\mathrm{PAA_{TS}}$ algorithms based on the PA update strategy have achieved significantly higher accuracy than the PEA-TS algorithm. This indicates that the PA update strategy is more effective than the perceptron-based strategy. Indeed, the perceptron-based update strategy never attempts to learn from instances that are correctly classified, whereas the PA strategy manages to fully exploit the potential of every queried instance for updating the classification model, including those that are correctly classified with low confidence. Because of this, analytically, the running time cost of the $\mathrm{PAA_{TS}}$ algorithms should be higher than PEA-TS. This can be confirmed by the running time results given in Table II.

Third, the two soft-margin algorithms, $\mathrm{PAA_{TS}}$-I and $\mathrm{PAA_{TS}}$-II, usually have similar accuracy performance, and they perform slightly better than $\mathrm{PAA_{TS}}$. This might be caused by over fitting on noisy training data, since the $\mathrm{PAA_{TS}}$ algorithm is more sensitive to noise.

3) Evaluation Under Varying Query Ratios: To understand further how the algorithms may be affected as the query ratio changes, we set the parameter C to 1, and vary the query ratio from 0.0 to 1.0 (by adjusting the parameter δ or p). The average classification accuracy and running time cost are plotted in Figs. 2 and 3, respectively.

As shown in Fig. 2, we observe that the accuracy usually increases with the increase of the query ratio in the beginning, and quickly reaches saturation after the query ratio exceeds a certain value. This is promising because it suggests that a well-performed linear classifier can be trained by revealing the labels of only a small fraction of the instances, regardless of the query strategy adopted. Most of the algorithms in comparison can reach their respective peak performance when the query ratio is close to 20%. It is interesting to note that on the covtype and HIGGS datasets, the accuracy decreases after reaching its peak performance, which is contrary to our thought that the more instances queried, the better the predictive performance. This might be caused by overfitting because these two datasets, as compared to others, have much more instances with a relatively smaller feature space.

Second, even when we are not comparing the peak performance as in Table II, the instance-regulated query strategy is still consistently more effective than the instance-irrelevant query strategy: we have $\mathrm{PAA_{TS}} > \mathrm{RPA_{TS}}$, $\mathrm{PAA_{TS}}$-I > $\mathrm{RPA_{TS}}$-I, $\mathrm{PAA_{TS}}$-II > $\mathrm{RPA_{TS}}$-II, and PEA-TS > $\mathrm{RPE_{TS}}$ hold for each dataset. This is more salient for bigger datasets like covtype and HIGGS. This again confirms that the instance-regulated strategy allows the learner to selectively query the most informative instances for model revision.

Third, the $\mathrm{PAA_{TS}}$ algorithms outperform PEA-TS and the $\mathrm{RPA_{TS}}$ algorithms outperform RPE-TS, which confirms that the PA update strategy is more effective than the perceptron-based strategy. This is consistent with the findings in [8], only that here the learners need to handle trapezoidal data streams.

As expected, Fig. 3 shows that the running time cost increases, almost linearly, as the query ratio increases. Also, the algorithms using the perceptron strategy has lower running time cost than those algorithms using the PA updating strategy.

4) Sensitivity to the Penalty Parameter: In this experiment, with the query ratio being set approximately to 10%, we evaluate the sensitivity of algorithms to the penalty parameter C, which is varied from 10^{-4} to 10^{4}.

Fig. 4 shows the performance of all the compared algorithms under different settings of C. First, we observe that the algorithms that use the soft-margin PA update rules can be greatly affected by the parameter C. The C setting that makes one algorithm achieve its peak performance can be different from that of another algorithm. The larger C is, the closer are the performance of the soft-margin algorithms $\mathrm{PAA_{TS}}$-I and $\mathrm{PAA_{TS}}$-II to $\mathrm{PAA_{TS}}$. This is because the step size τ_t in $\mathrm{PAA_{TS}}$-I and $\mathrm{PAA_{TS}}$-II becomes less affected by C as it increases. It is also worth noting that under the same setting of C, the algorithms using the instance-regulated query strategy always outperform those that use the instance-irrelevant strategy.

5) Comparison With State-of-the-Art OL_{SF}: In this section, we compare our proposed $\mathrm{PAA_{TS}}$, $\mathrm{PAA_{TS}}$-I, $\mathrm{PAA_{TS}}$-II algorithms with $\mathrm{OL_{SF}}$ and its two variants $\mathrm{OL_{SF}}$-I and $\mathrm{OL_{SF}}$-II [11],

