IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 3, 2020, accepted November 9, 2020, date of publication November 26, 2020, date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040816

Budgeted Passive-Aggressive Learning for Online Multiclass Classification

CHUNG-HAO WU[1], HENRY HORNG-SHING LU[2], AND HSUEH-MING HANG[1], (Fellow, IEEE)
[1]Institute of Electronics, National Chiao Tung University, Hsinchu 30010, Taiwan
[2]Institute of Statistics, National Chiao Tung University, Hsinchu 30010, Taiwan

Corresponding author: Henry Horng-Shing Lu (hslu@stat.nctu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, R.O.C., under Grant no. MOST 107-2118-M-009-006-MY3, in part by the Higher Education Sprout Project of the National Chiao Tung University and Ministry of Education, Taiwan, R.O.C., and in part by the Ministry of Education Yushan Scholar Program, Taiwan, R.O.C.

ABSTRACT Online multiclass classification is a specific problem of online learning that performs a sequence of multiclass classification tasks given the knowledge of previous tasks. The goal is to make correct predictions for this sequence. It is generally considered a more complicated problem than its binary counterpart, online binary classification. A popular algorithm, called the passive-aggressive algorithm, was primarily proposed for binary problems and later extended as the multiclass passive-aggressive (MPA) algorithm for multiclass problems. The nature of MPA allows itself to implement the kernel trick, which enables us to make better predictions with a kernel-based model. However, this approach suffers from the curse of kernelization that causes unbounded growth of the model in memory usage and runtime. To solve the growth problem, we first introduce a resource perspective that gives an alternative and equivalent interpretation of the kernel-based MPA algorithm. Based on the resource perspective, we propose the budgeted MPA (BMPA) algorithm, which approximates the kernel-based MPA algorithm. BMPA limits the maximum number of available resources by removal and fully exploits them through a constrained optimization. We study three removal strategies and give a relative mistake bound that provides a unified analysis. Simulation experiments on various datasets are conducted to demonstrate that BMPA is effective and competitive with state-of-the-art budgeted online algorithms.

INDEX TERMS Budget, budgeted algorithm, online learning, online multiclass classification, relative mistake bound, reproducing kernel Hilbert space, resource.

I. INTRODUCTION

Online learning aims to solve a sequence of prediction tasks given knowledge of correct targets of previous tasks [1]–[3]. On a task round, a prediction is made to the received instance, and then an update of the prediction model based on the correct target received later is performed to improve prediction for future tasks. The goal of online learning is to make accurate predictions for the sequence; as long as the prediction model can be adapted to the sequence, it does not matter if the model will converge or not. Because of its adaptive nature, online learning is suited for many practical applications that receive streaming data, such as real-time malicious URL detection [4] and ad click-through rate prediction [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

Numerous online algorithms have been proposed for the online setting [6]–[11]. Most of the methods are focused on the design of the update rule with a linear model. However, this simple linearity may limit the prediction performance. For a complex problem, a linear model may require the company with a superior feature extraction to achieve a good prediction performance.

Fortunately, the prevalence of support vector machines inspires the application of kernels to online learning [12]. Since the nature of many online algorithms allows themselves to be kernelized easily, e.g., the Perceptron algorithm [13] and the online gradient descent (OGD) algorithm [14], a kernel-based model, which is a nonlinear model composed of kernel functions, can usually be used to achieve a better prediction performance. This kernel-based approach is known as the kernel trick, which replaces all the inner products in an algorithm by kernel functions. While the kernel trick is simple yet effective, a kernelized online algorithm needs to store support vectors and associated combination weights together to represent a kernel-based model. It turns out that the kernelized online algorithm suffers from the curse of kernelization that results in unbounded growth in memory usage and runtime for a task round as more and more tasks are done [15]. It may make a kernelized online algorithm broken-down on some resource-insufficient occasions, e.g., making predictions on smartphones with limited computational power.

Many researchers have tried to address this issue in binary cases that deal with a sequence of binary classification tasks. Most existing works are focused on controlling the growth of a kernel-based model by restricting the number of support vectors [13], [14], [16]–[20]. Some works transform a kernel-based model into a linear model with kernel-induced feature approximation and thus avoid the growth [21], [22]. The same issue exists in multiclass cases that face a sequence of multiclass classification tasks. Although a multiclass problem is generally more complicated than a binary problem, several research works have attempted to cure the curse by controlling the growth of a model [13], [23], or using kernel-induced feature approximation [21], [22].

Among state-of-the-art online methods, a family of margin-based online algorithms called passive-aggressive (PA) algorithms has drawn lots of attention in recent years [10]; the popularity is likely because it can be used to solve many kinds of problems, such as classification, regression, and structured prediction, and the formulation for the update of a prediction model is simple and neat. The PA algorithms have facilitated fruitful applications [4], [24]–[27] and have inspired many subsequent online algorithms [28]–[32]. A PA algorithm can be kernelized to make more accurate predictions with a nonlinear model, but still suffers from the curse of kernelization making it difficult to work in applications provided with limited computational power. However, there exist only a few research works focused on overcoming the curse for PA algorithms applied to binary classification [20], [33]. It is worth to study how to overcome this issue for PA algorithms in various types of problems such as multiclass classification and structured prediction; consequently, kernel-based models can be applied safely in more and more practical applications. Moreover, for PA-based online algorithms, the study may shed some light on how to overcome their curse.

In this paper, we attempt to break the curse of kernelization for the multiclass classification version of the PA algorithm, which is referred to as the multiclass PA (MPA) algorithm in the rest of this paper. There are two main challenges of breaking the curse for the MPA algorithm. First, a kernel-based prediction model for m-class classification ($m > 2$) consists of m kernel-based hypotheses instead of only one hypothesis for binary classification; thus, we should somehow simultaneously limit the growth of all hypotheses to control the growth of the model. Second, since controlling the growth of a model will result in some sacrifice in the prediction performance, it is necessary to diminish the information loss in the updated model to maintain the performance.

To tackle these challenges, we propose a new budgeted method called the budgeted multiclass passive-aggressive (BMPA) algorithm to control and update all the hypotheses of a model at the same time. Concretely, we make the following contributions in this paper.

1) To provide a solid explanation of the proposed BMPA algorithm, we introduce the resource perspective that treats every encountered instance as a potential resource and the kernel-based MPA algorithm as a manager exploiting available resources for simultaneously constructing all hypotheses of a prediction model. It gives an alternative and equivalent interpretation of the kernel-based MPA algorithm.
2) Through the resource perspective, we propose the BMPA algorithm that exploits only a finite number of available resources to approximate the kernel-based MPA algorithm. Specifically, the BMPA algorithm employs a projection approach to diminish the information loss in the updated model when there exists any unaffordable resource.
3) We study three kinds of budget maintenance strategies about how to select the unaffordable resource to remove and suggest to use the smallest removal strategy that removes the resource with the smallest magnitude of the weights. The smallest removal strategy achieves a good tradeoff between prediction performance and runtime.
4) We justify the proposed BMPA algorithm and budget maintenance strategies by providing a unified relative mistake bound and conducting comprehensive empirical experiments on eight open datasets.

Deep learning (DL)-based classification methods, e.g., AlexNet [34], GoogLeNet [35], and ResNet [36], are typically trained by backpropagation in a batch learning setting, which requires the entire training data to be collected before the learning task. Therefore, batch learning is also called offline learning. Moreover, machine learning (ML)-based classification methods, e.g., multiclass logistic regression [37], multiclass Gaussian process classification [38], and multiclass SVMs [39], are also typically trained in a batch learning setting. These types of DL-based or ML-based methods aim to learn a fixed classification model that achieves good generalization for new testing data. On the other hand, the kernel-based MPA algorithm investigated in this study is used to deal with a sequence of online and real-time prediction tasks. Its goal is to make accurate predictions for the sequence of input data and the real-time prediction model can change dynamically. As long as the prediction model can be adapted to the input sequence, it can support the fixed or changing model. Because the kernel-based MPA algorithm and DL-based (or ML-based) methods are developed for different prediction problem settings, we focus on comparing the proposed BMPA algorithm with budgeted and non-budgeted online multiclass algorithms in this paper.

For those practical applications receiving streaming data, the targets are often assumed to be generated from specific target function or distribution. These applications may face the problem of concept drift that results in the change of the target function or distribution over time [40], [41]. On this subject, there are methods proposed in literature to address the adaptation of concept drift over time, e.g., adaptive random forests [42], Kappa updated ensemble [43], and leveraging bagging [44]. Moreover, with the unlearning framework [45], the PA algorithm can be applied on these applications to prevent degradation in the prediction performance. Similarly, the MPA algorithm can be applied for multiclass classification. In this paper, we focus on the development of the budgeted online algorithm that makes the MPA algorithm feasible given limited computational power.

The rest of the paper is organized as follows. We discuss related work in Section II. Section III reviews the learning setting for the MPA algorithm and its kernelization. In Section IV, we present the proposed BMPA algorithm and study three budget maintenance strategies. Section V provides a unified theoretical analysis for the proposed method. We conduct empirical experiments in Section VI and conclude the paper in Section VII.

II. RELATED WORK

In this section, we mainly discuss online classification problems in online learning. For a more comprehensive survey of online learning, please refer to [46].

A. ONLINE LEARNING

Online classification can be further divided into two cases: online binary classification for a sequence of binary classification tasks and online multiclass classification for a sequence of multiclass classification tasks. Since the latter is generally considered a more complicated problem, research works are focused primarily on the former at first.

The well-known Perceptron algorithm is probably the very first online algorithm for the binary case [6]. It performs a simple additive update on the parameters of a linear model whenever it makes a wrong prediction. Several theoretical studies suggest that the number of prediction mistakes made by the Perceptron algorithm can be bounded from above [8], [47], [48]. Crammer *et al.* [10] proposed the passive-aggressive (PA) algorithm that utilizes the notion of margin to update the linear model. If the margin of the current example is smaller than a predefined value, the PA algorithm updates the model so that the new model achieves a unit margin on the current example by solving a constrained optimization problem. Two variants of the PA algorithm, which relax the constraints by trading off between the model change and the desired margin, are also described so that the existence of noise can be taken into consideration, e.g., mislabeled examples. Relative loss bounds for all three variants of the PA algorithm are derived. Due to the prevalence of the PA algorithms, several subsequent works are conducted. Confidence-weighted (CW) learning brings uncertainty into the linear model [25], [28], [49]. The parameter confidence is modeled as a Gaussian distribution, and the CW algorithm updates the distribution by solving a constrained optimization problem mimicking the PA algorithm. Since the CW algorithm performs poorly on nonseparable data and noisy data due to its aggressive update rules, soft confidence-weighted (SCW) learning is proposed to alleviate the situation by trading off between the distribution change and the adaptive margin [29], [50]. Alternatively, adaptive regularization of weights (AROW) employs an adaptive regularization approach for each example according to its confidence [30], [51].

Some researchers paid their attention to online multiclass classification. Lots of efforts were put into how to cleverly turn an online algorithm designed originally for a binary problem to handle a multiclass problem. For example, Crammer and Singer [52] proposed a family of additive ultraconservative algorithms, which generalized the Perceptron algorithm to multiclass problems, and provided unified mistake bounds. Similarly, multiclass extensions for the PA, CW, AROW, and SCW algorithms were developed cleverly one after another [10], [31], [50], [51]. Based on the multiclass PA algorithm, Matsushima *et al.* [32] further proposed the support class PA algorithm that resolves the constraint relaxation through the idea of support classes.

B. KERNEL-BASED ONLINE LEARNING

With the kernel trick, an online algorithm can employ a kernel-based model to achieve usually a better prediction accuracy. However, it suffers from the curse of kernelization that causes unbounded growth in memory usage and runtime [15]; thus, several researchers have tried to address this issue.

