Online Kernel Learning with a Near Optimal Sparsity Bound

Lijun Zhang ZHANGLIJ@MSU.EDU
Jinfeng Yi YIJINFEN@MSU.EDU
Rong Jin RONGJIN@CSE.MSU.EDU
Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA

Ming Lin LIN-M08@MAILS.TSINGHUA.EDU.CN
Department of Automation, Tsinghua University, Beijing 100084, China

Xiaofei He XIAOFEIHE@CAD.ZJU.EDU.CN
State Key Laboratory of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou 310027, China

Abstract

In this work, we focus on *Online Sparse Kernel Learning* that aims to online learn a kernel classifier with a bounded number of support vectors. Although many online learning algorithms have been proposed to learn a sparse kernel classifier, most of them fail to bound the number of support vectors used by the final solution which is the average of the intermediate kernel classifiers generated by online algorithms. The key idea of the proposed algorithm is to measure the difficulty in correctly classifying a training example by the derivative of a smooth loss function, and give a more chance to a difficult example to be a support vector than an easy one via a sampling scheme. Our analysis shows that when the loss function is smooth, the proposed algorithm yields similar performance guarantee as the standard online learning algorithm but with a near optimal number of support vectors (up to a $poly(\ln T)$ factor). Our empirical study shows promising performance of the proposed algorithm compared to the state-of-the-art algorithms for online sparse kernel learning.

1. Introduction

Kernel methods (Schölkopf & Smola, 2002), such as support vector machine (SVM) (Burges, 1998) and kernel logistic regression(KLR) (Roth, 2001), are widely used in statistical learning. Many online learning algorithms have been proposed to improve the computational efficiency of kernel methods (Freund & Schapire, 1999; Kivinen et al., 2002; Cheng et al., 2007). Typically, online kernel learning receives training examples $(\mathbf{x}_t, y_t), t = 1, \ldots, T$ in sequel, and generates a sequence of kernel classifiers $\{f_1, \ldots, f_T\}$ based on the observed examples. The final solution $\hat{f}$ is obtained by taking the average of the intermediate classifiers, i.e., $\hat{f} = \frac{1}{T}\sum_{t=1}^{T} f_t$, a procedure often referred to as online-to-batch conversion (Cesa-Bianchi et al., 2004). The main problem of online kernel learning is that the number of support vectors used to construct the intermediate f_t may grow unboundedly, leading to a large storage requirement and a high computational cost for both training and testing.

The focus of this work is online sparse kernel learning that learns a kernel classifier with a limited number of support vectors. To generate sparse kernel classifiers, a common approach is to use a non-smooth loss function (e.g. hinge loss) whose derivative becomes zero when a data point is classified correctly with sufficiently large margin (Kivinen et al., 2002). The drawback of this approach is that it is unable to provide a formal bound on the number of support vectors.

An alternative approach for online sparse kernel learning is based on sampling (Zhang et al., 2012). It online determines if a training example $(\mathbf{x}_t, y_t)$ is a support vector based on $p(y_t|f_t(\mathbf{x}_t))$, the probability of correctly classifying $(\mathbf{x}_t, y_t)$. The larger the probability $p(y_t|f_t(\mathbf{x}_t)$, the less likely $(\mathbf{x}_t, y_t)$ will be used as a support vector. Similar to the approach of using non-smooth loss function, this method is not equipped with

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

a formal bound on the number of support vectors.

Budget online learning methods (Cavallanti et al., 2007; Dekel et al., 2008) were proposed to control the number of support vectors. It maintains through the iterations, a sequence of kernel classifiers with a fixed number of support vectors. The main shortcoming of budget online learning is that although the number of support vectors is bounded for each intermediate classifier f_t, it is usually not the case for the final solution, which is computed as the average of the intermediate classifiers. This is verified by our empirical study.

In this paper, we develop an online sparse kernel learning algorithm by utilizing a *sampling* approach and a *smooth* loss function $\ell(y, z)$. We note that we slightly abuse the term "smooth" as the smooth loss function defined in this work is slightly different from the conventional definition. The key idea of the proposed algorithm is to measure the difficulty in classifying a training example $(\mathbf{x}_t, y_t)$ by the derivative $\ell'(y_t, f_t(\mathbf{x}_t))$, instead of the loss $\ell(y_t, f_t(\mathbf{x}_t))$, and give more chance for a "difficult" example to be a support vector than an "easy" one via a sampling procedure. We choose the derivative for the difficulty measurement because it leads to an unbiased estimate of the gradient, an important property for our analysis.

