Online Sparse Passive Aggressive Learning with Kernels

Jing Lu* Peilin Zhao† Steven C.H. Hoi‡

Abstract

Conventional online kernel methods often yield an unbounded large number of support vectors, making them inefficient and non-scalable for large-scale applications. Recent studies on bounded kernel-based online learning have attempted to overcome this shortcoming. Although they can bound the number of support vectors at each iteration, most of them fail to bound the number of support vectors for the final output solution which is often obtained by averaging the series of solutions over all the iterations. In this paper, we propose a novel kernel-based online learning method, Sparse Passive Aggressive learning (SPA), which can output a final solution with a bounded number of support vectors. The key idea of our method is to explore an efficient stochastic sampling strategy, which turns an example into a new support vector with some probability that depends on the loss suffered by the example. We theoretically prove that the proposed SPA algorithm achieves an optimal regret bound in expectation, and empirically show that the new algorithm outperforms various bounded kernel-based online learning algorithms.

1 Introduction

Online learning with kernels represents an important family of machine learning algorithms for learning nonlinear predictive models in large-scale machine learning tasks [10, 8, 13]. Due to the curse of kernelization, a major limitation of many kernel-based online learning techniques is that the number of support vectors is unbounded and potentially large for large-scale applications. This has raised a huge challenge for applying them in practical applications since computational complexity (of both time and space) for a kernel-based online learning algorithm is often proportional to the support vector size.

Recent years have witnessed a variety of emerging studies for bounded kernel-based online learning. Examples include Budget Perceptron [4], Randomized Budget Perceptron (RBP) [1], Forgetron [6], Projectron [12], Budget Passive Aggressive learning [19], Bounded Online Gradient Descent (BOGD) [22, 17], Twin Support Vector Machine (TVM) [18], among others.

Although bounded kernel-based online learning has been actively studied, most existing algorithms suffer from a key drawback when applying them in online-to-batch conversion, a process that aims to convert online classifiers for batch classification purposes [7, 5]. Specifically, one of most commonly used approaches in online-to-batch conversion is to take the *averaging classifier*, that is the mean of all the online classifiers at every online iteration, as the final classifier for batch classification. This simple technique is not only computationally efficient, but also enjoys theoretical superiority in generalization performance compared with the classifier obtained in the last online iteration [15]. Unfortunately, most existing budget online kernel learning algorithms only guarantee the support vector size at each online iteration is bounded, but fail to yield a sparse averaging classifier in online-to-batch conversion.

Our work is closely related to two sparse kernel methods for online logistic regression [20, 21]. The main limitation of the existing work is that they only considered the problem settings with several smooth loss functions. This is a relatively strict setting since there are lots of situations (eg. SVM) where unsmooth loss functions , hinge loss for example, are adopted. Our paper studied the unsmooth loss setting where the loss function is not limited by assumptions in [21]. The second improvement of our paper compared to the previous work is the better theoretical analysis, which follows the standard analysis of online learning and thus is simpler and easier to follow. Our analysis also suggests an approach to fix a subtle mistake in the proof of bound in [20].

In this paper, we present a new method for bounded kernel-based online learning method, named "Sparse Passive Aggressive" (SPA) learning, which extends the online Passive Aggressive (PA) learning method [3] to ensure the final output averaging classifier has the bounded number of support vectors in online-to-batch conversion. Specifically, the basic idea of our method is to explore a simple stochastic sampling rule, which assigns an incoming training example to be a support vector with a higher probability when it suffers a higher loss. We theoretically prove that the proposed

*School of Information Systems, Singapore Management University, Singapore 178902. jing.lu.2014@phdis.smu.edu.sg

†Institute for Infocomm Research (I2R), A*STAR, Singapore. zhaop@i2r.a-star.edu.sg

‡School of Information Systems, Singapore Management University, Singapore 178902. chhoi@smu.edu.sg

algorithm not only bounds the number of support vectors but also achieves an optimal regret bound in expectation. Finally, we conduct an extensive set of empirical studies which show that the proposed algorithm outperforms a variety of bounded kernel-based online learning algorithms.

The rest of this paper is organized as follows. Section 2 formally formulates the problem and then presents the proposed SPA algorithm. Section 3 gives theoretical analysis. Section 4 presents our experimental studies and empirical observations, and finally Section 5 concludes this paper.

2 Sparse PA Learning with Kernels

In this section, we first formulate the problem setting for online learning with kernels, then review online Passive Aggressive (PA) algorithm [3], and finally present the details of the proposed Sparse PA learning with kernels (SPA) algorithm.

2.1 Problem Setting and Preliminaries We consider the problem of online learning by following online convex optimization settings. Our goal is to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ from a sequence of training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$, where instance $\mathbf{x}_t \in \mathbb{R}^d$ and class label $y_t \in \mathcal{Y}$. We refer to the output f of the learning algorithm as a *hypothesis* and denote the set of all possible hypotheses by $\mathcal{H} = \{f | f : \mathbb{R}^d \to \mathbb{R}\}$. We will use $\ell(f; (\mathbf{x}, y)) : \mathcal{H} \times (\mathbb{R}^d \times \mathcal{Y}) \to \mathbb{R}$ as the loss function that penalizes the deviation of estimating $f(\mathbf{x})$ from observed labels y. Further, we consider $\mathcal{H}$ a Reproducing Kernel Hilbert Space (**RKHS**) endowed with a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [16] implementing the inner product$\langle \cdot, \cdot \rangle$ such that: 1) κ has the reproducing property $\langle f, \kappa(\mathbf{x}, \cdot) \rangle = f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^d$; 2) $\mathcal{H}$ is the closure of the span of all $\kappa(\mathbf{x}, \cdot)$ with $\mathbf{x} \in \mathbb{R}^d$, that is, $\kappa(\mathbf{x}, \cdot) \in \mathcal{H}\ \forall \mathbf{x} \in \mathcal{X}$. The inner product $\langle \cdot, \cdot \rangle$ induces a norm on $f \in \mathcal{H}$ in the usual way: $\|f\|_{\mathcal{H}} := \langle f, f \rangle^{\frac{1}{2}}$. To make it clear, we denote by $\mathcal{H}_\kappa$ an RKHS with explicit dependence on kernel κ. Throughout the analysis, we assume $\kappa(\mathbf{x}, \mathbf{x}) \leq X^2\ \forall \mathbf{x} \in \mathbb{R}^d$.