TABLE II
EVALUATION OF THE PROPOSED PAA_{TS} ALGORITHMS AGAINST OTHER BASELINE ALGORITHMS

Dataset	Algorithm	Request 10% labels			Request 20% labels		
		Accuracy (%)	Query (%)	Times (s)	Accuracy (%)	Query (%)	Times (s)
svmguide3	RPE-TS	63.55 ± 2.99	9.57 ± 0.00	0.0109	63.86 ± 1.97	20.60 ± 0.00	0.0112
	RPA_{TS}	65.14 ± 3.37	9.57 ± 0.00	0.0111	64.75 ± 2.09	20.60 ± 0.00	0.0115
	RPA_{TS}-I	73.99 ± 2.70	9.57 ± 0.00	0.0112	75.32 ± 1.41	20.60 ± 0.00	0.0116
	RPA_{TS}-II	73.99 ± 2.70	9.57 ± 0.00	0.0111	75.31 ± 1.40	20.60 ± 0.00	0.0117
	PEA-TS	64.22 ± 2.79	9.76 ± 0.23	0.0111	64.25 ± 1.59	20.40 ± 0.46	0.0114
	PAA_{TS}(ours)	67.00 ± 2.61	9.32 ± 0.98	0.0112	66.77 ± 2.09	20.72 ± 1.67	0.0113
	PAA_{TS}-I(ours)	74.30 ± 2.01	9.68 ± 0.22	0.0110	$\mathbf{75.47 \pm 1.04}$	20.43 ± 0.15	0.0115
	PAA_{TS}-II(ours)	$\mathbf{74.46 \pm 2.15}$	9.95 ± 0.06	0.0111	75.46 ± 1.06	20.36 ± 0.16	0.0116
HAPT	RPE-TS	69.30 ± 5.55	9.83 ± 0.00	0.0353	77.28 ± 4.03	20.48 ± 0.00	0.0355
	RPA_{TS}	77.57 ± 1.32	9.83 ± 0.00	0.0362	83.20 ± 0.84	20.48 ± 0.00	0.0364
	RPA_{TS}-I	77.57 ± 1.32	9.83 ± 0.00	0.0356	83.20 ± 0.84	20.48 ± 0.00	0.0367
	RPA_{TS}-II	75.56 ± 1.32	9.83 ± 0.00	0.0365	83.49 ± 0.74	20.48 ± 0.00	0.0368
	PEA-TS	69.66 ± 5.42	9.63 ± 0.23	0.0358	77.18 ± 3.73	19.96 ± 0.36	0.0365
	PAA_{TS}(ours)	78.98 ± 1.30	10.08 ± 0.24	0.0364	84.44 ± 0.59	20.25 ± 0.29	0.0368
	PAA_{TS}-I(ours)	78.98 ± 1.30	10.08 ± 0.24	0.0364	84.46 ± 0.56	20.30 ± 0.25	0.0369
	PAA_{TS}-II(ours)	$\mathbf{79.03 \pm 1.25}$	10.10 ± 0.25	0.0363	$\mathbf{84.88 \pm 0.59}$	20.43 ± 0.23	0.0370
gisette	RPE-TS	79.30 ± 0.81	9.97 ± 0.00	0.3091	83.24 ± 0.00	20.00 ± 0.00	0.3104
	RPA_{TS}	85.56 ± 0.56	9.97 ± 0.00	0.2966	88.07 ± 0.30	20.00 ± 0.00	0.2919
	RPA_{TS}-I	85.59 ± 0.63	9.97 ± 0.00	0.2960	88.35 ± 0.36	20.00 ± 0.00	0.2919
	RPA_{TS}-II	85.94 ± 0.59	9.97 ± 0.00	0.2942	88.58 ± 0.30	20.00 ± 0.00	0.2919
	PEA-TS	79.89 ± 0.89	9.92 ± 0.10	0.2952	83.71 ± 0.67	20.29 ± 0.12	0.2977
	PAA_{TS}(ours)	86.82 ± 0.50	10.01 ± 0.17	0.2944	89.28 ± 0.34	20.22 ± 0.19	0.3012
	PAA_{TS}-I(ours)	87.04 ± 0.44	10.09 ± 0.16	0.2947	89.44 ± 0.38	20.41 ± 0.23	0.2988
	PAA_{TS}-II(ours)	$\mathbf{87.18 \pm 0.43}$	9.94 ± 0.16	0.2949	$\mathbf{89.71 \pm 0.24}$	20.46 ± 0.25	0.3000
mushrooms	RPE-TS	81.94 ± 0.98	10.02 ± 0.00	0.0724	83.48 ± 0.57	20.10 ± 0.00	0.0731
	RPA_{TS}	83.79 ± 0.79	10.02 ± 0.00	0.0736	85.11 ± 0.56	20.10 ± 0.00	0.0743
	RPA_{TS}-I	84.20 ± 0.90	10.02 ± 0.00	0.0740	85.68 ± 0.43	20.10 ± 0.00	0.0748
	RPA_{TS}-II	84.84 ± 0.74	10.02 ± 0.00	0.0735	86.38 ± 0.49	20.10 ± 0.00	0.0750
	PEA-TS	84.87 ± 0.68	9.97 ± 0.21	0.0730	83.88 ± 0.56	20.17 ± 0.15	0.0750
	PAA_{TS}(ours)	85.21 ± 0.65	9.93 ± 0.19	0.0734	85.80 ± 0.34	19.94 ± 0.58	0.0740
	PAA_{TS}-I(ours)	85.51 ± 0.55	10.12 ± 0.14	0.0740	86.47 ± 0.32	20.38 ± 0.35	0.0751
	PAA_{TS}-II(ours)	$\mathbf{85.74 \pm 0.59}$	10.13 ± 0.16	0.0739	$\mathbf{86.98 \pm 0.42}$	20.27 ± 0.29	0.0754
magic04	RPE-TS	54.35 ± 0.64	9.93 ± 0.00	0.1644	54.84 ± 0.57	19.93 ± 0.00	0.1687
	RPA_{TS}	55.61 ± 0.49	9.93 ± 0.00	0.1649	55.71 ± 0.57	19.93 ± 0.00	0.1691
	RPA_{TS}-I	64.58 ± 0.46	9.93 ± 0.00	0.1664	64.85 ± 0.11	19.93 ± 0.00	0.1727
	RPA_{TS}-II	64.66 ± 0.48	9.93 ± 0.00	0.1656	65.01 ± 0.15	19.93 ± 0.00	0.1719
	PEA-TS	54.49 ± 0.54	10.08 ± 0.04	0.1655	54.98 ± 0.47	20.25 ± 0.06	0.1704
	PAA_{TS}(ours)	56.41 ± 0.54	10.02 ± 0.14	0.1651	56.42 ± 0.54	20.29 ± 0.16	0.1677
	PAA_{TS}-I(ours)	$\mathbf{64.84 \pm 0.22}$	9.99 ± 0.18	0.1655	$\mathbf{65.07 \pm 0.13}$	20.34 ± 0.21	0.1714
	PAA_{TS}-II(ours)	64.70 ± 0.36	10.14 ± 0.14	0.1659	64.99 ± 0.13	20.06 ± 0.16	0.1730
a8a	RPE-TS	73.46 ± 0.47	10.03 ± 0.00	0.3029	73.90 ± 0.29	20.07 ± 0.00	0.3120
	RPA_{TS}	73.90 ± 0.45	10.03 ± 0.00	0.3057	74.17 ± 0.26	20.07 ± 0.00	0.3184
	RPA_{TS}-I	78.71 ± 0.25	10.03 ± 0.00	0.3088	79.12 ± 0.19	20.07 ± 0.00	0.3148
	RPA_{TS}-II	79.26 ± 0.25	10.03 ± 0.00	0.3141	79.77 ± 0.19	20.07 ± 0.00	0.3222
	PEA-TS	73.96 ± 0.54	10.04 ± 0.07	0.3091	74.32 ± 0.27	20.14 ± 0.14	0.3153
	PAA_{TS}(ours)	75.28 ± 0.40	9.97 ± 0.18	0.3113	75.29 ± 0.33	19.83 ± 0.25	0.3177
	PAA_{TS}-I(ours)	79.06 ± 0.23	10.17 ± 0.12	0.3108	79.45 ± 0.21	20.14 ± 0.18	0.3209
	PAA_{TS}-II(ours)	$\mathbf{79.46 \pm 0.15}$	10.06 ± 0.06	0.3166	$\mathbf{79.96 \pm 0.16}$	20.46 ± 0.16	0.3253
covtype	RPE-TS	51.90 ± 0.07	10.00 ± 0.00	5.3169	52.05 ± 0.06	19.98 ± 0.00	5.4153
	RPA_{TS}	51.22 ± 0.06	10.00 ± 0.00	5.3598	51.29 ± 0.06	19.98 ± 0.00	5.5308
	RPA_{TS}-I	54.83 ± 0.09	10.00 ± 0.00	5.4014	55.16 ± 0.05	19.98 ± 0.00	5.5965
	RPA_{TS}-II	55.00 ± 0.08	10.00 ± 0.00	5.4153	55.32 ± 0.06	19.98 ± 0.00	5.6230
	PEA-TS	52.05 ± 0.07	10.01 ± 0.02	5.3597	52.25 ± 0.08	20.20 ± 0.02	5.5078
	PAA_{TS}(ours)	51.84 ± 0.06	10.12 ± 0.02	5.4131	51.93 ± 0.05	20.41 ± 0.02	5.5759
	PAA_{TS}-I(ours)	54.92 ± 0.08	10.05 ± 0.04	5.4527	55.24 ± 0.05	20.28 ± 0.06	5.6349
	PAA_{TS}-II(ours)	$\mathbf{55.03 \pm 0.09}$	10.11 ± 0.02	5.4618	$\mathbf{55.33 \pm 0.06}$	20.27 ± 0.03	5.6746
HIGGS	RPE-TS	50.99 ± 0.01	10.00 ± 0.00	100.7369	50.99 ± 0.02	20.00 ± 0.00	102.5103
	RPA_{TS}	50.93 ± 0.01	10.00 ± 0.00	102.2356	50.93 ± 0.02	20.00 ± 0.00	105.0521
	RPA_{TS}-I	54.96 ± 0.04	10.00 ± 0.00	102.7267	55.10 ± 0.03	20.00 ± 0.00	106.9085
	RPA_{TS}-II	55.25 ± 0.03	10.00 ± 0.00	103.0964	55.30 ± 0.02	20.00 ± 0.00	107.1200
	PEA-TS	51.10 ± 0.02	10.08 ± 0.00	101.3929	51.08 ± 0.01	20.19 ± 0.00	104.2197
	PAA_{TS}(ours)	51.15 ± 0.02	10.12 ± 0.00	101.8341	51.12 ± 0.02	20.41 ± 0.01	105.3895
	PAA_{TS}-I(ours)	55.22 ± 0.06	10.01 ± 0.02	102.2176	55.31 ± 0.05	20.31 ± 0.04	106.1785
	PAA_{TS}-II(ours)	$\mathbf{55.29 \pm 0.02}$	10.11 ± 0.01	102.5441	$\mathbf{55.33 \pm 0.02}$	20.32 ± 0.01	106.5670