Using a smooth loss function may sound counter-intuitive because it usually leads to dense kernel classifiers. One nice property of a smooth loss function is that its derivative directly reflects the degree of misclassification. As a result, given kernel classifier $f(\cdot)$ learned from a smooth loss, if we randomly select its support vector $(\mathbf{x}_t, y_t)$ based on the derivative $\ell'(y_t, f(\mathbf{x}_t))$, most of the support vectors that are difficult to be classified by $f(\cdot)$ will be kept, which allows us to preserve the core of $f(\cdot)$. More specifically, our theoretical analysis reveals the following important properties of the proposed algorithm compared to the available methods for online kernel learning:

- Unlike the existing approaches for sparse online kernel learning, we provide not only the regret bound for the proposed algorithm but also the bound for the number of support vectors. Our analysis also shows that the bound for the number of support vectors achieved by the proposed algorithm is tight up to a $poly(\ln T)$ factor.
- Unlike budget online learning that only bounds the number of support vectors for intermediate solutions, the proposed algorithm guarantees sparse solutions for both the intermediate classifiers and the final classifier.
- The proposed algorithm provides a flexible mechanism to control the number of support vectors and allows users to make appropriate tradeoff between classifier sparsity and classification accuracy.

2. Related Work

In this section, we briefly review the existing work on sparse learning.

Sparse Kernel Learning in Batch Setting A number of algorithms have been developed for batch sparse kernel learning (Burges & Schölkopf, 1997; Lee & Mangasarian, 2001; Zhu & Hastie, 2001; Keerthi et al., 2006). In post-processing based approach (Cotter et al., 2013), a non-sparse kernel classifier is first learned from the training examples, and a sparse solution is then computed to approximate the dense solution. An alternative approach is to reformulate the kernel learning problem such that a sparse solution is guaranteed (Wu et al., 2006).

Online Sparse Kernel Learning Most online sparse kernel learning algorithms are built upon non-smooth loss functions, and none of them is able to provide explicit bound on the number of support vectors for the final solution[1]. Finally, it is worthwhile noting that our work is closely related to the recent work on sparse kernel logistic regression (Zhang et al., 2012) in that both adopt a sampling strategy for reducing the number of support vectors, they differ significantly in the sampling procedure. More importantly, the proposed algorithm here achieves a bounded number of support vectors while (Zhang et al., 2012) did not.

Online Kernel Learning on a Budget The objective of budget online kernel learning is to generate a sequence of kernel classifiers with a fixed number of support vectors. In (2002), Kivinen et al. consider online kernel learning for regularized losses, where the coefficients of support vectors are shrunk by a small constant at each iteration. To improve the sparsity, they propose to drop the support vectors with smallest coefficients. Forgetron (Dekel et al., 2008) applies a similar idea and turns kernel perceptron into a budget kernel learning algorithm. In randomized budget perceptron (Cavallanti et al., 2007) and bounded online gradient descent (Zhao et al., 2012), random sampling approaches are developed to remove support vec-

[1] Although mistake bounds given in these studies are closely related to the number of support vectors, they can not be translated into the sparsity guarantee for the final kernel classifier since binary loss function is used in the mistake bound analysis.

tors when the number of support vectors exceeds the budget.

In (2004), Crammer et al. develop a heuristic approach that removes *redundant* support vectors. A similar but more sophisticated strategy is developed in Projectron (Orabona et al., 2008), which introduces a new support vector only when it cannot be well approximated by the existing ones. Recently, Wang et al. present budgeted stochastic gradient descent for kernel SVM through several budget maintenance strategies (Wang et al., 2012).

Since budget online learning typically uses both insertion and deletion operations to control the number of support vectors, the final solution, i.e., the average of intermediate classifiers, is usually dense in the number of support vectors. In contrast, our algorithm only allows insertion operation, leading to a sparse kernel classifier even for the final solution.

Online Learning for Sparse Linear Models Several online methods have been proposed to learn sparse *linear* models (Langford et al., 2009; Duchi & Singer, 2009). These approaches cannot be applied directly to kernel learning because they rely on the ℓ_1 regularization and are developed specifically for linear classification.

3. Online Sparse Kernel Learning by Sampling and Smooth Losses (OSKL)

Before we describe our algorithm, we first define a few notations that will be used throughout the paper.