Passive Aggressive (PA) algorithms [3] are a family of margin based online learning algorithms, which can achieve a bound on the cumulative loss comparable with the smallest loss that can be attained by any fixed hypothesis.

Specifically, consider a classification problem, an online PA algorithm sequentially updates the online classifier. At the t-step, the online hypothesis will be updated by the following strategy

$$f_{t+1} = \min_{f \in \mathcal{H}_\kappa} \frac{1}{2} \|f - f_t\|^2_{\mathcal{H}_\kappa} + \eta \ell(f; (\mathbf{x}_t, y_t))$$

where $\eta > 0$. This optimization involves two objectives: the first is to keep the new function close to the old one, while the second is to minimize the loss of the new function on the current example. To simplify the discussion, we denote $\ell_t(f) = \ell(f; (\mathbf{x}_t, y_t))$ throughout the paper.

2.2 Sparse Passive Aggressive Algorithm Similar to conventional kernel-based online learning methods, the critical limitation of PA is that it does not bound the number of support vectors, making it very expensive in both computational time and memory cost for large-scale applications. Some existing work has attempted to propose budget PA [19] for learning a bounded kernel classifier at each step using some budget maintenance strategy (e.g., discarding an old SV and replacing it by a new one). Although the kernel classifier is bounded at each step, their approach cannot bound the number of support vectors for the average of classifiers over all learning iterations. This drawback of the existing budgeted kernel algorithms limits their application to online-to-batch conversion tasks where the output final classifier is typically obtained by averaging classifiers at every learning steps. Furthermore, even in pure online setting, the prediction of the new coming instance $\mathbf{x}_t$ using $\frac{1}{t}\sum_{i=1}^{t} f_i$ often outperforms the result using a single classifier f_t. In this paper, we aim to overcome the above limitation by proposing a novel Sparse Passive Aggressive (SPA) learning algorithm for learning a sparse kernel classifier which guarantees not only the ratio of support vectors to total received examples is always bounded in expectation, but also the support vector size of the final average classifier is bounded.

Unlike the conventional budget maintenance idea, we propose a stochastic support vector sampling strategy which sequentially constructs the set of support vectors by sampling from the sequence of instances in which an instance whenever is added into the support vector set will never be discarded. This ensures that the support vector set of any intermediate classifier is always a subset of the final average classifier. The rest challenge then is how to design an appropriate sampling strategy so that we ensure that the support vector size of kernel classifiers is always bounded while maximizing the learning accuracy of the classifier. To tackle this challenge, we propose a simple yet effective sampling rule which decides if an incoming instance should be a support vector by performing a Bernoulli trial as follows:

$$\Pr(Z_t = 1) = \rho_t, \quad \rho_t = \frac{\min(\alpha, \ell_t(f_t))}{\beta}$$

where $Z_t \in \{0, 1\}$ is a random variable such that $Z_t = 1$ indicates a new support vector should be added to

update the classifier at the t-th step, and $\beta \geq \alpha > 0$ are parameters to adjust the ratio of support vectors with some given budget. The above sampling rule has two key concerns:

(i) The probability of making the t-th step update is always less than α/β, which avoids assigning too high probability on a noisy instance.

(ii) An example suffering higher loss has a higher probability of being assigned to the support vector set.

In the above, the first is to guarantee that the ratio of support vectors to total received instances is always bounded in expectation, and the second is to maximize the learning accuracy by adding informative support vectors or equivalently avoid making unnecessary updates. For example, for the extreme case of $\ell_t(f_t) = 0$, we always have $Pr(Z_t = 1) = 0$, which means we never makes an update if an instance does not suffer loss. Different from the existing work where the sampling probability is related to the derivative of classification loss, our probability is directly based on the scale of hinge loss.

After obtaining the random variable Z_t, we will need to develop an effective strategy for updating the classifier. Following the PA learning principle, we propose the following updating method:

$$f_{t+1} = \min_{f \in \mathcal{H}_\kappa} P_t(f) := \frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} + \frac{Z_t}{\rho_t}\eta\ell_t(f) \tag{2.1}$$

Note when $\rho_t = 0$, then $Z_t = 0$ and we set $Z_t/\rho_t = 0$. We adopt the above update because its objective is an unbiased estimation of that of PA update, that is,

$$\mathbb{E}(P_t(f)) = \frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} + \eta\ell_t(f)$$

Passive Aggressive based algorithms enjoy lots of advantages compared to gradient based ones. For instance, PA updating strategy is more effective because of the adaptive step size. And PA is less sensitive to the variance of parameter setting. Finally, we summarize the proposed Sparse Passive Aggressive algorithm in Algorithm 1.

2.3 Application to Binary Classification The proposed SPA method is a generic online learning framework that can be applied to various online learning tasks. Examples include online classification, regression, and uniclass prediction tasks. Without loss of generality, we focus on the discussion on the application of the proposed algorithm for online binary classification task.