For binary problems, most works are focused on controlling the growth of a kernel-based model by limiting the number of support vectors (SVs). The budget Perceptron algorithm proposed by Crammer *et al.* [16] is the first algorithm to limit the number of SVs by a predefined value, which is called the budget. Once the number of SVs reaches the budget, it selects one of the SVs that meets some rule and replaces it by the new instance. A similar strategy is used in the NORMA algorithm [14] and the tighter budget Perceptron algorithm [17]. Dekel *et al.* [18] proposed the Forgetron algorithm that is the first algorithm having a relative mistake bound derived on a budget. The key ingredient making the theoretical analysis possible is the repeated shrinking of the kernel-based model followed by removing the oldest SV every time an update is performed. Later, Cavallanti *et al.* [19] proposed the randomized budget Perceptron algorithm that chooses at random an SV to remove and enjoys an expected mistake bound similarly to the mistake bound of the Forgetron algorithm. A merging approach was proposed to combine two SVs into another new one [53]. Wang and Vucetic [20] proposed the budgeted PA (BPA) algorithm that performs the PA algorithm on a fixed budget through a constrained

TABLE 1. Comparison between BMPA and BPA [20].

Method	Classification Problem	Budget Maintenance	Explanation	Theoretical Support
BPA	Binary	Subset Selection	Heuristics	No
BMPA	Multiclass	Removal Strategy	Approximation	Mistake Bound

optimization problem. Based on the kernel-based Perceptron algorithm, Orabona *et al.* [13] proposed the Projectron algorithm that takes a different route to control the growth of a kernel-based model. It either includes the current instance as an SV or projects the kernel of the instance onto the subspace spanned by kernels centered on SVs if the projection error is small enough. The number of SVs is guaranteed to be bounded yet unknown, and a relative mistake bound is derived. The authors also proposed an improved algorithm called Projectron++ that considers the notion of margin. Instead of focusing on the number of SVs, Lu *et al.* [22] proposed a new framework that turns a kernel-based model into a linear model with kernel-induced feature approximation. Under this framework, they proposed two algorithms with loss bounds based on the OGD algorithm. One is the Fourier OGD algorithm that approximates shift-invariant kernels by using random Fourier features, and the other is the Nyström OGD algorithm that approximates the kernel matrix by using the Nyström method.

In the literature, a few kernel-based methods have been proposed for multiclass problems, which are more complicated than binary problems. Orabona *et al.* [13] proposed a multiclass version of the Projectron++ algorithm and presented the relative mistake bound. However, the number of SVs cannot be known in advance. It may result in the broken-down problem when only a finite amount of computational resources is usable. In [21] and [22], both the Fourier OGD and Nyström OGD algorithms were extended to multiclass problems. For the former, the resulting numbers of features should be large enough to approximate the shift-kernel kernels well; for the latter, the matrix approximation rank should be larger to have a better classification accuracy.

Finally, TABLE 1 summarizes the difference between our proposed BMPA algorithm and the BPA algorithm [20] that motivates our work in this paper. Firstly, BPA is designed to tackle binary problems while BMPA aims for multiclass problems. Secondly, BPA studies which subset of SVs is selected for representing a removed SV and adopts a fixed removal strategy that picks the SV minimizing a regularized loss; BMPA always selects the entire set of SVs for representing a removed SV and studies which SV is selected for removal. Thirdly, as shown in FIGURE 1, we introduce the resource perspective to give an equivalent interpretation of the kernel-based MPA algorithm and provide a solid explanation of how BMPA approximates the kernel-based MPA; however, BPA includes a budget constraint heuristically to limit the maximum number of SVs

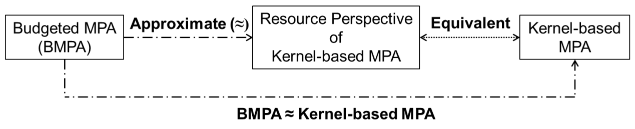


FIGURE 1. A schematic of how BMPA approximates the kernel-based MPA.

without any statement on the approximation. Lastly, BMPA enjoys a relative mistake bound while there is no theoretical support for BPA.

III. PROBLEM SETTING

We consider the problem of online multiclass classification, which is to solve a sequence of multiclass classification tasks given the knowledge of correct class labels of previous tasks. On the t-th task round, the learner first receives an instance $x_t \in \mathcal{X}$ and predicts its class $\widehat{y}_t \in \mathcal{Y} = \{1, 2, \ldots, m\}$, $m > 2$, based on some prediction rules. After receiving the correct class label y_t, the learner has to decide whether to update the prediction rules such that future tasks may be done well without the knowledge of future tasks. The goal is to make correct predictions as many as possible for this sequence. In the rest of this section, we review the multiclass passive-aggressive (MPA) algorithm and its kernelization. The kernel-based MPA algorithm will serve as our basics to design the budgeted algorithm that can be used in resource-insufficient occasions, e.g., performing online multiclass classification on smart devices with limited computational power.

A. MPA ALGORITHM

To tackle the problem of online multiclass classification, MPA employs an m-class discriminant comprising m linear hypotheses of the following inner product form [10],

$$f^s(x) = \boldsymbol{w}^s \cdot \boldsymbol{\phi}(x), \quad \forall s \in \mathcal{Y}, \tag{1}$$

where $\mathcal{Y} = \{1, 2, \ldots, m\}$ is the set of m class labels and $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathbb{R}^d$ represents feature extraction that transforms instances to a desired feature space. $f^s : \mathcal{X} \rightarrow \mathbb{R}$ measures the score of class s for an instance and is parameterized by the weight vector $\boldsymbol{w}^s \in \mathbb{R}^d$. The process of MPA for a task round can be summarized in four steps: scoring, prediction, evaluation, and update.

On round t, MPA first computes the scores of all classes after receiving the instance x_t,

$$f_t^s(x_t) = \boldsymbol{w}_t^s \cdot \boldsymbol{\phi}(x_t), \quad \forall s \in \mathcal{Y}, \tag{2}$$

where $f_t^s(\cdot) = \boldsymbol{w}_t^s \cdot \boldsymbol{\phi}(\cdot)$ measures the score of class s at round t. Then, MPA predicts the class with the highest score

$$\widehat{y}_t = \arg\max_{s \in \mathcal{Y}} f_t^s(x_t). \tag{3}$$

After receiving the true class label y_t, the corresponding prediction mistake can be determined by $\mathbb{I}(\widehat{y}_t \neq y_t)$ where $\mathbb{I}(\cdot)$ is an indicator function.

Another popular way to evaluate the prediction is using the hinge loss function, which is defined as

$$\ell\left(\{f^s\}_{s=1}^m; (x, y)\right) = \max\{0, 1 - [f^y(x) - \max_{s\in\mathcal{Y}, s\neq y} f^s(x)]\}, \tag{4}$$

which penalizes the prediction if the margin of the discriminant, $f^y(x) - \max_{s\in\mathcal{Y},s\neq y} f^s(x)$, is less than 1. At round t, MPA evaluates the prediction of the discriminant parameterized with $\{\boldsymbol{w}_t^s\}_{s=1}^m$ by computing the hinge loss

$$\ell\left(\{\boldsymbol{w}_t^s\}_{s=1}^m; (x_t, y_t)\right) = \max\{0, 1 - [\boldsymbol{w}_t^{y_t} \cdot \boldsymbol{\phi}(x_t) - \boldsymbol{w}_t^{s_t} \cdot \boldsymbol{\phi}(x_t)]\}, \tag{5}$$

where s_t is the most misleading class on round t,

$$s_t = \arg\max_{s\in\mathcal{Y}, s\neq y_t} f_t^s(x_t). \tag{6}$$

Instead of $\ell\left(\{f_t^s\}_{s=1}^m; (x_t, y_t)\right)$, here we slightly abuse the notation by using $\ell\left(\{\boldsymbol{w}_t^s\}_{s=1}^m; (x_t, y_t)\right)$ since f_t^s is parameterized by $\boldsymbol{w}_t^s$.

At the end of round t, MPA solves the following constrained optimization problem for the updated determinant parameterized by $\{\boldsymbol{w}_{t+1}^s\}_{s=1}^m$,

$$\begin{aligned} \underset{\boldsymbol{w}^s, s\in\mathcal{Y}}{\text{minimize}} \quad & \frac{1}{2}\sum_{s=1}^m \|\boldsymbol{w}^s - \boldsymbol{w}_t^s\|_2^2 \\ \text{subject to} \quad & \boldsymbol{w}^{y_t} \cdot \boldsymbol{\phi}(x_t) - \boldsymbol{w}^{s_t} \cdot \boldsymbol{\phi}(x_t) \geq 1, \end{aligned} \tag{7}$$

which requires the update to have minimum change while only having to achieve enough score difference between the correct class y_t and the most misleading class s_t on the current example (x_t, y_t).[1] Note that this optimization problem is a relaxation because it considers only the single incorrect class s_t instead of all incorrect classes [10]. According to (4), it does not guarantee that the updated discriminant has zero hinge loss on the current example. (7) has a closed-form solution, and the resulting update rule is as follows

$$\boldsymbol{w}_{t+1}^s = \boldsymbol{w}_t^s + \alpha_t^s \boldsymbol{\phi}(x_t), \quad \forall s \in \mathcal{Y}, \tag{8a}$$

$$\alpha_t^s = (\delta_{s,y_t} - \delta_{s,s_t})\tau_t, \tag{8b}$$

$$\tau_t = \frac{\ell\left(\{\boldsymbol{w}_t^s\}_{s=1}^m; (x_t, y_t)\right)}{2\|\boldsymbol{\phi}(x_t)\|_2^2}, \tag{8c}$$

where $\delta_{a,b}$ is the Kronecker delta

$$\delta_{a,b} = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{if } a \neq b. \end{cases} \tag{9}$$

This means only weight vectors of classes y_t and s_t are modified by adding or subtracting the scaled feature vector, $\tau_t\boldsymbol{\phi}(x_t)$, if the prediction suffers nonzero hinge loss, otherwise all weight vectors remain unchanged. It should be noted that the update requires to explicitly compute the feature vector $\boldsymbol{\phi}(x_t)$ and thus the feature space is restricted to be finite-dimensional.

[1] In this paper, we focus on the version without a slack variable.

B. KERNEL-BASED MPA ALGORITHM

Through the kernel trick, MPA can be kernelized to make predictions with a kernel-based m-class discriminant. To see this, we first represent the weight vector associated with the class s on round t as a linear combination of feature vectors of support elements (SEs) as follows

$$\begin{aligned} \boldsymbol{w}_t^s &= \boldsymbol{w}_{t-1}^s + \alpha_{t-1}^s \boldsymbol{\phi}(x_{t-1}) \\ &= \left(\boldsymbol{w}_{t-2}^s + \alpha_{t-2}^s \boldsymbol{\phi}(x_{t-2})\right) + \alpha_{t-1}^s \boldsymbol{\phi}(x_{t-1}) \\ &= \ldots \\ &= \sum_{i\in\mathcal{I}_t} \alpha_i^s \boldsymbol{\phi}(x_i), \end{aligned} \tag{10}$$

where $\boldsymbol{w}_1^s$ is initialized as the zero vector for all classes and $\mathcal{I}_t = \left\{i \in \mathbb{N} \mid i < t \wedge \max_{s\in\mathcal{Y}} |\alpha_i^s| \neq 0\right\}$ is the support index set which collects indices of SEs. In this paper, an instance $x_i \in \mathcal{X}$ used to construct a prediction model is called an SE instead of a support vector because we suppose that x_i can be anything, e.g., a document of varied length, not just a fixed-length vector. In literature, an SE is called a support vector because it is a fixed-length vector.

The kernelization of MPA is to replace the inner product on the feature space, $\boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x')$, by a kernel function $k(x, x')$ that implements the inner product implicitly,

$$k(x, x') = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x'). \tag{11}$$

Thus, we can represent all linear hypotheses of the m-class discriminant on round t as kernel-based hypotheses simultaneously

$$f_t^s(\cdot) = \sum_{i\in\mathcal{I}_t} \alpha_i^s k(x_i, \cdot), \quad \forall s \in \mathcal{Y}, \tag{12}$$

which implicitly carries out the feature extraction through the kernel function k. Prediction of the class and evaluation of the hinge loss are computed by (3) and (4) using (12) and the current example (x_t, y_t). The update rule of the kernel-based MPA is re-organized as follows (cf. (8a)–(8c))

$$f_{t+1}^s(\cdot) = f_t^s(\cdot) + \alpha_t^s k(x_t, \cdot), \forall s \in \mathcal{Y}, \tag{13a}$$

$$\alpha_t^s = (\delta_{s,y_t} - \delta_{s,s_t})\tau_t, \tag{13b}$$

$$\tau_t = \frac{\ell\left(\{f_t^s\}_{s=1}^m; (x_t, y_t)\right)}{2k(x_t, x_t)}. \tag{13c}$$

Since the kernel-based MPA has to store and use all SEs and the associated combination weights, the computational burden on a single round, in terms of memory usage and runtime, may grow unboundedly as more and more rounds are done. This problem is called the curse of kernelization and demands a resource-efficient algorithm if we need the kernel-based MPA workable in practical applications, especially in resource-insufficient occasions. To solve the curse of kernelization, we propose the budgeted algorithm that limits the number of SEs in use and fully exploit them through a constrained optimization.