Let $\kappa(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be a kernel function, and $\mathcal{H}_\kappa$ be the reproducing kernel Hilbert space (RKHS) endowed with κ. For simplicity, we assume $\kappa(\mathbf{x}, \mathbf{x}) \leq 1$ for any $\mathbf{x} \in \mathcal{X}$. Let $\mathcal{B} = \{f \in \mathcal{H}_\kappa : \|f\|_{\mathcal{H}_\kappa} \leq R\}$ be the solution domain, where $R > 0$ specifies the domain size. We use $\pi_{\mathcal{B}}(f)$ for the projection of a function $f(\cdot) \in \mathcal{H}_\kappa$ into the domain $\mathcal{B}$, and $\mathrm{sgn}(x)$ for the sign function that outputs $+1$, 0, and -1 when x is positive, zero, and negative, respectively.

Let $\ell(y, z)$ be a non-negative loss function convex in the second argument. Similar to most online learning algorithms, we assume $\ell(y, z)$ to be Lipschitz continuous in the second argument, i.e.,

$$\mathbf{A1} \qquad |\ell'(y, z)| \leq G_1, \quad \forall z \in \mathcal{Z},$$

where $\mathcal{Z}$ is the domain for the predicted value[2]. Besides being Lipschitz continuous, we also assume that the magnitude of the derivative is upper bounded by the loss, i.e.,

$$\mathbf{A2} \qquad |\ell'(y, z)| \leq L\ell(y, z), \quad \forall z \in \mathcal{Z},$$

where $L > 0$ is a constant independent from y and z.

Table 1. Loss function satisfying condition **A2**

Name	Definition of $\ell(y, z)$	L
Logit loss	$\ln(1 + \exp(-yz))$, $y \in \{\pm 1\}$	1
Exponential loss	$\exp(-yz)$, $\lvert y\rvert \leq d$	d
Square loss	$(y - z)^2$, $\lvert y - z\rvert \geq \delta > 0$	$\frac{2}{\delta}$

Remark 1 It is the assumption **A2** that makes it possible to bound both the regret and the sparsity of the kernel classifier simultaneously. It is straightforward to check that assumption **A2** holds for logit loss $\ell(y, z) = \ln(1 + \exp(-yz))$ because

$$|\ell'(y, z)| = 1 - \frac{1}{1 + \exp(-yz)}$$
$$\leq -\ln \frac{1}{1 + \exp(-yz)} = \ell(y, z).$$

Table 1 shows a few examples of loss functions that satisfy condition **A2**. We note that condition **A2** is closely related to the conventional definition of H-smooth loss function. This is because, using Lemma 2.1 from (Srebro et al., 2010), it is easy to show that for any H-smooth loss function $\ell(y, z)$, if the absolute value of the derivative $|\ell'(y, z)|$ is bounded from below by a constant G_0 for domain $\mathcal{Z}$, it will satisfy assumption **A2** with $L = 4H/G_0$. Algorithm 1 shows the detailed steps of the proposed Online Sparse Kernel Learning (OSKL) algorithm. At each iteration, it first computes the derivative $\ell'(y_t, f_t(\mathbf{x}_t))$ and then samples a binary variable Z_t with

$$\Pr(Z_t = 1) = \frac{1}{G}|\ell'(y_t, f_t(\mathbf{x}_t))|,$$

where parameter G is introduced to adjust the sampling probability. Training example $(\mathbf{x}_t, y_t)$ is added to the kernel classifier as a support vector only when $Z_t = 1$. It is this sampling scheme that allows us to control the number of support vectors.

Note that we choose the derivative $|\ell'(y_t, f_t(\mathbf{x}_t))|$, instead of the loss $\ell(y_t, f_t(\mathbf{x}_t))$, as the basis for sampling. This is because using the derivative based sampling, the resulting gradient $g(\cdot)$ computed in (1) will be an

[2] We denote the partial derivative of $\ell(y, z)$ w.r.t. its second argument by $\ell'(y, z)$, i.e., $\ell'(y, z) = \partial\ell(y, z)/\partial z$.