Algorithm 1 Sparse PA learning with kernels (**SPA**)

Input: aggressiveness parameter $\eta > 0$, and parameters $\beta \geq \alpha > 0$
Initialize: $f_1(\mathbf{x}) = 0$
for $t = 1, 2, \ldots, T$ **do**
  Receive example: $(\mathbf{x}_t, y_t)$
  Suffer loss: $\ell_t(f_t) = \ell(f_t; (\mathbf{x}_t, y_t))$
  Compute $\rho_t = \frac{\min(\alpha, \ell_t(f_t))}{\beta}$
  Sample a Bernoulli random variable $Z_t \in \{0, 1\}$ by: $\Pr(Z_t = 1) = \rho_t$
  Update the classifier:
  $f_{t+1} = \min_{f \in \mathcal{H}_\kappa} \frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} + \frac{Z_t}{\rho_t}\eta\ell_t(f)$
end for
Output: $\bar{f}_T(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^T f_t(\mathbf{x})$

Specifically, we consider an online classification task with label set $\mathcal{Y} = \{-1, +1\}$, and adopt the widely used hinge loss function:

$$\ell(f; (\mathbf{x}, y)) = [1 - yf(\mathbf{x})]_+$$

where $[z]_+ = \max(0, z)$. The hinge loss function is $\kappa(\mathbf{x}, \mathbf{x})$-Lipschitz with respect to f:

$$\begin{aligned}
&|\ell(f; (\mathbf{x}, y)) - \ell(g; (\mathbf{x}, y))| \\
&= |[1 - yf(\mathbf{x})]_+ - [1 - yg(\mathbf{x})]_+| \\
&\leq |yf(\mathbf{x}) - yg(\mathbf{x})| \leq \sqrt{\kappa(\mathbf{x}, \mathbf{x})}\|f - g\|_{\mathcal{H}_\kappa},
\end{aligned}$$

where we used the fact $[1 - z]_+$ is 1-Lipschitz with respect to z. This further implies the corresponding $\ell_t(f)$s are X-Lipschitz, since $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq X^2$. This fact will be used in the theoretical analysis in the next section.

After choosing the hinge loss, the rest is how to solve the optimization (2.1). Fortunately, we can derive a closed-form solution, as shown in the following proposition.

PROPOSITION 1. *When $\ell_t(f) = [1 - y_t f(\mathbf{x}_t)]_+$, the optimization* (2.1) *enjoys the following closed-form solution*

$$f_{t+1}(\cdot) = f_t(\cdot) + \tau_t y_t \kappa(\mathbf{x}_t, \cdot), \quad \tau_t = \min(\frac{\eta Z_t}{\rho_t}, \frac{\ell_t(f_t)}{\kappa(\mathbf{x}_t, \mathbf{x}_t)})$$

This proposition is easy to prove, so we omit its proof.

3 Theoretical Analysis

SPA algorithm will be analyzed for binary classification case. To facilitate the discussion, we denote

$$f_* = \arg\min_f \sum_{t=1}^T \ell_t(f)$$

The following theorem analyzes the regret of SPA, i.e., $\sum_{t=1}^{T}\ell_t(f_t)-\sum_{t=1}^{T}\ell_t(f_*)$, which is a main performance measure of online algorithms.

THEOREM 3.1. *Let $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ be a sequence of examples where $\mathbf{x}_t\in\mathbb{R}^d$, $y_t\in\mathcal{Y}=\{-1,+1\}$ for all t. If we assume $\kappa(\mathbf{x},\mathbf{x})=1$ and the hinge loss function $\ell_t(\cdot)$ is 1-Lipschitz, then for any $\beta\geq\alpha>0$, and $\eta>0$, the proposed SPA algorithm satisfies the following inequality*

$$\mathbb{E}[\sum_{t=1}^{T}(\ell_t(f_t)-\ell_t(f_*))]<\frac{1}{2\eta}\|f_*\|_{\mathcal{H}_\kappa}^2+\frac{\eta\beta}{\min(\alpha,\sqrt{\beta\eta})}T$$

where η is the aggressiveness parameter. When setting $\eta=\|f_*\|_{\mathcal{H}_\kappa}\sqrt{\frac{\alpha}{2\beta T}}$ and $\alpha^3\leq\frac{\beta}{2T}\|f_*\|_{\mathcal{H}_\kappa}^2$, we will have

$$\mathbb{E}[\sum_{t=1}^{T}(\ell_t(f_t)-\ell_t(f_*))]<\|f_*\|_{\mathcal{H}_\kappa}\sqrt{2\beta T/\alpha}$$

The detailed proof can be found in Appendix A.

Remark 1. The theorem indicates that the expected regret of the proposed SPA algorithm can be bounded by $\|f_*\|_{\mathcal{H}_\kappa}\sqrt{2\beta T/\alpha}$ in expectation. In practice, β/α is usually a small constance. Thus, we can conclude that the proposed algorithm can achieve a strong regret bound in expectation.

Remark 2. The expected regret has a close relationship with the two parameters α and β. As indicated above, to avoid assigning too high probability on a noisy instance, the parameter α can not be too large. Assuming $\alpha\leq\sqrt{\beta\eta}$ (which is accessible in the practical parameter setting), the expected regret bound is proportion to the ratio β/α. This consists with the intuition that larger chances of adding SVs leads to smaller loss. Further more, for $\alpha>\sqrt{\beta\eta}$, the expected regret bound is less tight than the above case, which consists with the analysis before that too large α involves large number of noisy instances and might be harmful.

Next, we would give a bound on the number of support vectors of the final classifier $\bar{f}_T$ in expectation. Since we never delete an existing support vector, it should be the same with the number of support vectors of the final intermediate classifier f_T, which equals to the random number $\sum_{t=1}^{T}Z_t$.

THEOREM 3.2. *Let $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ be a sequence of examples where $\mathbf{x}_t\in\mathbb{R}^d$, $y_t\in\mathcal{Y}=\{-1,+1\}$ for all t. If we assume $\kappa(\mathbf{x},\mathbf{x})=1$ and the hinge loss function $\ell_t(\cdot)$ is 1-Lipschitz, then for any $\beta\geq\alpha>0$, and $\eta>0$, the proposed SPA algorithm satisfies the following inequality*

$$\mathbb{E}[\sum_{t=1}^{T}Z_t]\leq$$
$$\min\left\{\frac{\alpha}{\beta}T,\frac{1}{\beta}[\sum_{t=1}^{T}\ell_t(f_*)+\frac{1}{2\eta}\|f_*\|_{\mathcal{H}_\kappa}^2+\frac{\eta\beta}{\min(\alpha,\sqrt{\beta\eta})}T]\right\}$$

Especially, when $\eta=\|f_*\|_{\mathcal{H}_\kappa}\sqrt{\frac{\alpha}{2\beta T}}$ and $\alpha^3\leq\frac{\beta}{2T}\|f_*\|_{\mathcal{H}_\kappa}^2$, we have

$$\mathbb{E}[\sum_{t=1}^{T}Z_t]\leq\min\left\{\frac{\alpha}{\beta}T,\frac{1}{\beta}[\sum_{t=1}^{T}\ell_t(f_*)+\|f_*\|_{\mathcal{H}_\kappa}\sqrt{2\beta T/\alpha}]\right\}$$

The detailed proof can be found in Appendix B.