the state-of-the-art online learning approach for trapezoidal data streams.

OL_{SF} differs from PAA_{TS} in two aspects. First, OL_{SF} queries every instance whereas PAA_{TS} queries only the most informative instances using an instance-regulated query probability. Second, OL_{SF} has a sparsity step to control the proportion of features used, whereas PAA_{TS} takes all the features into consideration. The original codes of OL_{SF} (OL_{SF}-I and OL_{SF}-II) can be obtained at https://github.com/BlindReview/onlineLearning.

For OL_{SF} (and its variants, OL_{SF}-I and OL_{SF}-II), the algorithmic parameters are set to either their default values or the values that have produced the best performance as reported in [11]. In particular, we set $\lambda = 30$, and set the parameter

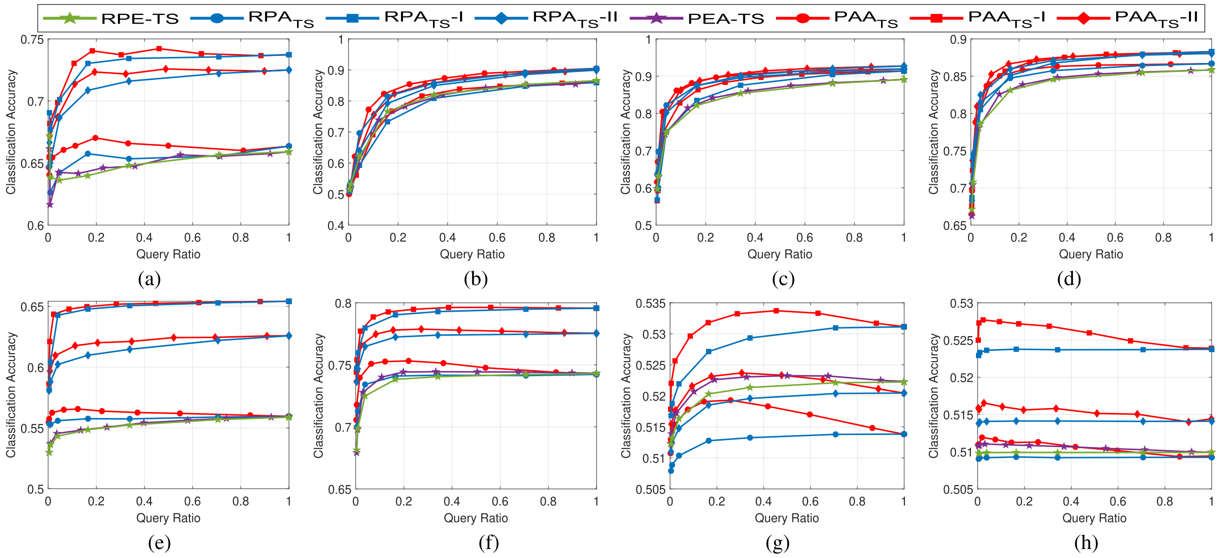

Fig. 2. Classification accuracy against the query ratio. The plotted curves are averaged over 20 random permutations when $C = 1$. (a) svmguide3. (b) HAPT. (c) gisette. (d) mushrooms. (e) magic04. (f) a8a. (g) covtype. (h) HIGGS.