Algorithm 1 Online Sparse Kernel Learning (OSKL)

Input: step size η, domain size R, and parameter $G \geq G_1$

1: Initialize $f_1(\mathbf{x}) = 0$
2: **for** $t = 1, \ldots, T$ **do**
3: Receive an example $(\mathbf{x}_t, y_t)$
4: Compute the derivative $\ell'(y_t, f_t(\mathbf{x}_t))$
5: Sample a binary random variable Z_t with

$$\Pr(Z_t = 1) = \frac{1}{G}|\ell'(y_t, f_t(\mathbf{x}_t))|$$

6: Update the classifier by

$$f_{t+1}(\cdot) = \pi_{\mathcal{B}}\left[f_t(\cdot) - \eta g_t(\cdot)\right]$$

where

$$g_t(\cdot) = \mathrm{sgn}(\ell'(y_t, f_t(\mathbf{x}_t)))Z_t G\kappa(\mathbf{x}_t, \cdot) \tag{1}$$

7: **end for**

Output $\hat{f}(\cdot) = \frac{1}{T}\sum_{t=1}^{T} f_t(\cdot)$

unbiased estimate of the true gradient $\nabla\ell(y_t, f_t(\mathbf{x}_t))$, i.e.,

$$\mathrm{E}\left[g(\cdot)\right] = \nabla\ell(y_t, f_t(\mathbf{x}_t)) = \ell'(y_t, f_t(\mathbf{x}_t))\kappa(\mathbf{x}_t, \cdot).$$

This property is the key to the analysis of the regret bound and the sparsity for the proposed algorithm. In addition, since $|\ell'(y_t, f_t(\mathbf{x}_t))|$ is assumed to be bounded by $\ell(y_t, f_t(\mathbf{x}_t)$ in condition **A2**, using derivative for sampling may result in a smaller number of support vectors.

Evidently, the number of support vectors of $\hat{f}$ is given by $\sum_{t=1}^{T} Z_t$. The following theorem shows that both the regret and the number of supported vectors can be bounded by the cumulative loss of the optimal kernel classifier.

Theorem 1. *Assume that loss function $\ell(y, z)$ satisfies the assumptions* **A1**, **A2**, *and $T \geq 18/[G\ln(1/\delta)]$. For a fixed $\gamma \in (0, 1)$, we set $\eta \leq \gamma/(LG)$. Let $f_1, \ldots, f_T$ be the sequence of classifiers generated by by Algorithm 1. With probability at least $1 - 2\delta$, for any $f_* \in \mathcal{B}$, we have*

$$\begin{aligned}\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) \leq \frac{1}{1-\gamma}\sum_{t=1}^{T}\ell(y_t, f_*(\mathbf{x}_t)) \\ + \frac{\eta G^2}{(1-\gamma)}c + \frac{R^2}{(1-\gamma)\eta}c + \frac{RG}{1-\gamma}c + Rc,\end{aligned} \tag{2}$$

and

$$\sum_{t=1}^{T} Z_t \leq \frac{3L}{2G}\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) + c, \tag{3}$$

where

$$c = \max\left\{2\sqrt{G_1}, \frac{1}{2} + 16\ln\frac{m}{\delta}, \frac{8}{3}\ln\frac{m}{\delta} + \frac{19}{18}\ln\frac{1}{\delta}\right\}, \tag{4}$$

and $m = \lceil\log_2(G_1 T^2)\rceil$.

Remark 2 Note that although in (3) we bound the number of support vectors $\sum_{t=1}^{T} Z_t$ by $\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t))$, it is easy to relate the number of support vectors to the loss of the optimal classifier f_* using the bound in (2). To better understand the structure of the bounds in (2) and (3), we set $\eta = \gamma/(LG)$, leading to the following bounds for the regret and the number of support vectors

$$\begin{aligned}&\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) - \left(\frac{1}{1-\gamma}\sum_{t=1}^{T}\ell(y_t, f_*(\mathbf{x}_t)) + Rc\right) \\ \leq& G\left(\frac{\gamma}{L(1-\gamma)} + \frac{LR^2}{(1-\gamma)\gamma} + \frac{R}{1-\gamma}\right)c,\end{aligned}$$

and

$$\begin{aligned}&\sum_{t=1}^{T} Z_t - \left(1 + \frac{3\gamma}{2(1-\gamma)} + \frac{3L^2R^2}{2(1-\gamma)\gamma} + \frac{3LR}{2(1-\gamma)}\right)c \\ &\qquad \leq \frac{1}{G}\left(\frac{3L}{2(1-\gamma)}\sum_{t=1}^{T}\ell(y_t, f_*(\mathbf{x}_t)) + \frac{3LR}{2}c\right).\end{aligned}$$

It is not surprising to observe that the smaller the G, the smaller the regret and the larger the number of support vectors. Thus, parameter G allows us to compromise between classifier sparsity and classification accuracy.