Remark. First, this theorem indicates the expected number of support vectors is less than $\alpha T/\beta$. Thus, by setting $\beta\geq\alpha T/n$ $(1<n\leq T)$, we guarantee the expected number of support vectors of the final classifier is bounded by a budget n. Second, this theorem also implies that, by setting $\beta\geq[\sum_{t=1}^{T}\ell_t(f_*)+\|f_*\|_{\mathcal{H}_\kappa}\sqrt{2\beta T/\alpha}]/n$ $(1<n\leq T)$, the expected number of support vectors is always less than n, no matter how is the value of α.

4 Experiments

In this section, we conduct extensive experiments to evaluate the empirical performance of the proposed SPA algorithm for online binary classification tasks.

4.1 Experimental Testbed Table 1 summarizes details of some binary classification datasets in our experiments. The first five can be downloaded from LIBSVM[1]or KDDCUP competition site [2]. The last "*Gaussian*" is a synthetic dataset for large-scale evaluation, which was generated based on a mixture of two 2-dimensional Gaussians: $\mathcal{N}((0,0),I)$ (40%) and $\mathcal{N}((2,0),4I)$ (60%).

Table 1: Summary of binary classification datasets.

DATA SET	# INSTANCES	#FEATURES
KDD08	102,294	117
A9A	48,842	123
W7A	49,749	300
CODRNA	59,535	8
COVTYPE	581,012	54
GAUSSIAN	1,000,000	2

4.2 Compared Algorithms and Setup We evaluate the proposed SPA algorithm by comparing with many state-of-the-art online kernel learning algorithms. First, we implement the following non-budget kernel learning algorithms as a yardstick for evaluation:

- "Perceptron": the kernelized Perceptron [9];
- "OGD": the kernelized OGD algorithm [11];
- "PA-I": the kernelized passive aggressive algorithm with soft margin [3].

[1]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[2]http://www.sigkdd.org/kddcup/

Further, we compare our SPA algorithm with a variety of budget online kernel learning algorithms:

- "RBP": the Randomized Budget Perceptron by random removal [1];
- "Forgetron": the Forgetron by discarding oldest support vectors [6];
- "Projectron": the Projectron algorithm using the projection strategy [12];
- "Projectron++": the aggressive version of Projectron algorithm [12];
- "BPA-S": the Budget Passive-Aggressive algorithm [19], we only adopt the BPA-Simple algorithm since the other two variants are too computational expensive to scale to large datasets;
- "BOGD": the Bounded Online Gradient Descent algorithm [22, 17];
- "OSKL":the Online Sparse Kernel Learning algorithm [21].

To make a fair comparison, we adopt the same experimental setup for all the algorithms. We use the hinge loss for gradient-based algorithms (OGD and BOGD). For all the algorithms, we adopt a gaussian kernel $exp(-\gamma||x_1 - x_2||^2)$ with parameter γ set to 0.4 for all the datasets. The learning rate parameter η for OGD, BOGD, OSKL and SPA is automatically chosen by searching from $\{10^3, ..., 10^{-3}\}$ based on a random permutation of each dataset. We adopt the PA-I algorithm for comparison since it was proved to be robust to noise and achieved better performance than PA without soft margin. The soft margin parameter C is optimized by searching from range {0.25, 0.5, 1, 2}. To decide the support vector size of the proposed SPA algorithm, it requires to choose parameters α and β. The parameter α is chosen according to the distribution of the loss function scale, for which we set $\alpha = 0.5$ for *Gaussian* and $\alpha = 1$ for all other datasets. We choose a proper β parameter so that the resulting support vector size is roughly a proper fraction of that of the non-budget OGD algorithm (specifically, we set $\beta = 20$ for three small datasets, and $\beta = 200$ for three large datasets). Note that the OSKL algorithm also adopts a stochastic approach to sample the support vectors, which indicates that there is no ideal method for setting the same number of support vectors and thus absolutely fair comparison between OSKL and our proposed algorithm. We tune the sampling probability parameter G in the OSKL algorithm so that the averaged number of support vectors is close to that of the proposed algorithm. Finally, to ensure that all budget algorithms adopt the same budget size, we choose the budget size B of all the other compared algorithms according to the average number of support vectors generated by the proposed SPA algorithm.

For each dataset, all the experiments were repeated 20 times on different random permutations of instances in the dataset and all the results were obtained by averaging over these 20 runs. For performance metrics, we evaluate the online classification performance by mistake rates and running time. Finally, all the algorithms were implemented in C++, and all experiments were conducted in a Windows machine with 3.2 GHz CPU.

4.3 Evaluation of Online Learning Performance

The first experiment is to evaluate the performance of kernel-based online learning algorithms for online classification tasks. Table 2 shows the experimental results. To demonstrate the superiority of averaging classifier in pure online setting, the reported accuracy of the proposed SPA algorithm and OSKL is obtained by the averaged classifier $\frac{1}{t}\sum_{i=1}^{t} f_i$. We can draw some observations as follows.

We first examine the time cost comparisons. First of all, all the budget algorithms are significantly more efficient than the non-budget algorithms (Perceptron, OGD, and PA-I) for most cases, especially on large-scale datasets. This validates the importance of studying bounded kernel-based online learning algorithms. Second, by comparing different budget algorithms, Projectron++ is the least efficient due to its expensive projection strategy, and the proposed SPA algorithm is the most efficient. The other budget algorithms (RBP, Forgetron, BPAS, BOGD) are in general quite efficient as they all are based on simple SV removal strategy for budget maintenance. The reason that our SPA algorithm is even more efficient than these algorithms is because our algorithm adds the support vectors incrementally while the other algorithms perform the budget maintenance only when the support vector size reaches the budget. This encouraging result validates the high efficiency advantage of our stochastic support vector sampling strategy.