IV. KERNEL-BASED MPA ON A FIXED BUDGET

In this section, we first introduce the resource perspective for the kernel-based MPA, and then propose the budgeted MPA (BMPA) algorithm based on this perspective. Finally, we study three SE removal strategies for the budget maintenance.

Since the m-class discriminant adopted by the kernel-based MPA consists of m kernel-based hypotheses, we consider that hypotheses of an m-class discriminant are selected from the *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}$ of a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ [54]. $\mathcal{H}$ is a Hilbert space of real-valued functions $f : \mathcal{X} \to \mathbb{R}$ endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that it satisfies (1) $k(x, \cdot) \in \mathcal{H}, \forall x \in \mathcal{X}$, and (2) the reproducing property, $\langle f, k(x, \cdot)\rangle_{\mathcal{H}} = f(x), \forall x \in \mathcal{X}, \forall f \in \mathcal{H}$. The inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ induces a norm on $\mathcal{H}$ such that $\|f\|_{\mathcal{H}} = \langle f, f\rangle_{\mathcal{H}}^{\frac{1}{2}}, \forall f \in \mathcal{H}$.

A. RESOURCE PERSPECTIVE

According to (12), the kernel-based MPA employs an m-class discriminant comprising of m kernel-based hypotheses. All kernel-based hypotheses on round t are linear combinations of kernels centered on the same set of SEs corresponding to $\mathcal{I}_t$. However, (13a)–(13c) suggest that the update of the kernnel-based MPA includes the current instance x_t as a new SE and only adjusts its combination weights; all weights corresponding to the other SEs stay the same. It implies if we can somehow adjust weights of all available SEs including x_t and get a update rule different from that of the kernel-based MPA.

Now, we introduce the *resource perspective* of a kernel-based online algorithm that treats every encountered instance x_i as a potential resource that may be included as an SE or removed in the online learning process. On round t, SEs are treated as available resources used to construct the prediction model, and the corresponding weights are treated as the degrees of utilization of the SEs. During the update step, the learner first selects which instances to store as available resources (i.e., SEs) for prediction on the next round and determines their degrees of utilization; instances that are not stored as SEs are treated as unavailable resources which are removed and cannot be used on later rounds. The remaining question is whether we can get a different update rule by using the resource perspective and following the update idea of MPA, which asks for the minimum change in the prediction model while achieving enough score difference.

Let us start from the prediction model used by the resource perspective. Assume hypotheses of the m-calss discriminant on round t are some linear combinations of kernels centered on the same set of some SEs,

$$\widetilde{f}_t^s = \sum_{i \in \widetilde{\mathcal{I}}_t} \widetilde{\alpha}_i^s k(x_i, \cdot), \quad \forall s \in \mathcal{Y}, \tag{14}$$

where $\widetilde{\mathcal{I}}_t$ is the support index set which collects corresponding indices of SEs. We use the symbol $\sim$ to emphasize that even with the same sequence of tasks (14) and (12) may be different in SEs and the combination weights.

Prediction of the class and evaluation of the hinge loss are computed by (3) and (4) using (14) and the current example (x_t, y_t). If the hinge loss is zero, we keep the same hypotheses for the next round,

$$\widetilde{f}_{t+1}^s = \widetilde{f}_t^s, \quad \forall s \in \mathcal{Y}. \tag{15}$$

In case the discriminant suffers nonzero hinge loss, i.e., $\ell(\{\widetilde{f}_t^s\}_{s=1}^m; (x_t, y_t)) > 0$, without any constraint on the number of resources the resource perspective suggests to select all available instances including x_t as available resources for prediction on the next round. In other words, we seek for new hypotheses that are some linear combinations of kernels centered on the new set of SEs,

$$\widetilde{f}^s = \sum_{i \in \widetilde{\mathcal{I}}_{t+1}} a_i^s k(x_i, \cdot), \quad \forall s \in \mathcal{Y}, \tag{16}$$

where $\widetilde{\mathcal{I}}_{t+1} = \widetilde{\mathcal{I}}_t \cup \{t\}$ is the new support index set. It leaves us to determine the degrees of utilization a_i^s's.

Then, we follow the update idea of MPA: the degrees of utilization for the next round should result in minimum change in the update of the prediction model while achieving enough score difference on the current example (x_t, y_t),

$$\begin{aligned} \underset{a_i^s, \forall i, \forall s}{\text{minimize}} \quad & \frac{1}{2}\sum_{s=1}^m \|\widetilde{f}^s - \widetilde{f}_t^s\|_{\mathcal{H}}^2 \\ \text{subject to} \quad & \widetilde{f}^{y_t}(x_t) - \widetilde{f}^{\widetilde{s}_t}(x_t) \geq 1, \end{aligned} \tag{17}$$

where $\widetilde{s}_t = \arg\max_{s \in \mathcal{Y}, s \neq y_t} \widetilde{f}_t^s(x_t)$ is the most misleading class determined by using (14) and (x_t, y_t). Substituting the solution of (17) back into (16), we will get the new hypotheses $\{\widetilde{f}_{t+1}^s\}_{s=1}^m$ for the next round. Note that (17) is different from (7) because we solve for the degrees of utilization of available resources in (17) instead of the weight vectors in (7).

Somewhat surprisingly, the resource perspective combining with the update idea of MPA turns out to have the same update rule with the kernel-based MPA. Moreover, if we set the same initialization for both approaches, we will get the same prediction results for the entire sequence of tasks. We state this result formally by the following proposition.

Proposition 1: Assume all Gram matrices of encountered instances in the RKHS $\mathcal{H}$ of a kernel k are strictly positive definite. Let the resource perspective combining the update idea of MPA adopt the following setting.

1) *Initialize the hypotheses of the m-class discriminant to zero, i.e., $\widetilde{f}_1^s = 0, \forall s \in \mathcal{Y}$.*
2) *Predict the class with the highest score on round t,*

$$\hat{y}_t = \arg\max_{s \in \mathcal{Y}} \widetilde{f}_t^s(x_t). \tag{18}$$

3) *Update the hypotheses for round $t + 1$ by solving* (17) *when the hinge loss is positive, otherwise update by* (15).

This setting follows the same prediction path made by the kernel-based MPA, that is, for $t = 1, 2, \ldots$,

$$\widetilde{f}_t^s = f_t^s = \sum_{i \in \mathcal{I}_t} \alpha_i^s k(x_i, \cdot), \quad \forall s \in \mathcal{Y}, \tag{19}$$

where $\{f_t^s\}_{s=1}^m$ are the hypotheses employed by the kernel-based MPA on round t.

We give the proof in the Appendix. Although the resource perspective starts from a different viewpoint, Proposition 1 guarantees that the kernel-based MPA and the resource perspective considering the degrees of utilization of available resources employ the same sequence of m-class discriminants to make predictions. In other words, the resource perspective provides an alternative and equivalent interpretation for the kernel-based MPA. This bring us to the proposed BMPA algorithm that approximates the kernel-based MPA when a limited number of resources is available.

B. BUDGETED MULTICLASS PASSIVE-AGGRESSIVE (BMPA) ALGORITHM

As described in Section III-B, the kernel-based MPA suffers from the curse of kernelization. Fortunately, Proposition 1 suggests that we can approximate the kernel-based MPA based on the re-interpretation from the resource perspective. Therefore, based on the resource perspective, we propose the BMPA algorithm to limit the number of available resources used to construct the kernel-based hypotheses of an m-class discriminant by a predefined number B, called the budget. In practical applications, the budget B can be easily defined in advance by considering the available computing power. Note that a single budget B can be used to control the growth of the prediction model by simultaneously controlling the growth of all kernel-based hypotheses.

To be more clear about how to use the budget B with the resource perspective, again we assume hypotheses on round t are some linear combinations of kernels centered on some SEs,

$$f_t^s = \sum_{i \in \mathcal{I}_t} \alpha_i^s k(x_i, \cdot), \quad \forall s \in \mathcal{Y}, \tag{20}$$

where $\mathcal{I}_t$ is the corresponding support index set. From now on we simplify the notations by neclecting the symbol $\sim$. If on round t the hinge loss is zero or the number of SEs is less than the budget B, the proposed BMPA algorithm performs the update exactly like what the kernel-based MPA does in (13a)–(13c). We call this kind of update the MPA update. Otherwise, we have to deal with the case that the discriminant suffers nonzero hinge loss and the number of SEs reaches the budget, i.e., $|\mathcal{I}_t| = B$. BMPA will remove one of current SEs and set the remaining SEs and the current instance x_t as available resources for prediction on the next round. In other words, BMPA seeks for new hypotheses that are some linear combinations of kernels centered on the set of only B SEs,

$$f^s = \sum_{j \in \mathcal{I}_{t+1}} a_j^s k(x_j, \cdot), \quad \forall s \in \mathcal{Y}, \tag{21}$$

where $\mathcal{I}_{t+1} = (\mathcal{I}_t \backslash \{r\}) \cup \{t\}$ corresponds to the B available SEs and $r \in \mathcal{I}_t$ is the index of the removed SE. Then, BMPA determines the degrees of utilization for the next round by solving (17) with (20) and (21). We call this type of update the BMPA update and state it specifically by the following proposition.

Proposition 2: Assume all Gram matrices of encountered instances in the RKHS $\mathcal{H}$ of a kernel k are strictly positive definite. If on round t the discriminant suffers nonzero hinge loss and reaches the budget, $|\mathcal{I}_t| = B$, BMPA updates the hypotheses as follows

$$f_{t+1}^s = f_t^s - \alpha_r^s k(x_r, \cdot) + \sum_{j \in \mathcal{I}_{t+1}} \beta_j^s k(x_j, \cdot), \quad \forall s \in \mathcal{Y}, \tag{22a}$$

$$\boldsymbol{\beta}^s = \alpha_r^s \boldsymbol{K}_r^{-1} \boldsymbol{k}_r + \begin{bmatrix} \boldsymbol{0}_{B-1} \\ (\delta_{s,y_t} - \delta_{s,s_t})\tau_t \end{bmatrix}, \tag{22b}$$

$$\tau_t = \frac{\ell(\{f_t^s\}_{s=1}^m; (x_t, y_t))}{2k(x_t, x_t)}, \tag{22c}$$

where $r \in \mathcal{I}_t$, $\mathcal{I}_{t+1} = (\mathcal{I}_t \backslash \{r\}) \cup \{t\}$, $\boldsymbol{\beta}^s = [\beta_j^s], j \in \mathcal{I}_{t+1}$, $k_{ab} = k(x_a, x_b), a, b \in \mathbb{N}$, $\boldsymbol{K}_r = [k_{ij}], i, j \in \mathcal{I}_{t+1}$, and $\boldsymbol{k}_r = [k_{jr}], j \in \mathcal{I}_{t+1}$. $\boldsymbol{0}_{B-1}$ is $(B-1)$-dimensional vector of zeros.

The proof is given in the Appendix. It is worth to note that (22a) of the BMPA update can be decomposed into two parts as follows

$$f_{t+1}^s = (f_t^s + (\delta_{s,y_t} - \delta_{s,s_t})\tau_t k(x_t, \cdot)) \tag{23a}$$

$$+ > (-\alpha_r^s k(x_r, \cdot) + \mathcal{P}_t[\alpha_r^s k(x_r, \cdot)]), \tag{23b}$$

where $\mathcal{P}_t[f]$ is the orthogonal projection of f onto the subspace span($\{k(x_j, \cdot) \mid j \in \mathcal{I}_{t+1}\}$). The first part (23a) can be interpreted as an MPA update (cf. (13a)–(13c)) and the second part (23b) corresponds to the projection of the removed SE x_r. Although in this subsection we start from removing x_r and then determine the degrees of utilization of B available SEs, (23a)–(23b) suggest that the BMPA update can be interpreted in another way: the BMPA update first performs the MPA update and then remove the unaffordable resource x_r by projection. The projection $\mathcal{P}_t[\alpha_r^s k(x_r, \cdot)]$ preserves the information of x_r, so it minimizes the information loss in the prediction model when x_r is removed. We summarize the proposed BMPA algorithm in Algorithm 1.