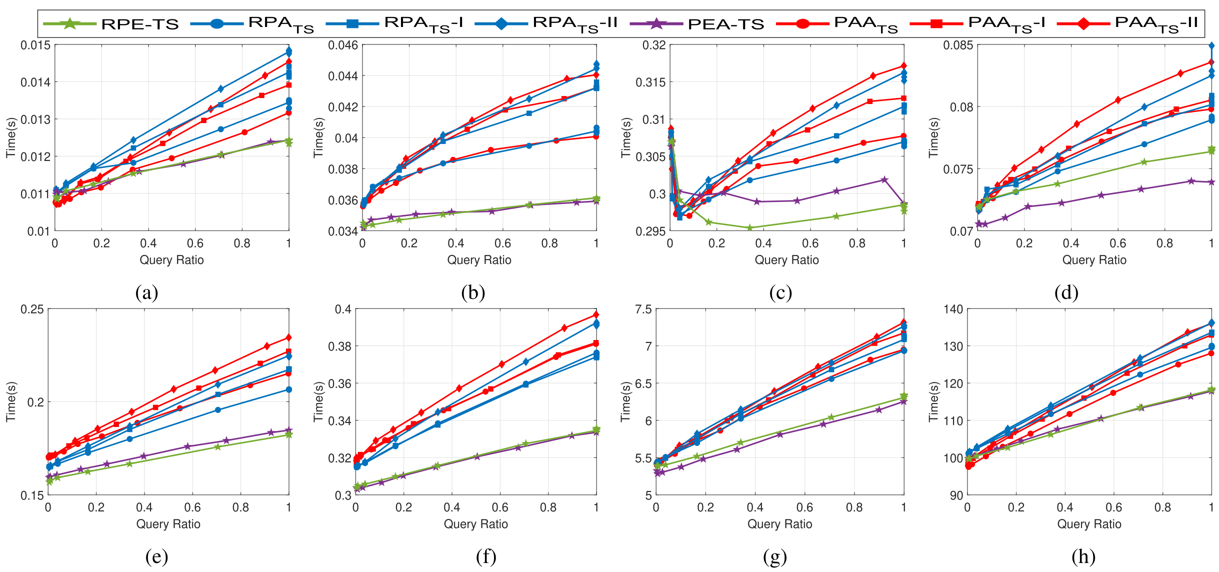

Fig. 3. Running time cost (seconds) with respect to the query ratio. The plotted curves are averaged over 20 random permutations when $C = 1$. (a) svmguide3. (b) HAPT. (c) gisette. (d) mushrooms. (e) magic04. (f) a8a. (g) covtype. (h) HIGGS.

B to vary in the range $[0.16, 0.32, 0.64, 1]$, i.e., we use 16%, 32%, 64%, and 100% of the features for learning the models, respectively. For our proposed algorithms, we set the query ratio to be near 20% by adjusting δ. For each of the algorithms, the penalty parameter C is searched in the range $10^{[-4:4]}$ to locate the value under which the best performance is produced. Table III lists the classification accuracy on different datasets.

From the result, we can observe that on all the datasets except for svmguide3 and magic04, our proposed algorithms PAA_{TS}, PAA_{TS}-I, and PAA_{TS}-II outperform the corresponding OL_{SF}, OL_{SF}-I, and OL_{SF}-II algorithms in different settings for B (feature sparsity). On the datasets svmguide3 and magic04, when OL_{SF}-I and OL_{SF}-II use all the features, their classification accuracy are only within 0.4% higher than our algorithms. It is worth reiterating that our algorithms only query 20% of instance labels, while the OL_{SF} algorithms use the label of every instance.

To get more insights, we plot the classification accuracy under different settings for C in Fig. 5. We can clearly see that except for a few C settings for the datasets svmguide3 and magic04, PAA_{TS} perform significantly better than OL_{SF}. This is even more true when it comes to real-world datasets as shown next.

6) Applications to Real-World Datasets: We compare PAA_{TS}-I with OL_{SF}-I and OL_{SF}I-all, for which we use the result reported in [11], where OL_{SF}-I uses only 0.1% of features (i.e., $B = 0.001$), and OL_{SF}I-all uses all the features on a dataset.

We use the same two binary classification datasets as used in [11]. Some characteristics of the datasets are given in

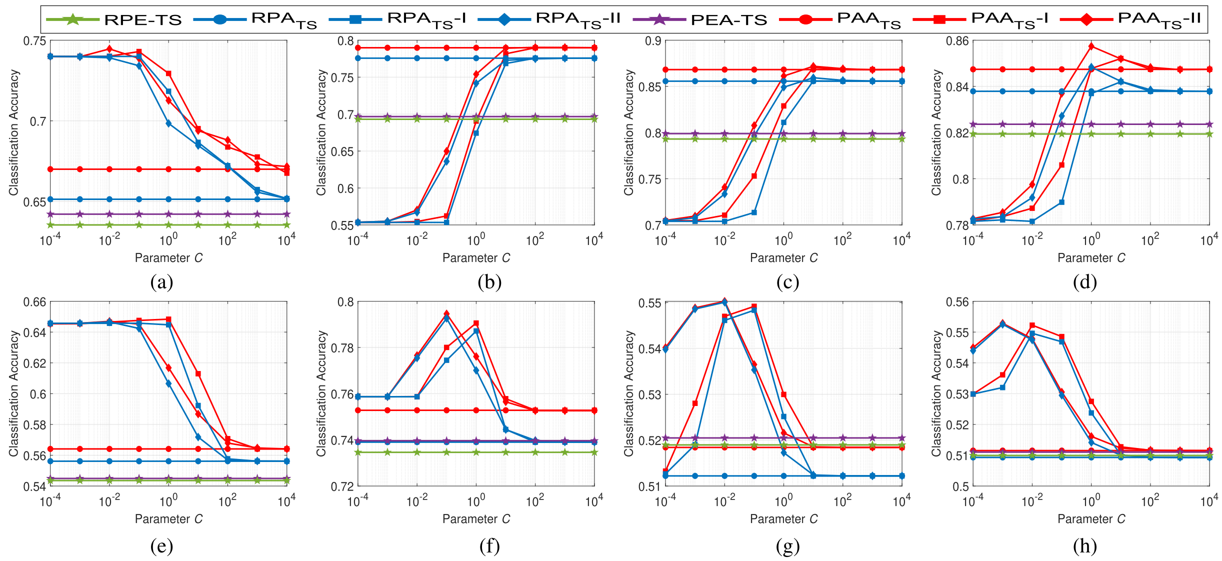

Fig. 4. Evaluation of classification accuracy against parameter C on all the datasets. The plotted curves are averaged over 20 random permutations. Query ratio is set to 10%. (a) svmguide3. (b) HAPT. (c) gisette. (d) mushrooms. (e) magic04. (f) a8a. (g) covtype. (h) HIGGS.