We then compare online classification accuracy of different algorithms. First, the non-budget algorithms have better accuracy than their budget variants. This is not surprising as the non-budget algorithms use a larger SV size. Second, comparing different budget algorithms, we found that PA and OGD based algorithms (BPAS, BOGD, and SPA) generally outperform Perceptron-based algorithms (RBP, Forgetron, Projectron, and Projectron++). Finally, our SPA algorithm achieves the best accuracy among all the budget algorithms most of the cases and is comparable to the OSKL algorithm. These results again validate the effectiveness of the proposed budget learning strategy.

Table 2: Evaluation of Online Kernel Classification on Six Datasets. Time in seconds

	KDD08, $\beta = 20$			a9a, $\beta = 20$			codrna, $\beta = 20$		
Algorithm	Accuracy (%)	Time	#SVs	Accuracy (%)	Time	#SVs	Accuracy (%)	Time	#SVs
Perceptron	99.04±0.01	11.58	0.9k	79.40± 0.11	19.93	9.9k	90.79 ±0.09	6.21	5.4k
OGD	99.44±0.01	14.54	1.2k	83.41 ± 0.05	54.20	23.5k	93.31±0.05	10.41	8.8k
PA-I	99.45±0.01	39.53	3.2k	84.07±0.08	57.91	22.8k	93.65±0.05	15.82	12.4k
RBP	98.96±0.03	3.24	149	78.83±0.38	5.34	1,350	86.59±0.22	1.68	822
Forgetron	98.93±0.03	3.28	149	78.08±0.22	6.45	1,350	86.62±0.15	1.91	822
Projectron	99.02±0.01	4.19	149	79.25±0.13	32.47	1,350	83.35±0.23	19.38	822
Projcetron++	99.32±0.01	4.86	149	79.35±0.13	150.63	1,350	84.71±0.19	32.99	822
BPAS	**99.41±0.01**	3.93	149	80.44±0.14	7.75	1,350	91.23±0.05	2.21	822
BOGD	99.39±0.01	3.40	149	80.97±0.04	6.17	1,350	85.62±0.06	1.92	822
OSKL	**99.41±0.01**	**2.63**	149	82.01±0.32	4.08	1,344	**92.07±0.37**	1.37	825
SPA	**99.41±0.01**	**2.60**	149	**82.04± 0.25**	**3.84**	1,350	91.59±0.35	**1.31**	822
	w7a, $\beta = 200$			covtype, $\beta = 200$			gaussian, $\beta = 200$		
Algorithm	Accuracy (%)	Time	#SVs	Accuracy (%)	Time	#SVs	Accuracy (%)	Time	#SVs
Perceptron	97.64 ±0.04	2.50	1.1k	73.08±0.03	2861	156.0k	73.22±0.04	3507	268.5k
OGD	98.06±0.02	38.16	10.1k	78.34±0.03	5113	296.7k	80.36±0.01	5237	456.2k
PA-I	98.16±0.02	63.47	19.6k	78.18±0.05	4402	298.2k	78.95±0.02	5164	473.4k
RBP	96.78±0.16	0.68	175	65.63±0.23	50.27	1,510	72.87±0.52	19.42	1,100
Forgetron	96.36±0.06	0.72	175	65.33±0.22	67.60	1,510	72.94±0.03	25.08	1,100
Projectron	95.19±0.39	0.87	175	57.82±6.09	316.05	1,510	61.24±4.37	29.22	1,100
Projcetron++	95.90±0.38	3.01	175	59.33±5.85	470.35	1,510	60.03±7.93	30.36	1,100
BPAS	96.83±0.23	1.77	175	**72.37±0.13**	73.48	1,510	76.09±0.04	34.90	1,100
BOGD	96.75±0.03	1.05	175	68.75±0.03	54.44	1,510	79.21±0.01	23.27	1,100
OSKL	96.98±0.43	0.71	177	72.11±0.67	29.77	1,508	79.56±0.24	16.00	1,112
SPA	**97.05±0.13**	**0.56**	175	71.34±0.59	**27.53**	1,510	**79.82±0.15**	**15.51**	1,100

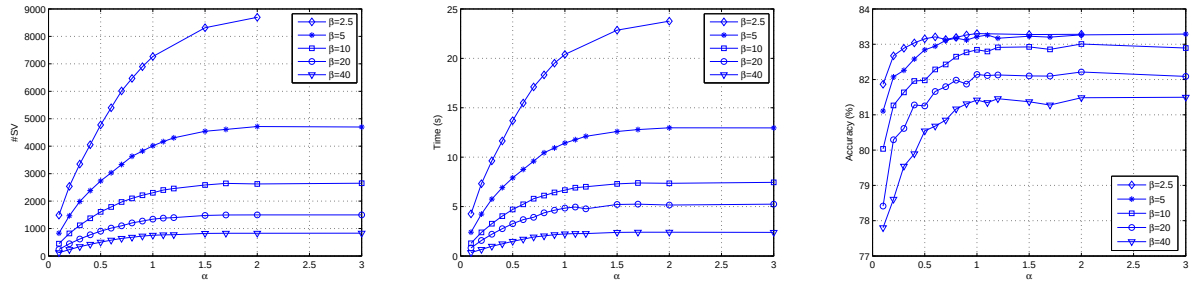
Table 3: Evaluation of Final Classifiers for Test Data (Time in Seconds).