C. BUDGET MAINTENANCE

Although the proposed BMPA algorithm requests that the index of the removed SE on round t should be selected from current support index set $\mathcal{I}_t$ without any other specific rule, in practice the determination of the removed SE plays an important role. In this subsection, we study three removal strategies for the budget maintenance.

1) OLDEST REMOVAL (BMPA-O) [55]

Since online multiclass classification deals with multiclass classification tasks one after another, adjacent tasks may usually be more relevant that those far apart. In other words,

Algorithm 1 BMPA Algorithm

```
1:  Input: budget $B$, kernel $k$
2:  Initialize: $\mathcal{I}_1 = \emptyset$, hypotheses $f_1^s = 0, \forall s \in \mathcal{Y}$
3:  for $t = 1, 2, \dots$ do
4:      Receive an instance $x_t$
5:      Predict the class label $\hat{y}_t = \arg\max_{s \in \mathcal{Y}} f_t^s(x_t)$
6:      Receive the correct class label $y_t$
7:      Compute the hinge loss $\ell_t = \ell(\{f_t^s\}_{s=1}^m; (x_t, y_t))$
8:      if $\ell_t = 0$ then                                ▷ MPA update
9:          $\mathcal{I}_{t+1} = \mathcal{I}_t$
10:         $f_{t+1}^s = f_t^s, \forall s \in \mathcal{Y}$
11:     else if $|\mathcal{I}_t| < B$ then                   ▷ MPA update
12:         $\mathcal{I}_{t+1} = \mathcal{I}_t \cup \{t\}$
13:         Compute $\tau_t = \frac{\ell_t}{2k(x_t, x_t)}$
14:         $f_{t+1}^s = f_t^s + (\delta_{s,y_t} - \delta_{s,s_t})\tau_t k(x_t, \cdot), \forall s \in \mathcal{Y}$
15:     else                                               ▷ BMPA update
16:         Determine the removed index $r = r_t \in \mathcal{I}_t$
17:         $\mathcal{I}_{t+1} = (\mathcal{I}_t \backslash \{r\}) \cup \{t\}$
18:         Update $f_{t+1}^s$ by (22a)–(22c)
19:     end if
20: end for
```

older tasks may contain less information to make accurate predictions for future tasks. This means the oldest SE may contain the least information for future predictions. It suggests removing the oldest SE to maintain the budget,

$$r_t = \min \mathcal{I}_t, \tag{24}$$

which is the smallest element in $\mathcal{I}_t$.

The main computational burden of BMPA is located at the case where a BMPA update is required. The time complexity of BMPA-O is dominated by the computation of the matrix inverse $\boldsymbol{K}_{r_t}^{-1}$. Computing $\boldsymbol{K}_{r_t}^{-1}$ directly from $\boldsymbol{K}_{r_t}$ will take $\mathcal{O}(B^3)$ time and is not efficient. Instead, we use a recursive approach to compute the matrix inverse by exploiting the decremental and incremental natures of the Gram matrix. We compute $\boldsymbol{K}_{r_t}^{-1}$ from the previous matrix inverse $\boldsymbol{K}_{r_{t-1}}^{-1}$ in $\mathcal{O}(B^2)$ time by two recursive updates: the first one corresponds to shrink the matrix size to $(B-1) \times (B-1)$ and then the latter one is to enlarge the size back to $B \times B$. Each recursive update takes $\mathcal{O}(B^2)$ time, and therefore the total time complexity of BMPA-O is reduced to $\mathcal{O}(B^2)$. On the other hand, the space complexity of BMPA-O is $\mathcal{O}(B^2)$ because we need to store the matrix inverse, which dominates the main memory cost.

2) PROJECTION REMOVAL (BMPA-P)

Since a BMPA update can be interpreted as an MPA update followed by a projection contributed from the removed SE x_r (cf. (23a)–(23b)), a smaller total projection error leads to less information loss in the prediction model. Note that the magnitude of the total projection error on round t has an analytic form,

$$\sum_{s=1}^{m} \left\| \alpha_r^s k(x_r, \cdot) - \mathcal{P}_t[\alpha_r^s k(x_r, \cdot)] \right\|_{\mathcal{H}}^2 = \left(k_{rr} - \boldsymbol{k}_r^\top \boldsymbol{K}_r^{-1} \boldsymbol{k}_r \right) \sum_{s=1}^{m} (\alpha_r^s)^2. \tag{25}$$

Hence, it suggests removing the SE with the least amount of the total projection error on each round,

$$r_t = \arg\min_{q \in \mathcal{I}_t} \left(k_{qq} - \boldsymbol{k}_q^\top \boldsymbol{K}_q^{-1} \boldsymbol{k}_q \right) \sum_{s=1}^{m} (\alpha_q^s)^2. \tag{26}$$

The time complexity of BMPA-P is dominated by the computation of the matrix inverse $\boldsymbol{K}_q^{-1}$. Moreover, to determine r_t it needs to compute $\boldsymbol{K}_q^{-1}$ for every $q \in \mathcal{I}_t$. Again, we exploit the decremental and incremental natures of the Gram matrix to compute the matrix inverse. Each $\boldsymbol{K}_q^{-1}$ is computed from $\boldsymbol{K}_{r_{t-1}}^{-1}$ in $\mathcal{O}(B^2)$ time by two recursive updates that are used by BMPA-O. Therefore, the total time complexity of BMPA-P is $\mathcal{O}(B^3)$. On the other hand, the space complexity of BMPA-P is $\mathcal{O}(B^2)$ because the storage of $\boldsymbol{K}_{r_{t-1}}^{-1}$ and $\boldsymbol{K}_q^{-1}$ dominates the main memory cost.

3) SMALLEST REMOVAL (BMPA-S)

If the weight associated with some SE is far from zero, e.g., $|\alpha_j^s| \gg 0$ for some class s, the score of class s may change dramatically because of the removal of the associated SE, x_j. Thus, it may degrade the prediction accuracy. If the weights associated with some SE are almost zero, e.g., $\alpha_j^s \approx 0$, $\forall s \in \mathcal{Y}$, we can safely remove the associated SE, x_j, without changing the scores of all classes and the prediction accuracy too much. It implies that the weights associated with the SEs may contain the important information for future predictions. This suggests removing the SE with the smallest sum of squared weights,

$$r_t = \arg\min_{q \in \mathcal{I}_t} \sum_{s=1}^{m} (\alpha_q^s)^2. \tag{27}$$

It is worth noting that BMPA-P becomes BMPA-S if $\left(k_{qq} - \boldsymbol{k}_q^\top \boldsymbol{K}_q^{-1} \boldsymbol{k}_q \right)$ is assumed to be a constant value for all choices of SEs in $\mathcal{I}_t$. It means BMPA-S can be treated as an approximation of BMPA-P.

Compared with BMPA-O, BMPA-S only needs to additionally compute $\sum_{s=1}^{m} (\alpha_q^s)^2$ for all choices of SEs in $\mathcal{I}_t$. The time and space complexities of BMPA-S are still dominated by the matrix inverse $\boldsymbol{K}_{r_t}^{-1}$. Hence, the time and space complexities of BMPA-S are the same as those of BMPA-O, both of which are $\mathcal{O}(B^2)$.

Remark: These removal strategies are feasible plans to remove less important SEs and keep more important SEs for the proposed BMPA algorithm. These strategies are introduced based on the resource perspective proposed in this study.

V. THEORETICAL ANALYSIS

In this section, we give a theoretical analysis of the proposed BMPA algorithm. Specifically, we provide a unified relative mistake bound that is applicable to any SE removal strategy as long as the removed SE is selected properly, i.e., $r_t \in \mathcal{I}_t$, and then conduct an empirical study of the theoretical bound on all three removal strategies studied in Section IV-C.

Let us first re-present the proposed BMPA algorithm. As long as the hinge loss is zero or the number of SEs is less than the budget B, BMPA performs the MPA update. The number of SEs therefore grows with every nonzero hinge loss and eventually reaches the budget B. Once the number of SEs reaches the budget B and the discriminant suffers nonzero hinge loss, BMPA first adds an SE by performing the MPA update, and then it reduces the number of SEs to B by projecting the kernel centered on the removed SE onto the subspace spanned by the kernels centered on the new set of SEs, i.e., $\operatorname{span}(\{k(x_j, \cdot) \mid j \in \mathcal{I}_{t+1}\})$.

Let hypotheses of the m-class discriminant on round t be

$$f_t^s = \sum_{i \in \mathcal{I}_t} \alpha_{i,t}^s k(x_i, \cdot), \quad \forall s \in \mathcal{Y}, \tag{28}$$

where $\mathcal{I}_t = \{i \in \mathbb{N} \mid i < t \wedge \max_{s \in \mathcal{Y}} |\alpha_{i,t}^s| \neq 0\}$ is the support index set and $s_t = \arg\max_{s \in \mathcal{Y}, s \neq y_t} f_t^s(x_t)$ is the most misleading class. Note that the combination weights are denoted by $\alpha_{i,t}^s$'s instead of α_i^s's because their values may change when the hypotheses are updated. Denote by $\overline{\mathcal{I}}_t$ the index set obtained on round t after applying the MPA update, that is,

$$\overline{\mathcal{I}}_t = \begin{cases} \mathcal{I}_t & \text{if } \ell(\{f_t^s\}_{s=1}^m;\ (x_t, y_t)) = 0, \\ \mathcal{I}_t \cup \{t\} & \text{if } \ell(\{f_t^s\}_{s=1}^m;\ (x_t, y_t)) > 0. \end{cases} \tag{29}$$

Let $\{\overline{f}_t^s\}_{s=1}^m$ denote the corresponding set of hypotheses,

$$\overline{f}_t^s = f_t^s + \alpha_{t,t}^s k(x_t, \cdot), \quad \forall s \in \mathcal{Y}, \tag{30}$$

where $\alpha_{t,t}^s = (\delta_{s,y_t} - \delta_{s,s_t})\tau_t$ and $\tau_t = \frac{\ell(\{f_t^s\}_{s=1}^m; (x_t, y_t))}{2k(x_t, x_t)}$. Now, define $\mathcal{I}_{t+1}$ to be

$$\mathcal{I}_{t+1} = \begin{cases} \overline{\mathcal{I}}_t & \text{if } |\overline{\mathcal{I}}_t| \leq B, \\ \overline{\mathcal{I}}_t \backslash \{r_t\} & \text{if } |\overline{\mathcal{I}}_t| = B + 1, \end{cases} \tag{31}$$

where $r_t \in \mathcal{I}_t$ is the index of the removed SE. Note that as long as r_t is selected from $\mathcal{I}_t$, it does not affect the analysis. Then, the corresponding hypotheses $\{f_{t+1}^s\}_{s=1}^m$ are

$$f_{t+1}^s = \begin{cases} \overline{f}_t^s & \text{if } |\overline{\mathcal{I}}_t| \leq B, \\ \overline{f}_t^s - \Delta f_t^s & \text{if } |\overline{\mathcal{I}}_t| = B + 1 \end{cases} \tag{32}$$

with the corresponding projection errors

$$\Delta f_t^s = \alpha_{r_t,t}^s k(x_{r_t}, \cdot) - \mathcal{P}_t[\alpha_{r_t,t}^s k(x_{r_t}, \cdot)], \tag{33}$$

where $\mathcal{P}_t[f]$ is the orthogonal projection of f onto the subspace $\operatorname{span}(\{k(x_j, \cdot) \mid j \in \mathcal{I}_{t+1}\})$. Denote by T the number of rounds in the sequence and by J the set of rounds on which the discriminant suffers nonzero hinge loss, namely, $J = \{t \in \mathbb{N} \mid t \leq T \wedge \ell(\{f_t^s\}_{s=1}^m; (x_t, y_t)) > 0\}$. Note that $\mathcal{I}_{T+1}$ is a subset of J. Now, we are ready to present the relative mistake bound for BMPA.