TABLE III

COMPARISON WITH RESPECT TO CLASSIFICATION ACCURACY. (QUERY RATIO IS SET TO 20% FOR PAA$_{TS}$)

Algorithm	svmguide3	HAPT	gisette	mushrooms	magic04	a8a	covtype	HIGGS
OL$_{SF}$ ($B = 0.16$)	50.94 ± 0.86	50.23 ± 0.61	49.92 ± 0.62	45.03 ± 0.46	32.43 ± 0.28	55.93 ± 0.28	50.03 ± 0.06	39.98 ± 0.01
OL$_{SF}$ ($B = 0.32$)	63.74 ± 0.85	50.23 ± 0.61	49.92 ± 0.62	49.35 ± 0.49	43.45 ± 0.38	61.60 ± 0.31	50.03 ± 0.06	50.13 ± 0.02
OL$_{SF}$ ($B = 0.64$)	63.75 ± 0.85	50.23 ± 0.61	49.92 ± 0.62	50.35 ± 0.52	54.42 ± 0.40	63.95 ± 0.31	50.03 ± 0.06	50.15 ± 0.02
OL$_{SF}$ ($B = 1$)	63.74 ± 0.87	50.23 ± 0.61	49.92 ± 0.62	50.35 ± 0.51	54.42 ± 0.40	64.07 ± 0.31	50.03 ± 0.06	50.15 ± 0.02
PAA$_{TS}$(ours)	$\mathbf{66.77 \pm 2.09}$	$\mathbf{84.44 \pm 0.59}$	$\mathbf{89.28 \pm 0.34}$	$\mathbf{85.80 \pm 0.34}$	$\mathbf{56.42 \pm 0.54}$	$\mathbf{75.29 \pm 0.33}$	$\mathbf{51.93 \pm 0.05}$	$\mathbf{51.12 \pm 0.02}$
OL$_{SF}$-I($B = 0.16$)	60.84 ± 0.64	52.81 ± 0.86	83.47 ± 1.92	65.49 ± 2.10	38.84 ± 0.19	68.43 ± 2.57	52.18 ± 0.05	42.36 ± 0.13
OL$_{SF}$-I($B = 0.32$)	75.31 ± 1.53	54.01 ± 0.64	83.46 ± 1.52	76.92 ± 2.48	51.81 ± 0.16	73.69 ± 0.33	52.22 ± 0.04	52.99 ± 0.02
OL$_{SF}$-I($B = 0.64$)	75.33 ± 1.54	56.18 ± 1.39	82.38 ± 1.48	80.93 ± 0.33	65.05 ± 0.08	75.95 ± 0.06	52.24 ± 0.04	52.99 ± 0.02
OL$_{SF}$-I($B = 1$)	$\mathbf{75.73 \pm 0.63}$	56.82 ± 1.42	81.97 ± 1.42	80.89 ± 0.32	$\mathbf{65.26 \pm 0.11}$	76.00 ± 0.06	52.24 ± 0.04	53.00 ± 0.00
PAA$_{TS}$-I(ours)	75.47 ± 1.04	$\mathbf{84.46 \pm 0.56}$	$\mathbf{89.44 \pm 0.38}$	$\mathbf{86.47 \pm 0.32}$	65.07 ± 0.13	$\mathbf{79.45 \pm 0.21}$	$\mathbf{55.24 \pm 0.05}$	$\mathbf{55.31 \pm 0.05}$
OL$_{SF}$-II($B = 0.16$)	60.84 ± 0.63	52.94 ± 0.43	82.77 ± 1.80	68.60 ± 3.01	38.84 ± 0.19	69.27 ± 2.24	54.00 ± 0.10	42.39 ± 0.01
OL$_{SF}$-II($B = 0.32$)	75.31 ± 1.53	55.07 ± 1.12	82.03 ± 1.46	78.90 ± 1.64	51.79 ± 0.13	73.74 ± 0.37	53.98 ± 0.11	53.03 ± 0.02
OL$_{SF}$-II($B = 0.64$)	75.33 ± 1.54	57.03 ± 1.03	80.17 ± 1.49	80.89 ± 0.31	65.10 ± 0.10	75.90 ± 0.05	54.08 ± 0.05	53.08 ± 0.01
OL$_{SF}$-II($B = 1$)	$\mathbf{75.74 \pm 0.62}$	57.22 ± 0.81	79.67 ± 1.46	80.86 ± 0.30	$\mathbf{65.33 \pm 0.11}$	75.92 ± 0.01	54.07 ± 0.05	53.05 ± 0.01
PAA$_{TS}$-II(ours)	75.46 ± 1.06	$\mathbf{84.88 \pm 0.59}$	$\mathbf{89.71 \pm 0.24}$	$\mathbf{86.98 \pm 0.42}$	64.99 ± 0.13	$\mathbf{79.96 \pm 0.16}$	$\mathbf{55.33 \pm 0.06}$	$\mathbf{55.33 \pm 0.02}$