	a9a			codrna			KDD08			w7a		
Algorithm	Acc. (%)	Time	#SV	Acc. (%)	Time	#SV	Acc. (%)	Time	#SV	Acc. (%)	Time	#SV
LIBSVM	85.04	97.8	11.5k	96.67	80.2	8.0k	99.39	577.3	14.6k	98.31	92.61	8.0k
Pegasos	84.66±0.21	15.8	11.0k	95.63±0.43	14.9	12.2k	99.50±0.01	976.8	81.8k	97.56±0.01	29.21	24.7k
BOGD	82.49±1.22	5.85	2,079	92.10±1.32	5.34	2,386	99.37±0.01	31.84	1,760	97.05±0.01	8.33	3,710
BPAS	82.11±0.83	7.59	2,079	93.12±2.01	6.98	2,386	99.22±0.29	37.48	1,760	97.75±0.40	13.95	3,710
OSKL-last	80.17±3.91	3.19	2,073	94.15±1.66	3.42	2,410	99.20±0.01	22.79	1,752	97.92±0.20	5.08	3,726
OSKL-avg	**84.91±0.13**	3.19	2,073	95.67±0.13	3.42	2,410	99.20±0.01	22.79	1,752	97.94±0.07	5.08	3,726
SPA-last	82.97±2.59	**3.16**	2,079	91.23±3.40	**3.03**	2,386	99.41±0.07	**19.42**	1,760	97.49±2.05	**5.04**	3,710
SPA-avg	84.88±0.10	**3.16**	2,079	**96.05±0.09**	**3.03**	2,386	**99.46±0.01**	**19.42**	1,760	**97.97±0.04**	**5.04**	3,710

4.4 Parameter Sensitivity of α and β The proposed SPA algorithm has two critical parameters α and β which could considerably affect the accuracy, support vector size, and time cost. Our second experiment is to examine how different parameters of α and β affect the learning performance so as to give insights for how to choose them in practice. Figure 1 evaluate the performance (support vector size, time, accuracy) of the SPA algorithm on the "a9a" dataset with varied α values ranging from 0.1 to 3, and varied β in the range of $\{2.5, 5, 10, 20, 40\}$. Several observations can be drawn from the experimental results.

First of all, when β is fixed, increasing α generally results in (i) larger support vector size, (ii) higher time cost, but (iii) better classification accuracy, especially when α is small. However, when α is large enough (e.g., $\alpha > 1.5$), increasing α has very minor impact to the performance. This is because the number of instances whose hinge loss above α is relatively small. We also note that the accuracy decreases slightly when α is too large, e.g., $\alpha >= 3$. This might be because some (potentially noisy) instances with large loss are given a high chance of being assigned as SVs, which may harm the classifier due to noise. Thus, on this dataset ("a9a"), it is easy to find a good α in the range of [1,2].

Second, when α is fixed, increasing β will result in (i) smaller support vector size, (ii) smaller time cost, but (iii) worse classification accuracy. On one hand, β cannot be too small as it will lead to too many support vectors and thus suffer very high time cost.



(a) Number of support vectors (b) Time Cost (seconds) (c) Accuracy (%)

Figure 1: The impact of α and β for #SV, time cost and accuracy by the proposed SPA algorithm on "a9a".

On the other than, β cannot be too large as it will considerably decrease the classification accuracy. We shall choose β that yields a sufficiently accurate classifier while minimizing the support vector size and training time cost. For example, for this particular dataset, choosing β in the range of [5,10] achieves a good trade-off between accuracy and efficiency/sparsity.

4.5 Evaluation of Output Classifiers on Test Data A key advantage of SPA is that it assures the final output averaged classifier is sparse, which is very important when applying the output classifier in real applications. Our last experiment thus is to examine if the final output classifier of SPA is effective for batch classification. We evaluate two SPA classifiers: "SPA-last" that simply outputs the classifier at the last iteration as the final classifier, and "SPA-avg" that outputs the average of classifiers at every iteration.

We compare our algorithms with two state-of-the-art batch algorithms for SVM classifications: (1) LIBSVM: a widely used and the most accurate solution of kernel SVM for batch classification [2]; (2) Pegasos[3][14]: an efficient stochastic gradient descent solver for SVM, for which we adapt it for kernel learning. Moreover, we compare our solutions with the output classifiers by two bounded kernel-based online online algorithms: BOGD and BPAS, as they achieve the best among all the existing algorithms in previous experiments. For these two algorithms, as their average classifier is not sparse, we compare with their last classifiers.

To enable fair comparisons, all the algorithms follow the same setup for batch classification. We conduct the experiments on 4 median-scale datasets as used in previous online experiments: "a9a", "codrna", "w7a" and "KDDCUP08" (the other two large datasets were excluded due to too long training time by LIBSVM). We use the original splits of training and test sets on the LIBSVM website. We adopt the same Gaussian kernel with the same kernel parameter γ for all the algorithms. We perform cross validation on the training set to search for the best parameters of different algorithms. In particular, we search for the best kernel parameter γ in the range of $\{2^5, 2^4, \ldots, 2^{-5}\}$, the parameter C of SVM in the range of $\{2^5, ...2^{-5}\}$, both the regularization parameter λ in Pegasos and BOGD and the learning rate parameter η in BOGD and SPA in the range of $\{10^3, 10^2, \ldots, 10^{-3}\}$. For the proposed SPA-last and SPA-avg, we set $\alpha = 1$ and $\beta = 5$ for all the datasets.

Table 3 shows the results, where we only report the test set accuracy and training time (we exclude the test time as it is proportional to SV sizes). We can draw some observations. First of all, in terms of training time, the budget algorithms run much faster than LIBSVM and Pegasos, in which our SPA algorithm achieves the lowest training time among all. Specifically, compared with batch algorithms, SPA achieves the speedup of training time for about 20 to 30 times over LIBSVM, and about 5 times over Pegasos.

Second, by examining the test set accuracy, we found that LIBSVM always achieves the best. The proposed SPA-avg algorithm achieves the best accuracy among all the budget algorithms, which is slightly lower but fairly comparable to LIBSVM, and even beats the accuracy of Pegasos that has almost 4 times more SVs than our SPA algorithm. This promising result validates the efficacy of our SPA algorithm for producing sparse and effective average classifier.

Third, we notice that BOGD, BPAS and SPA-last achieve similar test set accuracy, but their standard deviations are in general much larger than that of SPA-avg for most cases. This shows that employing the averaged classifier with SPA for test data classification leads to more accurate and more stable performance than many budget learning algorithms that output the last classifier, which again validates the advantage of our technique.