To check the tightness of the bounds for Algorithm 1, we examine if it is always possible to devise an algorithm that achieves a similar regret bound as Algorithm 1 but with a significantly smaller number of support vectors. The answer to this question, as revealed by the following theorem, is no. We defer the proof to the supplementary document.

Theorem 2. *For any fixed integers n and T, there always exists a sequence of training examples $\{\mathbf{x}_t, y_t\}_{t=1}^{T}$ such that*

- *the optimal classifier f_* has n support vectors and $O(n)$ loss,*
- *the sequence of kernel classifiers $f_1, \ldots, f_T$ returned by Algorithm 1 achieves a regret bound of $O(n[\ln T]^2)$ with $O(n[\ln T]^2)$ support vectors, and*
- *the regret of any sequence of kernel classifiers $f_1', \ldots, f_T'$ with less than n support vectors is at least $\Omega(T/n)$.*

The result in Theorem 2 indicates that the bound for the number of support vectors achieved by Algorithm 1 is optimal up to a $poly(\ln T)$ factor.

4. Analysis

In this section, we present the analysis that leads to Theorem 1. To simplify the notations, we define

$$\tau_t = \mathrm{sgn}(\ell'(y_t, f_t(\mathbf{x}_t))), \text{ and } A_T = \sum_{t=1}^{T} |\ell'(y_t, f_t(\mathbf{x}_t))|.$$

In the analysis below, we consider two different scenarios, i.e., $A_T \leq 1/T$ and $A_T > 1/T$.

4.1. Bounds When $A_T \leq 1/T$

Under this condition, we have

$$\begin{aligned}
&\sum_{t=1}^{T} \ell(y_t, f_t(\mathbf{x}_t)) - \ell(y_t, f_*(\mathbf{x}_t)) \\
\leq &\sum_{t=1}^{T} \ell'(y_t, f_t(\mathbf{x}_t))(f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t)) \\
\leq &\sum_{t=1}^{T} \alpha \ell'(y_t, f_t(\mathbf{x}_t))^2 + \frac{(f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t))^2}{4\alpha} \\
\leq &\alpha \frac{G_1}{T} + \frac{TR^2}{\alpha} = 2R\sqrt{G_1},
\end{aligned} \tag{5}$$

where we set $\alpha = TR/\sqrt{G_1}$.

In addition, we can also bound the number of support vectors, i.e., $\sum_{t=1}^{T} Z_t$, by using Bernstein's inequality for martingales (Cesa-Bianchi & Lugosi, 2006).

Lemma 1. *Assume $A_T \leq 1/T$ and $T \geq 18/[G\ln(1/\delta)]$. With probability at least $1-\delta$, we have*

$$\sum_{t=1}^{T} GZ_t - |\ell'(y_t, f_t(\mathbf{x}_t))| \leq G \ln \frac{1}{\delta}.$$

The proof is provided in the supplementary document.

As a result, with probability at least $1-\delta$, we have

$$\sum_{t=1}^{T} Z_t \leq \frac{1}{GT} + \ln \frac{1}{\delta} \leq \frac{19}{18} \ln \frac{1}{\delta}. \tag{6}$$

4.2. Bounds When $A_T > 1/T$

We first consider bounding the number of support vectors $\sum_{t=1}^{T} Z_t$. Similar to the previous analysis, we derive an upper bound for $\sum_{t=1}^{T} GZ_t - |\ell'(y_t, f_t(\mathbf{x}_t))|$. However, in this case, there is no tight upper bound for A_T, so we cannot follow the approach used in the proof of Lemma 1. Instead, we combine the Bernstein's inequality for martingales (Cesa-Bianchi & Lugosi, 2006) with the peeling process introduced in (Bartlett et al., 2005), to give an upper bound involving the overall loss $\sum_{t=1}^{T} \ell(y_t, f_t(\mathbf{x}_t))$. To this end, we have the following lemma.

Lemma 2. *Assume $A_T > 1/T$. With probability at least $1-\delta$, we have*

$$\sum_{t=1}^{T} GZ_t - |\ell'(y_t, f_t(\mathbf{x}_t))| \leq 2\sqrt{GA_T \ln \frac{m}{\delta}} + \frac{2}{3} G \ln \frac{m}{\delta},$$

where $m = \lceil \log_2(G_1 T^2) \rceil$.