Theorem 1: Let $\mathcal{H}$ be the RKHS of a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)$ be a sequence of examples where $x_t \in \mathcal{X}$, $y_t \in \mathcal{Y} = \{1, 2, \ldots, m\}$, $m > 2$, and $k(x_t, x_t) = R^2$, $R > 0$, for all t. Then, for any competing m-class discriminant comprising m hypotheses $g^s \in \mathcal{H}$, $s \in \mathcal{Y}$, the number of prediction mistakes made by BMPA on this sequence is bounded above by

$$\left(R\sqrt{2\sum_{s=1}^m \|g^s\|_{\mathcal{H}}^2} + 2\sqrt{\sum_{t=1}^T \left[\ell(\{g^s\}_{s=1}^m; (x_t, y_t))\right]^2} \right.$$
$$\left. + 2R\sqrt{\left(\sum_{s=1}^m \|g^s\|_{\mathcal{H}}^2\right)^{\frac{1}{2}} \sum_{t \in J : |\mathcal{I}_t| = B} \left(\sum_{s=1}^m \|\Delta f_t^s\|_{\mathcal{H}}^2\right)^{\frac{1}{2}}} \right)^2. \tag{34}$$

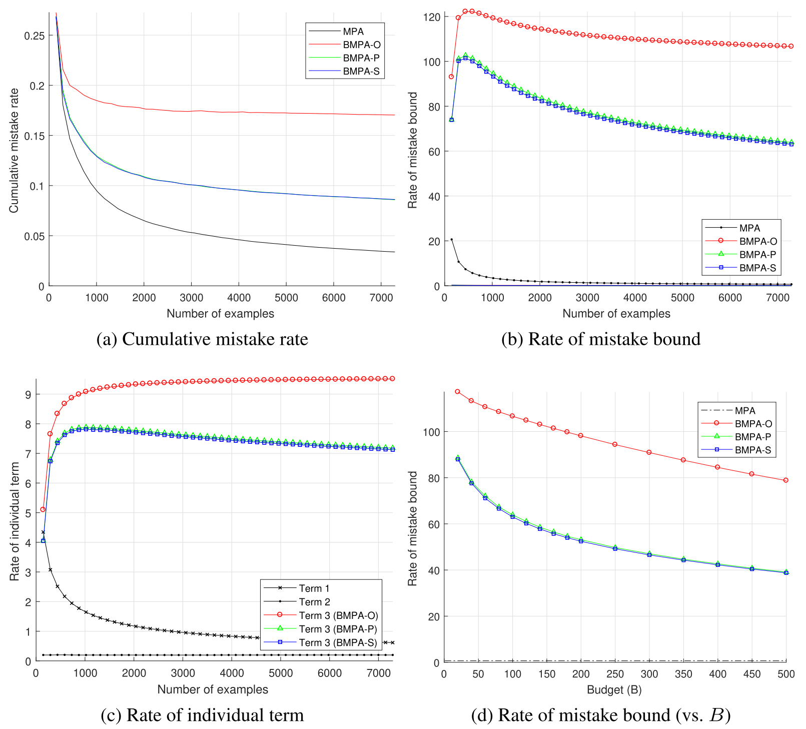
We give the proof in the Appendix. This bound is an upper bound measuring the prediction performance of BMPA relatively to any competing m-class discriminant comprising m hypotheses selected from the RKHS $\mathcal{H}$ of a kernel k. As long as r_t is selected from $\mathcal{I}_t$, the analysis is applicable no matter what removal strategy is executed when the budget is full.

The bound mainly consists of three terms. The first term measures the size of the competing discriminant, the second term evaluates the prediction performance of the competing discriminant applied to the entire sequence, and the third term assesses the influence of the projection errors performed by the proposed BMPA algorithm as well as the size of the competing discriminant. If either the budget is not reached, i.e., $|\mathcal{I}_t| < B$, or there is no budget constraint at all, i.e., $B = \infty$, BMPA always performs the MPA updates. Δf_t^s's become zero, and the theorem states the same mistake bound for the kernel-based MPA [10]. If the budget of BMPA is finite and reached, BMPA performs the BMPA update to remove an SE for every nonzero hinge loss. The effect of SE removal contributes to the projection errors in the third term of the mistake bound and thus makes the bound less tight.

Since the competing m-class discriminant can be chosen in hindsight from the RKHS of the kernel k, one typical choice to gain some insight is to choose the m fixed hypotheses that achieve the best prediction performance for the entire sequence of examples. In this case, the second term has the least influence on the bound and the effect of the first term diminishes gradually. Moreover, the third term involving projection errors dominates the mistake bound. This suggests that the removal strategy plays an important role for determining the prediction performance of BMPA.

It is worth to note that we can take one step of the proof back and have a relative loss bound for BMPA, although the theorem states the relative mistake bound. The loss bound is the same as the mistake bound as suggested by the last line of the proof in the Appendix.

In literature, there are a few online algorithms that break the curse of kernelization for multiclass problems and have theoretical analyses. The Projectron++ algorithm enjoys a



(a) Cumulative mistake rate (b) Rate of mistake bound

(c) Rate of individual term (d) Rate of mistake bound (vs. B)

FIGURE 2. Empirical study of the relative mistake bound on the *usps* dataset. (a)–(c) are conducted with $B = 100$. $\{g^s\}_{s=1}^m$ are chosen such that they minimize the cumulative squared hinge loss of the sequence of T examples. The rate of individual term is defined in the context.

relative mistake bound controlled by a sparseness parameter η [13]; however, the MFOGD and MNOGD algorithms have regret bounds that are fallen in a slightly different realm [22].

Remark 1: Examples of kernels satisfying the condition are the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$ and the exponential kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2)$ where $\gamma > 0$ and $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X} \subseteq \mathbb{R}^d$ [38]. Note that $R = 1$ for both examples.

Remark 2: Intuitively, BMPA-P attempts to reduce the total projection errors in a greedy way on each round. As the implication from the theoretical analysis, BMPA-P may achieve a bound tighter than BMPA-O and BMPA-S.

A. EMPIRICAL STUDY OF THE THEORETICAL BOUND

Because the mistake bound described in Theorem 1 is obtained by using inequalities in several times, it is by no means tight. However, we still can gain some insights through the following empirical study in which only the usps dataset is used to illustrate the results. The other datasets have similar tendencies as that demonstrated with the usps dataset. Unless specifically mentioned, we follow the same experimental setting described in Section VI. The m competing hypotheses $\{g^s\}_{s=1}^m$ are chosen such that they minimize the cumulative squared hinge loss of the sequence of T examples.

FIGURE 2 demonstrates the results of three removal strategies described in Section IV-C as well as the kernel-based MPA algorithm, which is abbreviated as MPA. We set $B = 100$ for FIGURE 2(a)–2(c). FIGURE 2(a) depicts the cumulative mistake rate and FIGURE 2(b) shows the rate of mistake bound, which is the mistake bound divided by the number of examples. We also plot curves of FIGURE 2(a) to FIGURE 2(b): they cannot be distinguished visually and are very close to the horizontal axis. Although FIGURE 2(b) shows that the mistake bounds are not tight at all, the bounds indeed show a relationship same as that of the cumulative mistake rates: BMPA-O $>$ BMPA-P $\approx$ BMPA-S $>$ MPA. To see the interaction of three terms shown in the mistake bound, we compare in FIGURE 2(c) the rate of individual term, which is the square-root term divided by the square root of the number of examples. To be more specific, we define, after round t':

$$\text{rate of Term 1} = \frac{R\sqrt{2\sum_{s=1}^{m} \|g^s\|_{\mathcal{H}}^2}}{\sqrt{t'}}, \tag{35a}$$

$$\text{rate of Term 2} = \frac{2\sqrt{\sum_{t=1}^{t'} \left[\ell(\{g^s\}_{s=1}^m; (x_t, y_t))\right]^2}}{\sqrt{t'}}, \tag{35b}$$

$$\text{rate of Term 3} = \frac{2R\sqrt{\left(\sum_{s=1}^{m} \|g^s\|_{\mathcal{H}}^2\right)^{\frac{1}{2}} \sum_{t \in J' : |\mathcal{I}_t| = B} \left(\sum_{s=1}^{m} \|\Delta f_t^s\|_{\mathcal{H}}^2\right)^{\frac{1}{2}}}}{\sqrt{t'}}, \tag{35c}$$

TABLE 2. Detailed information of datasets ($\mathcal{X} \subseteq \mathbb{R}^d$).

Dataset[a]	#examples (T)	#features (d)	#classes (m)	$\log_2 \gamma$
vehicle	846	18	4	0
dna	2,000	180	3	−4
segment	2,310	19	7	4
satimage	4,435	36	6	2
usps	7,291	256	10	−4
letter	15,000	16	26	4
protein	17,766	357	3	−2
mnist	60,000	780	10	−4

[a]Repository: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

where $J' = \{t \in \mathbb{N} \mid t \leq t' \wedge \ell(\{f_t^s\}_{s=1}^m; (x_t, y_t)) > 0\}$ records rounds with nonzero hinge loss so far. FIGURE 2(c) shows that the first term diminishes gradually, the second term has the least influence, and the third term dominates the bound. This confirms our previous claim and suggests that the removal strategy is important in determining the prediction performance. Lastly, FIGURE 2(d) verifies that the relationship remain among different budgets (cf. FIGURE 4(e)).

Remark: Intuitively, if we can make the mistake bound tighter, the number of prediction mistakes may be fewer. Since a BMPA update consists of an MPA update and a projection contributed by the removed SE, the corresponding projection error contributes to the third term of the mistake bound and thus make the bound less tight. This suggests us to find a way to minimize the total projection error term in the bound, $\sum_{t \in J: |\mathcal{I}_t| = B} \left(\sum_{s=1}^m \|\Delta f_t^s\|_{\mathcal{H}}^2 \right)^{\frac{1}{2}}$, to have the tightest bound. The tightest bound can be achieved only if we can access the whole sequence of examples at one time; it contradicts to the nature of online multiclass classification because we cannot anticipate what future tasks are when we deal with the current task. Therefore, BMPA-P serves as a turnaround to make the bound tighter by minimizing the mistake bound so far. The results in Section VI-B show that BMPA-P achieves the best prediction performance among the three removal strategies studied in this paper. However, in practical applications, we may take the execution time into account. This suggests that BMPA-S is better employed for practical applications (cf. FIGURE 5).

VI. EXPERIMENTS

We conduct experiments to demonstrate the effectiveness of the proposed BMPA algorithm. Overall, there are three goals to achieve through the experimental results. Firstly, we validate that BMPA indeed approximates the kernel-based MPA algorithm, which is abbreviated as MPA in this section. Secondly, different removal strategies of BMPA are compared. Finally, we show that BMPA is competitive with state-of-the-art budget online algorithms.

Table 2 summarizes the detailed information of datasets used in the experiments. All of them can be downloaded from the LIBSVM website. $\mathcal{X} \subseteq \mathbb{R}^d$ is adopted since examples in a dataset are of the fixed-length vector form of d dimensions. For each dataset, we generate 20 different sequences by using random permutation of all T examples in the dataset; cumulative mistake rate and elapsed time are evaluated. All the results are obtained by averaging over the 20 sequences for all experiments. All tested algorithms are kernel-based methods with the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\right)$ where the parameter γ is selected from $\{2^{-10}, 2^{-8}, \ldots, 2^{10}\}$. To have a fair comparison, we use the multiclass version of the Perceptron algorithm with max-score update (abbreviated as MPerceptron) [52] to determine the value of γ. The determination rule is similar to that used in [13]. Specifically, for each dataset, the parameter γ is selected to have the smallest average cumulative mistake rate with MPerceptron and is used for all the other algorithms. All algorithms are implemented in MATLAB and simulated by MATLAB R2017a in a personal computer equipped with 16GB RAM and Intel Core i7-4790 CPU at 3.6GHz. The operating system is Windows 7 Professional 64-bit. For the elapsed time measured in Section VI-B, we enable single-thread processing.