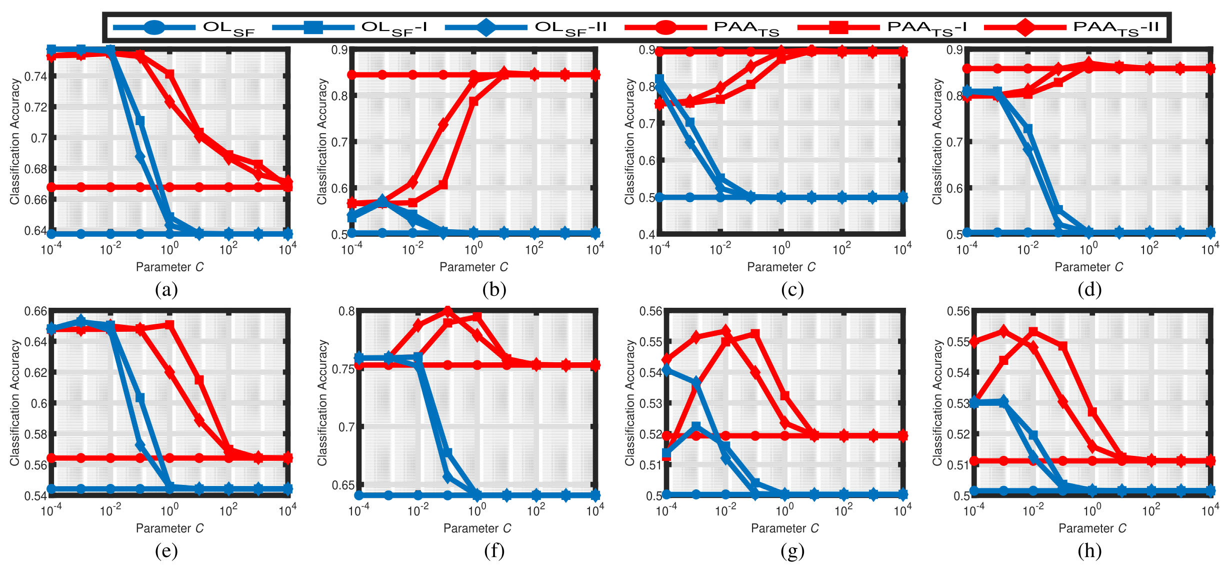

Fig. 5. Classification accuracy versus parameter C. The plotted curves are averaged over 20 random permutations. $B = 1$ for OL$_{SF}$. (a) svmguide3. (b) HAPT. (c) gisette. (d) mushrooms. (e) magic04. (f) a8a. (g) covtype. (h) HIGGS.

Table IV, where "data density" is the number of nonzero features versus the total number of features. The task of the rcv1 dataset is to classify JMLR articles into different groups, while the task of the URL dataset [35] is to use URL lexical

TABLE IV
DATASETS DESCRIPTION USED IN THE EXPERIMENTS

DID	Dataset	# Instances	# Features	Density
1	rcv1	697641	47236	0.15%
2	URL	2396130	3231961	0.04‰

TABLE V
COMPARISON OF CLASSIFICATION ACCURACY

Algorithms	rcv1	URL
OL_{SF}-I	65.66 ± 0.16	74.99 ± 0.37
OL_{SF}I-all	66.27 ± 0.21	74.67 ± 0.34
PAA_{TS}-I(ours)	$\mathbf{88.26 \pm 0.09}$	$\mathbf{96.46 \pm 0.01}$

TABLE VI
MULTICLASS CLASSIFICATION DATASETS DESCRIPTION USED IN THE EXPERIMENTS

DID	Dataset	# Instances	# Features	# Classes
1	dna	2000	180	3
2	satimage	4435	36	6
3	usps	7291	256	10
4	acoustic	78823	50	3
5	covtype	581012	54	7
6	poker	1000000	10	10

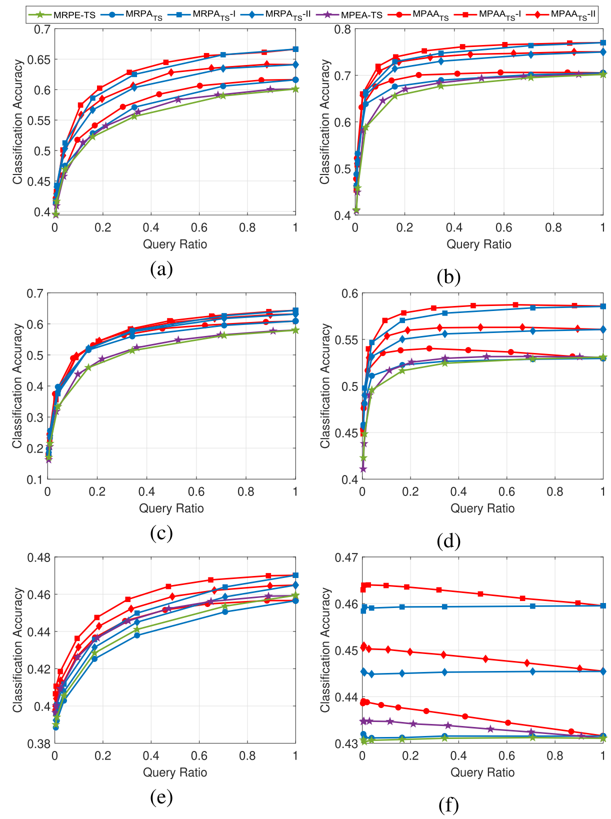

Fig. 6. Evaluation of classification accuracy versus query ratio. The plotted curves are averaged over 20 random permutations. $C = 1$. (a) dna. (b) satimage. (c) usps. (d) acoustic. (e) covtype. (f) poker.

and host-based features to detect malicious URLs from web pages.

For a fair comparison, in the $\mathrm{PAA}_{\mathrm{TS}}$-I algorithm we set $C = 0.1$, the same as used for $\mathrm{OL}_{\mathrm{SF}}$-I and $\mathrm{OL}_{\mathrm{SF}}$I-all. Due to the big volumes of the two datasets, for $\mathrm{PAA}_{\mathrm{TS}}$-I, we set the query ratio to be near 4% by adjusting δ. Note that the $\mathrm{OL}_{\mathrm{SF}}$-I and $\mathrm{OL}_{\mathrm{SF}}$I-all algorithms always query the label of every coming instance.

Table V shows the accuracy, where the results for $\mathrm{OL}_{\mathrm{SF}}$-I and $\mathrm{OL}_{\mathrm{SF}}$I-all are derived from Table VI in [11], the result of $\mathrm{PAA}_{\mathrm{TS}}$-I on the rcv1 dataset is the averages of 20 runs, and the result of $\mathrm{PAA}_{\mathrm{TS}}$-I on the URL dataset is the averages of five runs (due to days long running time).

Clearly, as far as the classification accuracy is concerned, our proposed algorithm $\mathrm{PAA}_{\mathrm{TS}}$-I significantly outperforms the $\mathrm{OL}_{\mathrm{SF}}$-I and $\mathrm{OL}_{\mathrm{SF}}$I-all algorithms, by over 20%. Obviously, the number of effective features is inherently small for real-world datasets like rcv1 and URL. Hence the strategy of querying only informative instances is a lot more beneficial than merely reducing the feature space.