[3]http://www.cs.huji.ac.il/ shais/code/

5 Conclusions

To overcome the curse of kernelization in large-scale kernel methods, this paper proposed a novel framework of Sparse Passive Aggressive algorithm for bounded kernel-based online learning. In contrast to traditional budget kernel algorithms that only bound the number of support vectors at each iteration, our algorithm can output a sparse final averaged classifier with bounded support vector size, making it not only suitable for online learning tasks but also applicable to batch classification tasks. The experimental results from both online and batch classification tasks showed that the proposed method achieved a significant speedup over LIBSVM and yielded sparse and more accurate classifiers than existing online kernel methods, validating the efficacy, efficiency and scalability of the proposed technique.

Appendix A: Proof for Theorem 3.1

Proof. Firstly, the $P_t(f)$ defined in the equality

$$f_{t+1} = \min_{f\in\mathcal{H}_\kappa} P_t(f) := \frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} + \frac{Z_t}{\rho_t}\eta\ell_t(f)$$

is 1-strongly convex. Further, f_{t+1} is the optimal solution of $\min_f P_t(f)$, we thus have the following inequality according the definition of strongly convex

$$\begin{aligned}&\frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} + \frac{1}{\rho_t}Z_t\eta\ell_t(f)\\ &\geq \frac{1}{2}\|f_{t+1} - f_t\|^2_{\mathcal{H}_\kappa} + \frac{1}{\rho_t}Z_t\eta\ell_t(f_{t+1}) + \frac{1}{2}\|f - f_{t+1}\|^2_{\mathcal{H}_\kappa}\end{aligned}$$

where the inequality used $\nabla P_t(f_{t+1}) = 0$. After rearranging the above inequality, we get

$$\begin{aligned}&\frac{1}{\rho_t}Z_t\eta\ell_t(f_{t+1}) - \frac{1}{\rho_t}Z_t\eta\ell_t(f)\\ &\leq \frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} - \frac{1}{2}\|f - f_{t+1}\|^2_{\mathcal{H}_\kappa} - \frac{1}{2}\|f_{t+1} - f_t\|^2_{\mathcal{H}_\kappa}\end{aligned}$$

Secondly, since $\ell_t(f)$ is 1-Lipshitz with respect to f

$$\ell_t(f_t) - \ell_t(f_{t+1}) \leq \|f_t - f_{t+1}\|_{\mathcal{H}_\kappa}.$$

Combining the above two inequalities, we get

$$\begin{aligned}&\frac{1}{\rho_t}Z_t\eta[\ell_t(f_t) - \ell_t(f)] \leq \frac{1}{2}\|f - f_t\|^2_{\mathcal{H}_\kappa} - \frac{1}{2}\|f - f_{t+1}\|^2_{\mathcal{H}_\kappa}\\ &-\frac{1}{2}\|f_{t+1} - f_t\|^2_{\mathcal{H}_\kappa} + \frac{1}{\rho_t}Z_t\eta\|f_t - f_{t+1}\|_{\mathcal{H}_\kappa}\end{aligned}$$

Summing the above inequalities over all t leads to

$$\begin{aligned}\sum_{t=1}^{T}\frac{1}{\rho_t}Z_t\eta\left[\ell_t(f_t) - \ell_t(f)\right] &\leq \frac{1}{2}\|f - f_1\|^2_{\mathcal{H}_\kappa}\\ &+ \sum_{t=1}^{T}\left[-\frac{1}{2}\|f_{t+1} - f_t\|^2_{\mathcal{H}_\kappa} + \frac{1}{\rho_t}Z_t\eta\|f_t - f_{t+1}\|_{\mathcal{H}_\kappa}\right]\end{aligned} \tag{5.2}$$

We now take expectation on the left side. Note, by definition of the algorithm, $\mathbb{E}_t Z_t = \rho_t$, where we used $\mathbb{E}_t$ to indicate conditional expectation give all the random variables $Z_1, \ldots, Z_{t-1}$. Assuming $\rho_t > 0$, we have

$$\begin{aligned}&\mathbb{E}\big[\frac{1}{\rho_t}Z_t\eta\left[\ell_t(f_t) - \ell_t(f)\right]\big]\\ (5.3) = \mathbb{E}\big[\frac{1}{\rho_t}&\mathbb{E}_t Z_t\eta\left[\ell_t(f_t) - \ell_t(f)\right]\big] = \eta\mathbb{E}\left[\ell_t(f_t) - \ell_t(f)\right]\end{aligned}$$

Note that in some iterations, $\rho_t = 0$, in that case, we have $\ell_t(f_t) = 0$, thus:

$$\eta[\ell_t(f_t) - \ell_t(f)] \leq 0 \tag{5.4}$$

As mentioned before, $\rho_t = 0$ indicates $Z_t = 0$ and $Z_t/\rho_t = 0$, we get

$$\frac{1}{\rho_t}Z_t\eta[\ell_t(f_t) - \ell_t(f)] = 0 \tag{5.5}$$

Combining (5.3), (5.4) and (5.5) and summarizing over all t leads to

$$\eta\mathbb{E}\sum_{t=1}^{T}\left[\ell_t(f_t) - \ell_t(f)\right] \leq \mathbb{E}\sum_{t=1}^{T}\frac{1}{\rho_t}Z_t\eta\left[\ell_t(f_t) - \ell_t(f)\right]$$

We now take expectation on the right side of (5.2)

$$\begin{aligned}&\mathbb{E}\left[\frac{1}{2}\|f - f_1\|^2_{\mathcal{H}_\kappa}\right]\\ &+\mathbb{E}\left[\sum_{t=1}^{T}\left[-\frac{1}{2}\|f_{t+1} - f_t\|^2_{\mathcal{H}_\kappa} + \frac{1}{\rho_t}Z_t\eta\|f_t - f_{t+1}\|_{\mathcal{H}_\kappa}\right]\right]\\ &\leq \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + \sum_{t=1}^{T}\mathbb{E}\left[-\frac{1}{2}\tau_t^2 + \frac{1}{\rho_t}Z_t\eta\tau_t\right]\end{aligned}$$