Remark: The kernel-based PA algorithm suffers from the curse of kernelization that may fail to work when t is very large, and is remedied by BPA as demonstrated with a large-scale dataset of one million examples in the experiments section of [20]. The kernel-based MPA algorithm, which is the multiclass extension of the kernel-based PA algorithm, also suffers from the curse of kernelization and is fixed by the proposed BMPA algorithm. Thus, BMPA can be treated as the multiclass extension of BPA. So, BMPA has the same space complexity and m times of the time complexity in comparison with BPA. Both time and space complexities are constant once the budget B is set. Therefore, BMPA is feasible for large-scale datasets.

A. APPROXIMATION OF MPA

We use the oldest removal strategy to verify the approximation; however, the observations are also applicable to the other strategies. FIGURE 3 demonstrates the cumulative mistake rate as a function of the number of examples for MPA and BMPA-O. To properly invoke BMPA updates of BMPA-O, the values of budgets are carefully chosen to be much smaller than the average number of total SEs for MPA. For each dataset, the average is taken over the 20 sequences; for each sequence, we record the number of total SEs used by MPA after receiving all T examples. We also include BMPA-O with the budget equal to the average of MPA.

The mistake rate of MPA drops when the number of examples increases and is almost always the lowest as compared with BMPA-O. The mistake rate of BMPA-O at the beginning is the same as that of MPA since the number of SEs at that time is smaller than the predefined budget. When later the number of SEs reaches the budget, the drop of the mistake rate becomes slower. Moreover, the curve of BMPA-O with a larger budget is lower and is much closer to that of MPA. Eventually, BMPA-O and MPA are not distinguishing if the budget is large enough. We also plot the mistake rate versus the budget for BMPA-O in FIGURE 4. It is evident that

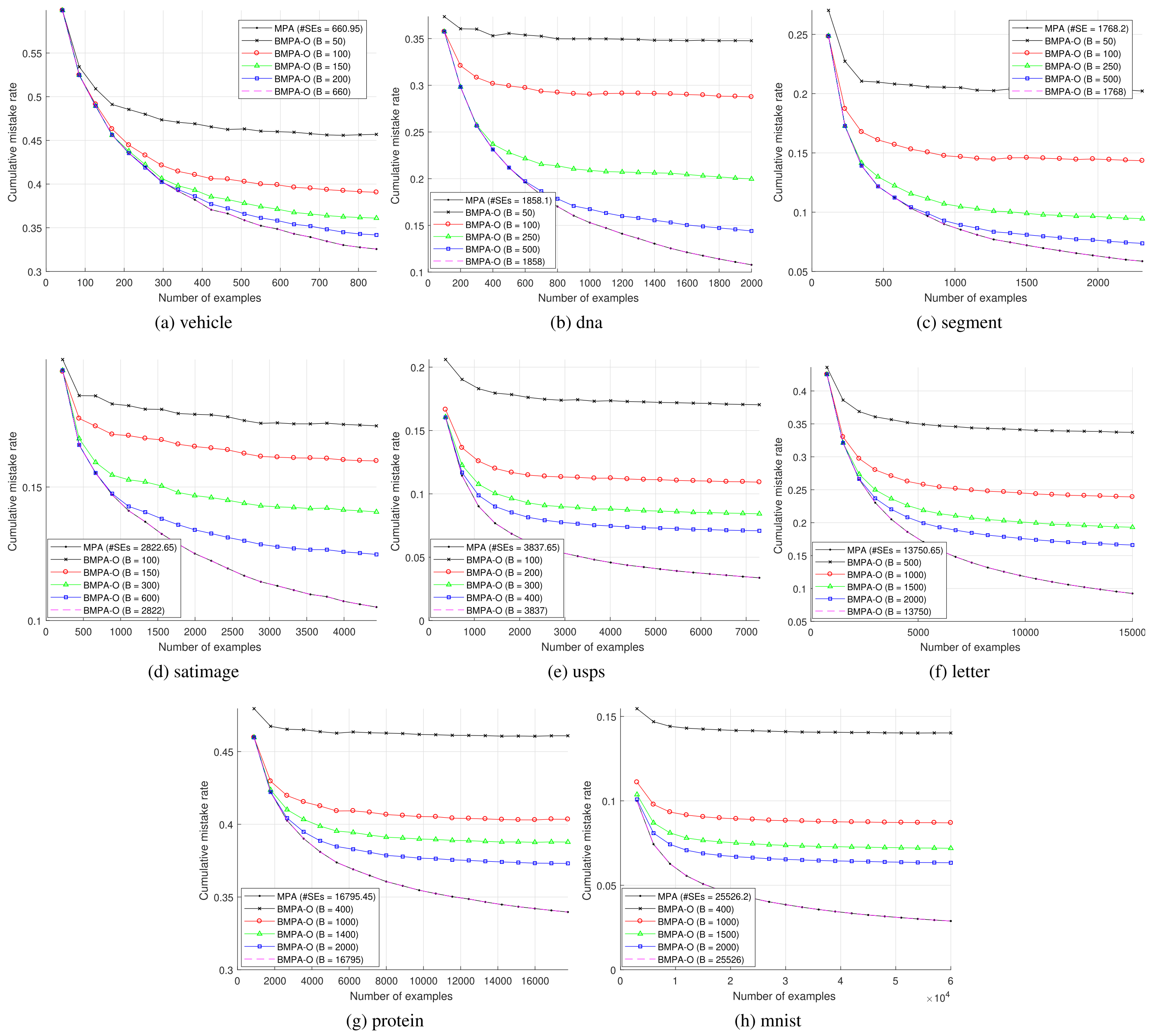


FIGURE 3. The cumulative mistake rate versus the number of examples for MPA and BMPA-O. The average number of total SEs used by MPA is provided for each individual dataset. To properly invoke BMPA updates of BMPA-O, the values of budgets are set to be much smaller than the average of MPA. BMPA-O with the budget equal to the average of MPA is also included for comparison.

the mistake rate converges gradually to that of MPA with the increase of the budget. We conclude that BMPA is an approximation of MPA.

B. REMOVAL STRATEGIES

Because the removal strategy plays a central role in practice, here we compare the performance of the three proposed strategies. FIGURE 4 depicts the cumulative mistake rate as a function of the budget B for MPA and BMPA. Since there is no SE removal in MPA, its mistake rate is plotted as a horizontal dash-dotted line. It is clear that the mistake rate for all strategies converges gradually, at a different pace, to that of MPA as the budget increases. This result supports our claim that BMPA is an approximation of MPA. As expected from the relative mistake bound, BMPA-P has almost the lowest mistake rate among three proposed strategies. Moreover, BMPA-S has a slightly worse or similar performance as compared with BMPA-P. This somehow justifies that BMPA-S approximates BMPA-P.

We also plot in FIGURE 5 the elapsed time as a function of the mistake rate for MPA and BMPA. FIGURE 5 shows that all three removal strategies trade the mistake rate for the elapsed time. Among them, BMPA-S achieves the best trade-off, and the gap in the elapsed time is more tremendous on a lower mistake rate. Furthermore, BMPA runs more efficiently for sequences with a large number of examples and high dimensions, e.g., *protein* and *mnist*. We can observe that their

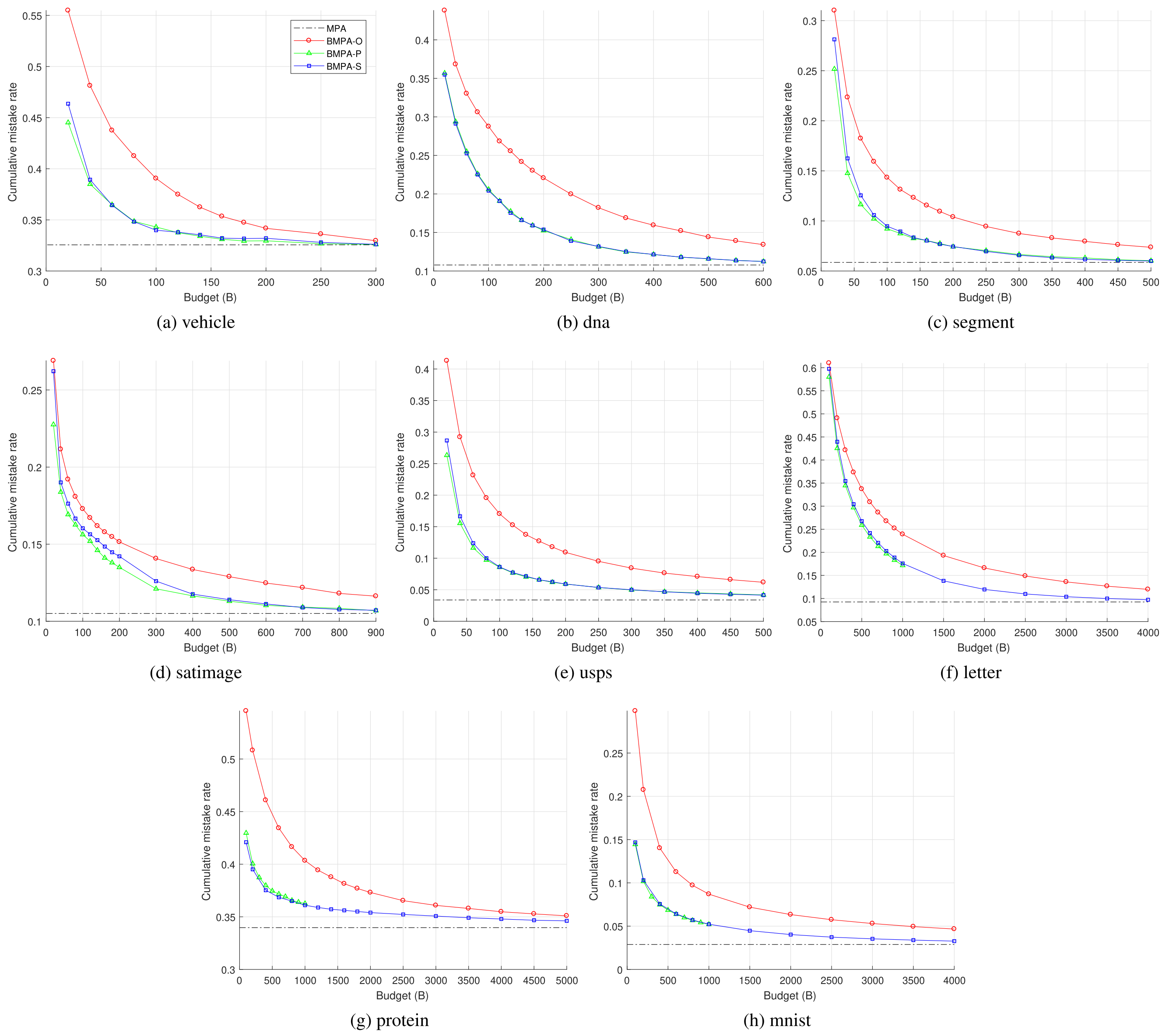


FIGURE 4. The cumulative mistake rate versus the budget for MPA and BMPA with different removal strategies. The dotted horizontal line corresponding to MPA is provided as the baseline for comparison.

curves of BMPA are mostly located at the bottom-right region with respect to MPA.

C. COMPETITIVENESS

We take BMPA-S, which acheves the best trade-off in Section VI-B, as the representative of BMPA and compare it with the following state-of-the-art budgeted online algorithms:

- MRBP: the multiclass randomized budget Perceptron algorithm with random removal strategy [13], [19];
- MProjectron++: the multiclass Projectron++ algorithm using the projection strategy [19];
- MFOGD: the multiclass Fourier online gradient descent algorithm using random Fourier features [22];
- MNOGD: the multiclass Nyström online gradient descent algorithm using Nyström-based features [22].