B. Evaluation of the MPAA_{TS} Algorithms

We now empirically evaluate the performance of the proposed $\mathrm{MPAA}_{\mathrm{TS}}$ algorithm and its two variants $\mathrm{MPAA}_{\mathrm{TS}}$-I and $\mathrm{MPAA}_{\mathrm{TS}}$-II on online multiclass classification tasks.

Similar to the evaluation of $\mathrm{PAA}_{\mathrm{TS}}$, we compare $\mathrm{MPAA}_{\mathrm{TS}}$ (and its variants) with the same group of algorithms with appropriate adaptation to multiclass tasks as follows.

1) *MRPE-TS:* An extension of RPE-TS, the Multiclass Random PErceptron algorithm for Trapezoidal data Streams, with the perceptron updating strategy for learning, i.e., $\mathbf{W}_{t+1} = [\tilde{\mathbf{W}}_{t+1}, \hat{\mathbf{W}}_{t+1}]$, where $\tilde{\mathbf{W}}_{t+1}^{y_t} = \mathbf{W}_t^{y_t} + \mathbf{x}_t^e$, $\tilde{\mathbf{W}}_{t+1}^{s_t} = \mathbf{W}_t^{s_t} - \mathbf{x}_t^e$, $\hat{\mathbf{W}}_{t+1}^{y_t} = \mathbf{x}_t^n$, $\hat{\mathbf{W}}_{t+1}^{s_t} = -\mathbf{x}_t^n$.
2) *MPEA-TS:* An extension of PEA-TS, the multiclass PEA learning algorithm, with the instance-regulated query strategy.
3) *MRPA_{TS}:* The multiclass random PA learning algorithms for Trapezoidal data Streams use the instance-irrelevant query strategy, including $\mathrm{MRPA}_{\mathrm{TS}}$, $\mathrm{MRPA}_{\mathrm{TS}}$-I, and $\mathrm{MRPA}_{\mathrm{TS}}$-II, which are the extensions of the $\mathrm{RPA}_{\mathrm{TS}}$ algorithms.
4) *MPAA_{TS}:* Our proposed $\mathrm{MPAA}_{\mathrm{TS}}$ in Algorithm 2, including $\mathrm{MPAA}_{\mathrm{TS}}$, $\mathrm{MPAA}_{\mathrm{TS}}$-I, and $\mathrm{MPAA}_{\mathrm{TS}}$-II.

Table VI shows the characteristics of six multiclass datasets, which can be freely downloaded from the LIBSVM website. For the simulation of trapezoidal data streams and settings for other parameters, we follow those in the binary classification experiments in Section V-A1.

Fig. 6 summarizes the average performance of the eight algorithms when $C = 1$ as the query ratio varies. Fig. 7 shows the performance under different settings for C. We can observe similar phenomena as that in the binary classification setting. This demonstrates that our proposed algorithms are also effective in dealing with multiclass online active learning for trapezoidal data streams.

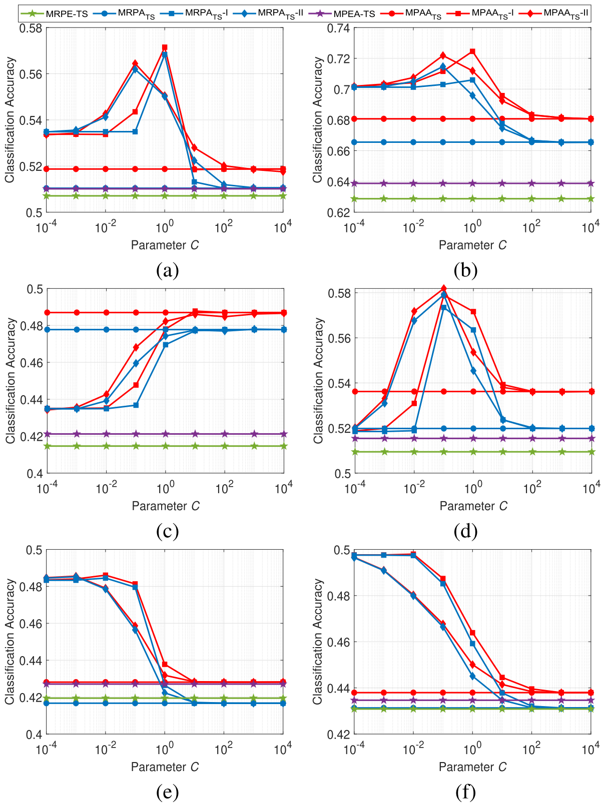

Fig. 7. Evaluation of classification accuracy versus parameter C. The plotted curves are averaged over 20 random permutations. Query ratio is set to 10%. (a) dna. (b) satimage. (c) usps. (d) acoustic. (e) covtype. (f) poker.

VI. Conclusion and Future Work

The idea of combining the active query strategy and the PA update strategy in online learning is originally proposed in the PAA approach [8], which has proven to be effective in learning linear classifiers from datasets with a fixed feature space. Building upon PAA, we propose a novel family of online active learning algorithms, named PAA_{TS} and MPAA_{TS} (and their variants), for binary and multiclass online classification tasks on trapezoidal data streams where the feature space may expand over time. Such an extension is nontrivial, because in the theoretical analysis of the mistake bounds of PAA_{TS} and MPAA_{TS}, we have to carefully deal with the complexity due to the introduction of the ever-changing feature space.

We have conducted experiments extensively to compare our proposed PAA_{TS} algorithms with other approaches that employ different combinations of query strategies and update strategies. The experiment results confirm that the combination of the instance-regulated active query strategy and the PA update strategy is much more effective in learning from trapezoidal data streams. We have also compared PAA_{TS} with OL_{SF}—the state-of-the-art approach in learning linear classifiers from trapezoidal data streams. PAA_{TS} could achieve much better classification accuracy, especially for large-scale real-world data streams.