Given all the random variables $Z_1, \ldots, Z_{t-1}$, we now calculate the conditional expectation of the variable $M_t = -\frac{1}{2}\tau_t^2 + \frac{1}{\rho_t}Z_t\eta\tau_t$: In probability ρ_t, $Z_t = 1$ and $\tau_t = \tau'_t = \min(\frac{\eta}{\rho_t}, \ell_t(f_t))$. We have $M_{t\ (Z_t=1)} = -\frac{1}{2}\tau_t'^2 + \frac{1}{\rho_t}\eta\tau'_t$. And in probability $1-\rho_t$, $Z_t = 0$ and $\tau_t = 0$. We have $M_{t\ (Z_t=0)} = 0$. Considering the two cases, the conditional expectation is:

$$\begin{aligned}\mathbb{E}_t[M_t] &= \rho_t M_{t\ (Z_t=1)} + (1-\rho_t)M_{t\ (Z_t=0)}\\ &= \rho_t\left[-\frac{1}{2}\tau_t'^2 + \frac{1}{\rho_t}\eta\tau'_t\right] < \eta\tau'_t\end{aligned}$$

In the case when $\alpha \leq \ell_t$ and $\rho_t = \frac{\alpha}{\beta}$, $\tau'_t = \min(\frac{\eta\beta}{\alpha}, \ell_t(f_t)) \leq \frac{\eta\beta}{\alpha}$, thus $\eta\tau'_t \leq \frac{\eta^2\beta}{\alpha}$.

And in the case $\alpha > \ell_t$ and $\rho_t = \frac{\ell_t}{\beta}$, $\tau'_t = \min(\frac{\eta\beta}{\ell_t(f_t)}, \ell_t(f_t)) \leq \sqrt{\eta\beta}$. Thus, $\eta\tau'_t \leq \frac{\eta^2\beta}{\sqrt{\beta\eta}}$.

Considering both of the cases leads to

$$\mathbb{E}_t[M_t] < \frac{\eta^2\beta}{\min(\alpha, \sqrt{\beta\eta})}$$

Summing the above inequality over all t and combining with (5.6), we get

$$\eta\mathbb{E}\sum_{t=1}^{T}[\ell_t(f_t) - \ell_t(f)] < \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + \frac{\eta^2\beta}{\min(\alpha, \sqrt{\beta\eta})}T$$

Setting $f = f_*$, and multiplying the above inequality with $1/\eta$ will conclude the theorem.

Note that f_{t+1} is also a random variable depending on Z_t. Thus, when taking the expectation on both sides of (5), special attention should be taken. Our approach suggests a proper way to fix some mistake in [20].

Appendix B: Proof for Theorem 3.2

Proof. Since $\mathbb{E}_t[Z_t] = \rho_t$, where $\mathbb{E}_t$ is the conditional expectation, we have

$$\mathbb{E}[\sum_{t=1}^{T} Z_t] = \mathbb{E}[\sum_{t=1}^{T} \mathbb{E}_t Z_t] = \mathbb{E}[\sum_{t=1}^{T} \rho_t] =$$

$$\mathbb{E}[\sum_{t=1}^{T} \min(\frac{\alpha}{\beta}, \frac{\ell_t(f_t)}{\beta})] \leq \min(\frac{\alpha}{\beta}T, \frac{1}{\beta}\mathbb{E}\sum_{t=1}^{T}\ell_t(f_t)) \leq$$

$$\min\left\{\frac{\alpha}{\beta}T, \frac{1}{\beta}[\sum_{t=1}^{T}\ell_t(f_*) + \frac{1}{2\eta}\|f_*\|^2_{\mathcal{H}_\kappa} + \frac{\eta\beta}{\min(\alpha, \sqrt{\beta\eta})}T]\right\}$$

which concludes the first part of the theorem. The second part of the theorem is trivial to be derived.

References

[1] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

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

[3] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[4] Koby Crammer, Jaz S Kandola, and Yoram Singer. Online classification on a budget. In *NIPS*, volume 2, page 5, 2003.

[5] Ofer Dekel. From online to batch learning with cutoff-averaging. In *Advances in Neural Information Processing Systems*, pages 377–384, 2009.

[6] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008.

[7] Ofer Dekel and Yoram Singer. Data-driven online to batch conversions. In *Advances in Neural Information Processing Systems*, pages 267–274, 2005.

[8] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, 1999.

[9] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[10] Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *NIPS*, pages 785–792, 2001.

[11] Jyrki Kivinen, Alex J Smola, and Robert C Williamson. Online learning with kernels. In *NIPS*, pages 785–792, 2001.

[12] Francesco Orabona, Joseph Keshet, and Barbara Caputo. Bounded kernel-based online learning. *The Journal of Machine Learning Research*, 10:2643–2666, 2009.

[13] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[14] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[15] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *arXiv preprint arXiv:1212.1824*, 2012.

[16] Vladimir N Vapnik. Statistical learning theory. 1998.

[17] Zhuang Wang, Koby Crammer, and Slobodan Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *Journal of Machine Learning Research*, 13:3103–3131, 2012.

[18] Zhuang Wang and Slobodan Vucetic. Twin vector machines for online learning on a budget. In *SDM*, pages 906–917. SIAM, 2009.

[19] Zhuang Wang and Slobodan Vucetic. Online passive-aggressive algorithms on a budget. In *International Conference on Artificial Intelligence and Statistics*, pages 908–915, 2010.

[20] Lijun Zhang, Rong Jin, Chun Chen, Jiajun Bu, and Xiaofei He. Efficient online learning for large-scale sparse kernel logistic regression. In *AAAI*, 2012.

[21] Lijun Zhang, Jinfeng Yi, Rong Jin, Ming Lin, and Xiaofei He. Online kernel learning with a near optimal sparsity bound. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 621–629, 2013.

[22] Peilin Zhao, Jialei Wang, Pengcheng Wu, Rong Jin, and Steven CH Hoi. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *ICML*, 2012.