The above algorithms are essentially designed for online multiclass classification. MRBP represents the multiclass extension of the randomized budget Perceptron algorithm in the nature manner [13], [19]. In particular the max-score update is used. MRBP discards an SE at random from the current set of SEs when the number of SEs reaches the budget and includes the current example as a new SE. Since the discarding of MRBP involves a randomized mechanism, we run it ten times and then take the average for each sequence. To have a more fair comparison to MRBP, we also implement BMPA with random removal (abbreviated as BMPA-R) that selects an SE to remove at random. MProjectron++ is the

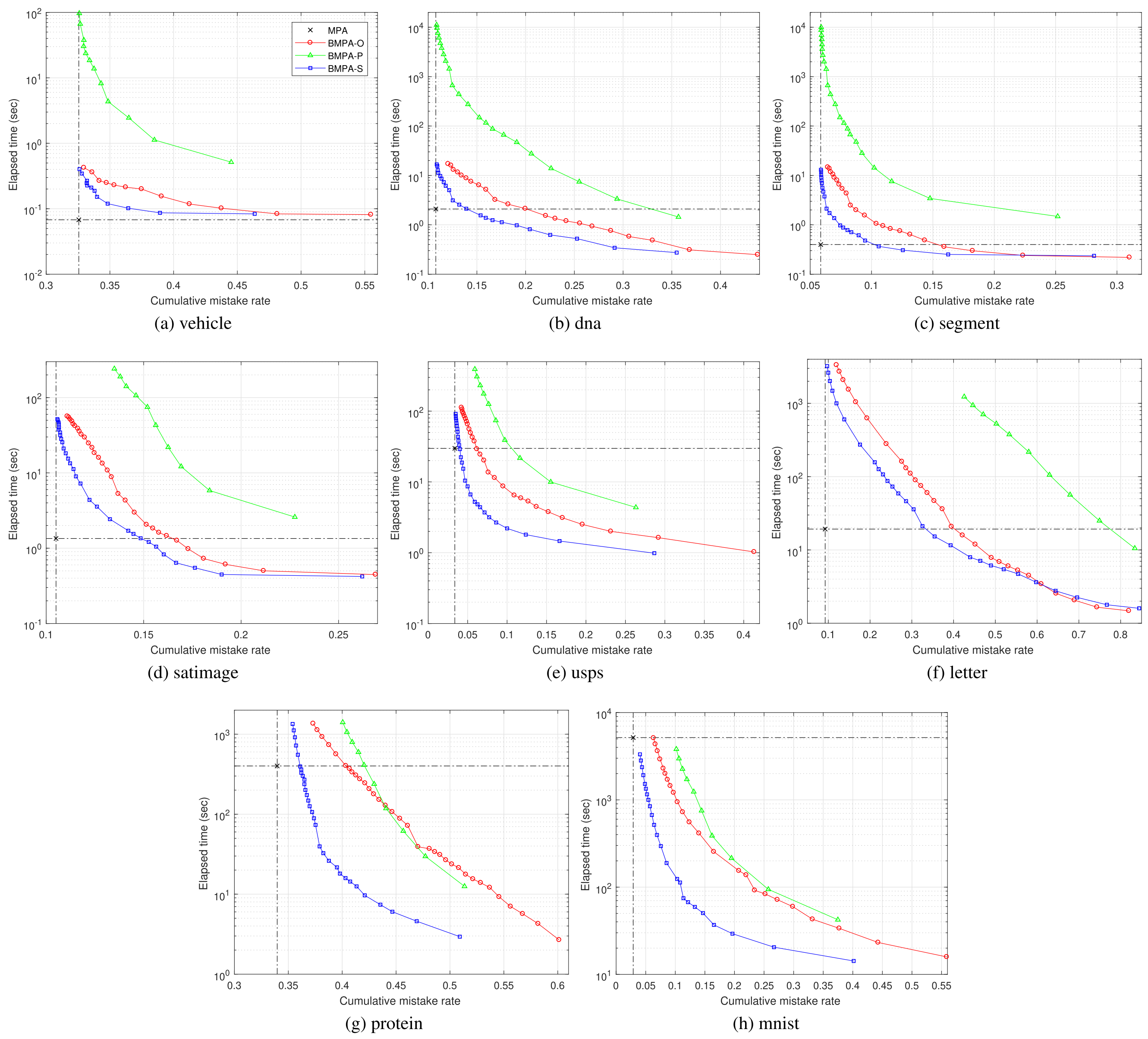
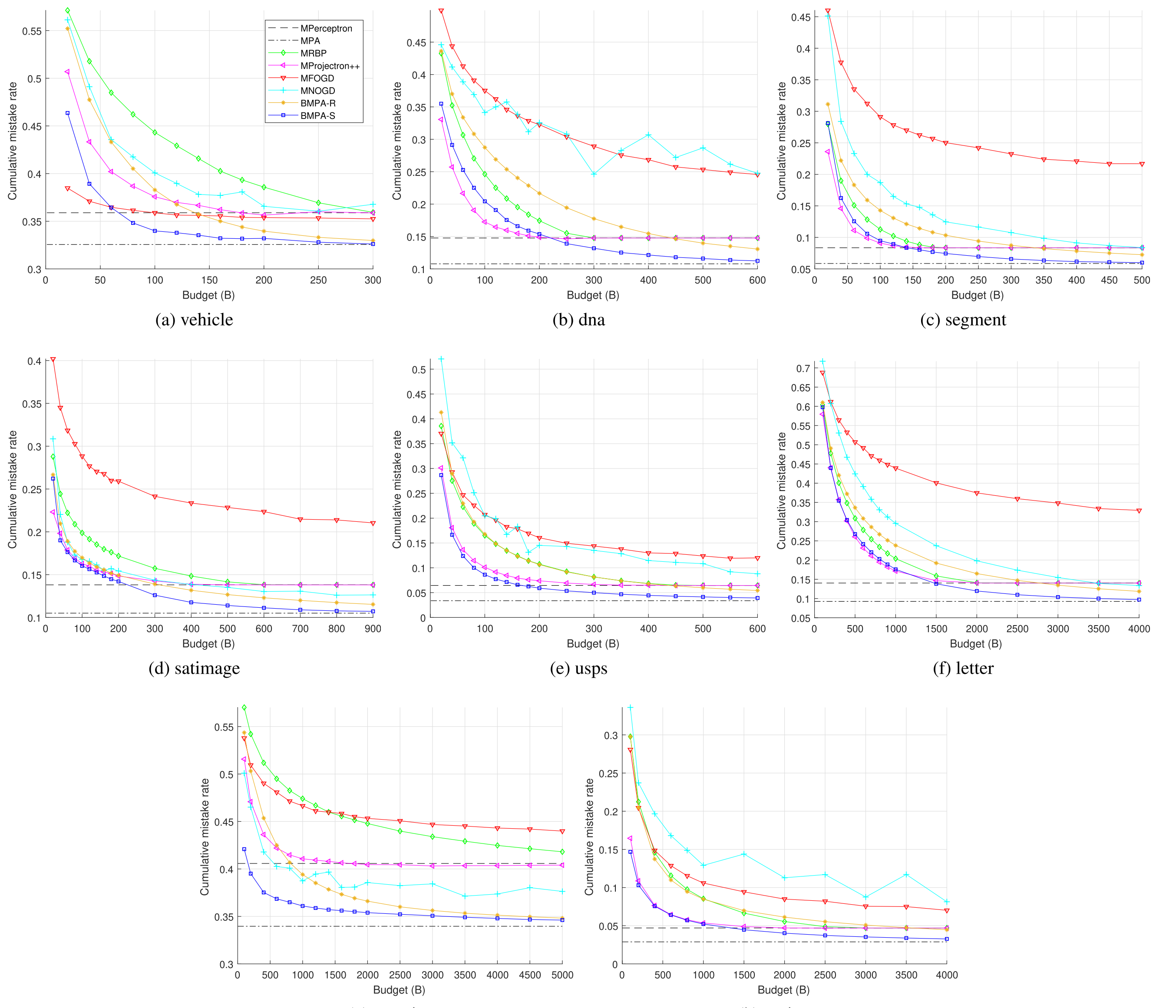


FIGURE 5. The elapsed time versus the cumulative mistake rate for MPA and BMPA with different removal strategies. The cross marker corresponding to MPA is provided as the baseline for comparison.

TABLE 3. Property comparison of of budgeted online algorithms.

Algorithm	Prototype	Hypotheses	Support Element (SE)		
			Removal	Selection	Information Preservation
MRBP	MPerceptron	Kernel-based	Yes	Random	Discarding
MProjectron++	MPerceptron	Kernel-based	Yes	Current	Projection
MFOGD	OGD	Linear	No	-	Kernel-induced Representation
MNOGD	OGD	Linear	No	-	Kernel-induced Representation
BMPA	MPA	Kernel-based	Yes	$r_t \in \mathcal{I}_t$	Projection

multiclass extension of the Projectron++ algorithm [13]. MProjectron++ adopts a projection approach to bound the number of SEs; basically, it projects the kernel centered on the current instance to the subspace spanned by kernels centered on SEs under proper conditions involving a sparseness parameter η. If the number of SEs reaches the predifined budget, the projection is always executed regardless of the conditions. For each budget, η is selected from $\{0.2, 0.4. \dots, 1.4\}$ and is set to the value attaining the lowest mistake rate. MFOGD approximates shift-invariant kernels



(a) vehicle (b) dna (c) segment

(d) satimage (e) usps (f) letter

(g) protein (h) mnist

FIGURE 6. **The cumulative mistake rate versus the budget for budgeted and non-budgeted online multiclass algorithms. The two dotted horizontal lines corresponding to MPerceptron and MPA respectively are provided as baselines for comparison.**

by using random Fourier features and learns the corresponding linear hypotheses by the online gradient descent algorithm [22]. MNOGD approximates the kernel matrix by using the Nyström method and learns the corresponding linear hypotheses by the online gradient descent algorithm [22]. Following the parameter recommendation in [22], we set the number of Fourier components as $6B$ for MFOGD and the rank approximation parameter as $0.4B$ for MNOGD. The gradient descent step size η for both algorithms is drawn randomly from $\{2, 0.2, \ldots, 0.0002\}$. MFOGD is run ten times for each sequence due to the random features. We also include MPerceptron and MPA for comparison. Both algorithms make use of the kernel trick and belong to non-budgeted methods.

As a summary, we summarize the properties of these budgeted online algorithms in TABLE 3. The second column indicates what algorithm prototype they adopt, and the third column records the form of the employed hypotheses. Among the last three columns that are related to SEs, the first one indicates the adoption of a removal strategy, the second one records how the removed SE is selected, and the third one describes how to preserve the information of the removed SE. It is worth to note that both MFOGD and MNOGD do not remove any SE and transform every SE into kernel-induced feature vectors.

FIGURE 6 depicts the cumulative mistake rate as a function of the budget B for budgeted online algorithms as well as non-budgeted methods. From the curves, we can draw

TABLE 4. Complexity comparison for budgeted and non-budgeted online multiclass algorithms.

Algorithm	Time Complexity	Space Complexity
MPerceptron	$\mathcal{O}(\|\mathcal{I}_t\|)$	$\mathcal{O}(\|\mathcal{I}_t\|)$
MPA	$\mathcal{O}(\|\mathcal{I}_t\|)$	$\mathcal{O}(\|\mathcal{I}_t\|)$
MRBP	$\mathcal{O}(B)$	$\mathcal{O}(B)$
MProjectron++	$\mathcal{O}(B^2)$	$\mathcal{O}(B^2)$
MFOGD	$\mathcal{O}(B)$	$\mathcal{O}(B)$
MNOGD	$\mathcal{O}(B^2)$	$\mathcal{O}(B^2)$
BMPA-R	$\mathcal{O}(B^2)$	$\mathcal{O}(B^2)$
BMPA-O	$\mathcal{O}(B^2)$	$\mathcal{O}(B^2)$
BMPA-P	$\mathcal{O}(B^3)$	$\mathcal{O}(B^2)$
BMPA-S	$\mathcal{O}(B^2)$	$\mathcal{O}(B^2)$

some observations as follows. First of all, MPA outperforms MPerceptron since the mistake rate is lower for MPA than for Mperceptron. Secondly, the mistake rate of all budgeted algorithms drops as the budget increases. BMPA converges to MPA while there is a performance gap between other budgeted algorithms and MPA. Thirdly, BMPA generally outperforms other budgeted algorithms since it achieves the lowest mistake rate for each dataset and each budget. Lastly, the comparison of MRBP, BMPA-R, and BMPA-S suggests that both the removal strategy and the non-budgeted algorithm itself are important to have a good prediction performance. All in all, we conclude that BMPA is competitive with state-of-the-art budgeted online algorithms.