In this article, PAA_{TS} algorithms are limited to learn linear decision boundaries which may perform poorly on nonlinear separable data. Thus, stacked bidirectional long short-term memory (LSTM) [36] can be employed to cope with nonlinear problems. Besides, trapezoidal data stream is only one type of data streams with a dynamic feature space. Some real-world problems may involve datasets where the instances have completely different sets of features, or the feature space may shrink over time. how to study SOAL techniques with an irregularly-changing feature space is also worth studying.

References

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–407, Sep. 1958.

[2] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.

[3] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 414–422.

[4] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Online learning with adaptive rebalancing in nonstationary environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4445–4459, Oct. 2021.

[5] S. C. Hoi, J. Wang, and P. Zhao, "LIBOL: A library for online learning algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 495–499, 2014.

[6] Y. Liu, Y. Yan, L. Chen, Y. Han, and Y. Yang, "Adaptive sparse confidence-weighted learning for online feature selection," in *Proc. 33th AAAI Conf. Artif. Intell.*, 2019, pp. 4408–4415.

[7] H. Li, W. Dong, and B.-G. Hu, "Incremental concept learning via online generative memory recall," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3206–3216, Jul. 2021.

[8] J. Lu, P. Zhao, and S. C. H. Hoi, "Online passive-aggressive active learning," *Mach. Learn.*, vol. 103, no. 2, pp. 141–183, May 2016.

[9] S. Dasgupta, A. T. Kalai, and A. Tauman, "Analysis of perceptron-based active learning," *J. Mach. Learn. Res.*, vol. 10, no. 2, pp. 281–299, 2009.

[10] K. Shah and N. Manwani, "Online active learning of reject option classifiers," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5652–5659.

[11] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Online learning from trapezoidal data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2709–2723, Oct. 2016.

[12] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Towards mining trapezoidal data streams," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 1111–1116.

[13] C. Hou and Z.-H. Zhou, "One-pass learning with incremental and decremental features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 11, pp. 2776–2792, Nov. 2018.

[14] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Learning with feature evolvable streams," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1417–1427.

[15] Z.-Y. Zhang, P. Zhao, Y. Jiang, and Z.-H. Zhou, "Learning with feature and distribution evolvable streams," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 11317–11327.

[16] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Learning with feature evolvable streams," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2602–2615, Jun. 2021.

[17] Z.-Y. Zhang, P. Zhao, Y. Jiang, and Z.-H. Zhou, "Storage fit learning with feature evolvable streams," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 7729–7736.

[18] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Prediction with unpredictable feature evolution," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 16, 2021, doi: 10.1109/TNNLS.2021.3071311.

[19] Y. He, B. Wu, D. Wu, E. Beyazit, S. Chen, and X. Wu, "Online learning from capricious data streams: A generative approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2491–2497.

[20] E. Beyazit, J. Alagurajah, and X. Wu, "Online learning from data streams with varying feature spaces," in *Proc. 33th AAAI Conf. Artif. Intell.*, 2019, pp. 3232–3239.

[21] A. Onan, "Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering," *IEEE Access*, vol. 7, pp. 145614–145633, 2019.

[22] A. Onan, "Sentiment analysis on massive open online course evaluations: A text mining and deep learning approach," *Comput. Appl. Eng. Educ.*, vol. 29, no. 3, pp. 572–589, May 2021.

[23] K. Zhai and J. Boyd-Graber, "Online latent Dirichlet allocation with infinite vocabulary," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 561–569.

[24] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 905–912.

[25] S. Shalev-Shwartz, K. Crammer, O. Dekel, and Y. Singer, "Online passive-aggressive algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 1229–1236.

[26] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1178–1192, May 2013.

[27] X. Wu, K. Yu, H. Wang, and W. Ding, "Online streaming feature selection," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 1159–1166.

[28] E. Beyazit, M. Hosseini, A. Maida, and X. Wu, "Learning simplified decision boundaries from trapezoidal data streams," in *Proc. 27th Int. Conf. Artif. Neural Netw.*, 2018, pp. 508–517.

[29] J. Alagurajah, X. Yuan, and X. Wu, "Scale invariant learning from trapezoidal data streams," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 505–508.

[30] C. Schreckenberger, T. Glockner, H. Stuckenschmidt, and C. Bartelt, "Restructuring of Hoeffding trees for trapezoidal data streams," in *Proc. Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2020, pp. 416–423.

[31] S. Hao, J. Lu, P. Zhao, C. Zhang, S. C. H. Hoi, and C. Miao, "Second-order online active learning and its applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1338–1351, Jul. 2018.

[32] S. Hao, P. Zhao, Y. Liu, S. C. H. Hoi, and C. Miao, "Online multitask relative similarity learning," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1823–1829.

[33] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "A second-order perceptron algorithm," *SIAM J. Comput.*, vol. 34, no. 3, pp. 640–668, 2005.

[34] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Worst-case analysis of selective sampling for linear classification," *J. Mach. Learn. Res.*, vol. 7, pp. 1205–1230, Jul. 2006.

[35] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 681–688.

[36] A. Onan and M. A. Toçoğlu, "A term weighted neural language model and stacked bidirectional LSTM based framework for sarcasm identification," *IEEE Access*, vol. 9, pp. 7701–7722, 2021.

Yanfang Liu received the M.S. degree from Minnan Normal University, Zhangzhou, China, in 2014. She is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science and Technology, Nanjing University, Nanjing, China.

She has been a Lecturer with the College of Mathematics and Information Engineering, Longyan University, Longyan, China, Since 2014. Her main research interests include online learning and machine learning.

Xiaocong Fan (Senior Member, IEEE) received the Ph.D. degree from the Department of Software Engineering, Nanjing University, Nanjing, China, in 1999.

He is currently a Professor with the College of Science, Technology, Engineering and Mathematics, California State University, San Marcos, CA, USA. His main research interests include multiagent systems and machine learning.

Wenbin Li received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2019.

He is currently an Assistant Researcher with the Department of Computer Science and Technology, Nanjing University. His research interests include machine learning and computer vision, particularly in metric learning, few-shot learning, and their applications to face recognition and image classification.

Yang Gao (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2000.

He is currently a Professor and the Deputy Director of the Department of Computer Science and Technology, Nanjing University, where he is also directing the Reasoning and Learning Research Group. He has authored or coauthored more than 100 papers in top-tier conferences and journals. His current research interests include artificial intelligence, machine learning, multiagent systems, and intelligent system.

Dr. Gao is also the program chair and area chair for many international conferences.