Finally, as a summary, TABLE 4 compares the time and space complexities per update for budgeted and non-budgeted online multiclass algorithms evaluated in this section. Because MPerceptron and MPA are non-budgeted algorithms, their time and space complexities depend on the number of SEs, $|\mathcal{I}_t|$, and grow gradually. On the other hand, budgeted algorithms, including the proposed BMPA algorithm, have constant time and space complexities once the budget B is given. Although none of the removal strategies associated with the proposed BMPA algorithm shows benefits based on the complexity comparison, BMPA-S achieves the best prediction performance in general according to the results demonstrated in FIGURE 6.

VII. CONCLUSION

Based on the kernel-based MPA algorithm, we propose the BMPA algorithm that controls the growth of a kernel-based model by limiting the maximum number of available support elements via removal and fully exploits them through a constrained optimization. BMPA is derived from the resource perspective, which provides an alternative and equivalent interpretation of the kernel-based MPA algorithm. BMPA can be treated as an approximation to MPA justified by the resource perspective. Moreover, it breaks the curse of kernelization that makes kernel-based models fail in resource-insufficient occasions. We study three removal strategies along with a unified theoretical analysis for prediction mistakes made by BMPA. Among them, BMPA-S achieves the best trade-off. We conduct simulation experiments on open datasets and show that BMPA is effective and competitive. In future works, we plan to study the combination of the resource perspective and the budget idea with more online algorithms, particularly those based on optimization. Besides, we will study the integration of the proposed BMPA algorithm with other kernels and the extension of the proposed BMPA algorithm to scenarios with dynamic computational power.

Note that the proposed BMPA algorithm is designed to deal with online problems where the sequences of examples are generated without any statistical assumption. There is another research thread dealing with the sequences of examples generated from a fixed source (or a source model with specific parameters), and this is the primary focus of batch learning. Although the proposed BMPA algorithm may be used for this kind of problem, it is not the main focus of this paper. We leave it as one of the future works. On the other hand, it has come to our attention that some of the most representative computational intelligence algorithms are bio-inspired, e.g., monarch butterfly optimization [56], elephant herding optimization [57], the earthworm optimisation algorithm [58], and the moth search algorithm [59]. These types of algorithms may be used to design powerful online algorithms and are worth for further study.

ACKNOWLEDGMENT

Parts of the work were presented at the 2017 IEEE International Symposium on Circuits and Systems, May 2017.

REFERENCES

[1] P. Zhao, S. C. H. Hoi, and R. Jin, "Double updating online learning," *J. Mach. Learn. Res.*, vol. 12, pp. 1587–1615, May 2011.

[2] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.

[3] S. C. H. Hoi, R. Jin, P. Zhao, and T. Yang, "Online multiple kernel classification," *Mach. Learn.*, vol. 90, no. 2, pp. 289–316, 2013.

[4] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. ICML*, Montreal, QC, Canada, Jun. 2009, pp. 681–688.

[5] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica, "Ad click prediction: A view from the trenches," in *Proc. KDD*, Chicago, IL, USA, Aug. 2013, pp. 1222–1230.

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

[7] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. ICML*, Washington, DC, USA, Aug. 2003, pp. 928–936.

[8] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.

[9] C. Gentile, "A new approximate maximal margin classification algorithm," *J. Mach. Learn. Res.*, vol. 2, pp. 213–242, Dec. 2001.

[10] 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.

[11] 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.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[13] F. Orabona, J. Keshet, and B. Caputo, "Bounded kernel-based online learning," *J. Mach. Learn. Res.*, vol. 10, pp. 2643–2666, Dec. 2009.

[14] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[15] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, Oct. 2012.
[16] K. Crammer, J. Kandola, and Y. Singer, "Online classification on a budget," in *Proc. NIPS*, Vancouver, BC, Canada, Dec. 2003, pp. 225–232.
[17] J. Weston, A. Bordes, and L. Bottou, "Online (and offline) on an even tighter budget," in *Proc. AISTATS*, Bridgetown, Barbados, Jan. 2005, pp. 413–420.
[18] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based Perceptron on a budget," *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, 2008.
[19] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Mach. Learn.*, vol. 69, no. 2, pp. 143–167, 2007.
[20] Z. Wang and S. Vucetic, "Online passive-aggressive algorithms on a budget," in *Proc. AISTATS*. Sardinia, Italy: Chia Laguna Resort, Mar. 2010, pp. 908–915.
[21] J. Wang, S. C. H. Hoi, P. Zhao, J. Zhuang, and Z. Liu, "Large scale online kernel classification," in *Proc. IJCAI*, Beijing, China, Aug. 2013, pp. 1750–1756.
[22] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *J. Mach. Learn. Res.*, vol. 17, no. 47, pp. 1–43, 2016.
[23] Z. Wang, K. Crammer, and S. Vucetic, "Multi-class pegasos on a budget," in *Proc. ICML*, Haifa, Israel, Jun. 2010, pp. 1143–1150.
[24] B. Li, P. Zhao, S. C. H. Hoi, and V. Gopalkrishnan, "PAMR: Passive aggressive mean reversion strategy for portfolio selection," *Mach. Learn.*, vol. 87, no. 2, pp. 221–258, 2012.
[25] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *Proc. ICML*, Helsinki, Finland, Jul. 2008, pp. 264–271.
[26] M. Blondel, Y. Kubo, and N. Ueda, "Online passive-aggressive algorithms for non-negative matrix factorization and completion," in *Proc. AISTATS*, Reykjavik, Iceland, Apr. 2014, pp. 96–104.
[27] J. Lu, P. Zhao, and S. C. H. Hoi, "Online passive aggressive active learning and its applications," in *Proc. ACML*, Nha Trang City, Vietnam, Nov. 2014, pp. 266–282.
[28] K. Crammer, M. Dredze, and F. Pereira, "Exact convex confidence-weighted learning," in *Proc. NIPS*, Vancouver, BC, Canada, Dec. 2008, pp. 345–352.
[29] J. Wang, P. Zhao, and S. C. H. Hoi, "Exact soft confidence-weighted learning," in *Proc. ICML*, Edinburgh, U.K., 2012, pp. 121–128.
[30] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," in *Proc. NIPS*, Vancouver, BC, Canada, Dec. 2009, pp. 414–422.
[31] K. Crammer, M. Dredze, and A. Kulesza, "Multi-class confidence weighted algorithms," in *Proc. EMNLP*, Singapore, Aug. 2009, pp. 496–504.
[32] S. Matsushima, N. Shimizu, K. Yoshida, T. Ninomiya, and H. Nakagawa, "Exact passive-aggressive algorithm for multiclass classification using support class," in *Proc. SDM*, Columbus, OH, USA, 2010, pp. 303–314.
[33] J. Lu, P. Zhao, and S. C. H. Hoi, "Online sparse passive aggressive learning with kernels," in *Proc. SDM*, Miami, FL, USA, May 2016, pp. 675–683.
[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1106–1114.
[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, Boston, MA, USA, Jun. 2015, pp. 1–9.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[37] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
[38] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[39] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
[40] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Apr. 2014.
[41] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, May 2010.
[42] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Mach. Learn.*, vol. 106, nos. 9–10, pp. 1469–1495, Oct. 2017.
[43] A. Cano and B. Krawczyk, "Kappa updated ensemble for drifting data stream mining," *Mach. Learn.*, vol. 109, no. 1, pp. 175–218, Jan. 2020.
[44] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Proc. ECML PKDD*, 2010, pp. 135–150.
[45] S. You and H. Lin, "A simple unlearning framework for online learning under concept drifts," in *Proc. PAKDD*, 2016, pp. 115–126.
[46] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," Oct. 2018, *arXiv:1802.02871*. [Online]. Available: http://arxiv.org/abs/1802.02871
[47] A. B. J. Novikoff, "On convergence proofs on perceptrons," in *Proc. Symp. Math. Theory Automata*, New York, NY, USA, vol. 12, 1962, pp. 615–622.
[48] C. Gentile, "The robustness of the *p*-norm algorithms," *Mach. Learn.*, vol. 53, no. 3, pp. 265–299, 2003.
[49] K. Crammer, M. Dredze, and F. Pereira, "Confidence-weighted linear classification for text categorization," *J. Mach. Learn. Res.*, vol. 13, pp. 1891–1926, Jun. 2012.
[50] J. Wang, P. Zhao, and S. C. H. Hoi, "Soft confidence-weighted learning," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 1, pp. 15:1–15:32, 2016.
[51] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," *Mach. Learn.*, vol. 91, no. 2, pp. 155–187, May 2013.
[52] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *J. Mach. Learn. Res.*, vol. 3, pp. 951–991, Mar. 2003.
[53] Z. Wang and S. Vucetic, "Twin vector machines for online learning on a budget," in *Proc. SDM*, Sparks, NV, USA, 2009, pp. 906–917.
[54] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.
[55] C.-H. Wu, W.-C. His, H. H.-S. Lu, and H.-M. Hang, "Online multiclass passive-aggressive learning on a fixed budget," in *Proc. ISCAS*, Baltimore, MD, USA, May 2017, pp. 1–4.
[56] G. G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 1995–2014, 2019.
[57] G.-G. Wang, S. Deb, and L. S. dos Coelho, "Elephant herding optimization," in *Proc. ISCBI*, Bali, Indonesia, Dec. 2015, pp. 1–5.
[58] G.-G. Wang, S. Deb, and L. dos S. Coelho, "Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems," *Int. J. Bio-Inspired Comput.*, vol. 12, no. 1, pp. 1–22, Jan. 2018.
[59] G.-G. Wang, "Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Comput.*, vol. 10, pp. 151–164, Jun. 2018.
[60] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.

CHUNG-HAO WU was born in Taichung, Taiwan, in 1987. He received the B.S. degree from the Electrical Engineering and Computer Science (EECS) Undergraduate Honors Program, National Chiao Tung University, Taiwan, in 2009, where he is currently pursuing the Ph.D. degree with the Institute of Electronics. His research interests include machine learning and signal processing.

HENRY HORNG-SHING LU received the B.S. degree in electrical engineering from National Taiwan University, in 1986, and the Ph.D. degree in statistics from Cornell University, in 1994. He is currently a Professor with the Institute of Statistics, National Chiao Tung University, Taiwan. He serves as the Vice President for Academic Affairs. His findings have been published in a wide spectrum of journals, conference papers, and book chapters. He also co-edited the *Handbook of Statistical Bioinformatics* and *Big Data Analytics*, published by (Springer, in 2011 and 2018). His research interests include statistics, image science, bioinformatics, and big data analytics. He analyzes different types of data by developing statistical methodologies for machine learning with the power of statistical inference and computation algorithms. He is an Elected Member of the International Statistical Institute (ISI).

HSUEH-MING HANG (Fellow, IEEE) received the B.S. and M.S. degrees from National Chiao Tung University, Hsinchu, Taiwan, in 1978 and 1980, respectively, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1984.

From 1984 to 1991, he was with AT&T Bell Laboratories, Holmdel, NJ, USA, and then he joined the Department of Electronics Engineering, National Chiao Tung University (NCTU), in December 1991. From 2006 to 2009, he was appointed as the Dean of the College of Electrical Engineering and Computer Science (EECS), National Taipei University of Technology (NTUT). From 2014 to 2017, he was the Dean of the College of Electrical and Computer Engineering, NCTU. He has been actively involved in the international MPEG standards, since 1984. He holds 14 patents (Taiwan, USA, and Japan). He has published over 200 technical articles related to image compression, signal processing, and video codec architecture. His current research interests include multimedia compression, spherical image/video processing, and deep-learning based image/video processing. He is a Fellow of IET and a member of Sigma Xi. He was a Board Member of the Asia–Pacific Signal and Information Processing Association (APSIPA), from 2013 to 2018. He was an IEEE Circuits and Systems Society Distinguished Lecturer, from 2014 to 2015. He was a recipient of the IEEE Third Millennium Medal. He was an Associate Editor (AE) of the IEEE TRANSACTIONS ON IMAGE PROCESSING, from 1992 to 1994 and from 2008 to 2012, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, from 1997 to 1999. He was a General Co-Chair of the 2019 IEEE International Conference on Image Processing (ICIP).