The proof is provided in the supplementary document.

Following Lemma 2, with probability at least $1-\delta$, we have

$$\begin{aligned}
&\sum_{t=1}^{T} Z_t \\
\leq &\frac{1}{G}\left(A_T + 2\sqrt{GA_T \ln \frac{m}{\delta}} + \frac{2}{3} G \ln \frac{m}{\delta}\right) \\
\leq &\frac{1}{G}\left(A_T + \frac{1}{2}A_T + 2G \ln \frac{m}{\delta} + \frac{2}{3} G \ln \frac{m}{\delta}\right) \\
= &\frac{3L}{2G} \sum_{t=1}^{T} \ell(y_t, f_t(\mathbf{x}_t)) + \frac{8}{3} \ln \frac{m}{\delta}.
\end{aligned} \tag{7}$$

Now, we give the analysis of the regret bound. Using the standard analysis of online learning (Cesa-Bianchi & Lugosi, 2006), we have

$$\begin{aligned}
&\ell(y_t, f_t(\mathbf{x}_t)) - \ell(y_t, f_*(\mathbf{x}_t)) \\
\leq &\langle \ell'(y_t, f_t(\mathbf{x}_t)) \kappa(\mathbf{x}_t, \cdot), f_t - f_* \rangle_{\mathcal{H}_\kappa} \\
= &\tau_t Z_t G \langle \kappa(\mathbf{x}_t, \cdot), f_t - f_* \rangle_{\mathcal{H}_\kappa} + \\
&(\ell'(y_t, f_t(\mathbf{x}_t)) - \tau_t Z_t G)(f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t)) \\
\leq &\frac{\|f_t - f_*\|_{\mathcal{H}_\kappa}^2 - \|f_{t+1} - f_*\|_{\mathcal{H}_\kappa}^2}{2\eta} + \frac{\eta G^2}{2} Z_t + \\
&\tau_t \left(|\ell'(y_t, f_t(\mathbf{x}_t))| - GZ_t\right)\left(f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t)\right).
\end{aligned} \tag{8}$$

By adding the inequality in (8) over all the iterations, we have

$$\begin{aligned}
&\sum_{t=1}^{T} \ell(y_t, f_t(\mathbf{x}_t)) - \ell(y_t, f_*(\mathbf{x}_t)) \\
\leq &\frac{\|f_*\|_{\mathcal{H}_\kappa}^2}{2\eta} + \frac{\eta G^2}{2} \sum_{t=1}^{T} Z_t + \\
&\sum_{t=1}^{T} (|\ell'(y_t, f_t(\mathbf{x}_t))| - GZ_t)(\tau_t [f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t)]).
\end{aligned} \tag{9}$$

Following (7), we can bound the second expression on the R.H.S of the above inequality as

$$\begin{aligned}&\frac{\eta G^2}{2}\sum_{t=1}^{T} Z_t\\ \le&\frac{3}{4}\eta LG\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) + \frac{4}{3}\eta G^2\ln\frac{m}{\delta} \qquad (10)\\ \le&\frac{3}{4}\gamma\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) + \frac{4}{3}\eta G^2\ln\frac{m}{\delta},\end{aligned}$$

where the last inequality follows from the condition $\gamma \geq \eta LG$. Similar to Lemma 2, we develop the following Lemma to bound the last expression in (9).

Lemma 3. *Assume $A_T > 1/T$. With probability at least $1-\delta$, we have*

$$\begin{aligned}&\sum_{t=1}^{T}(|\ell'(y_t, f_t(\mathbf{x}_t))| - GZ_t)(\tau_t[f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t)])\\ \le&4R\sqrt{GA_T\ln\frac{m}{\delta}} + \frac{4}{3}RG\ln\frac{m}{\delta},\end{aligned}$$

where $m = \lceil\log_2(G_1T^2)\rceil$.

We skip the proof since it is identical to that for Lemma 2. From Lemma 3, we have

$$\begin{aligned}&\sum_{t=1}^{T}(|\ell'(y_t, f_t(\mathbf{x}_t))| - GZ_t)(\tau_t[f_t(\mathbf{x}_t) - f_*(\mathbf{x}_t)])\\ \le&\frac{\gamma}{4L}A_T + \frac{4L}{\gamma}4R^2G\ln\frac{m}{\delta} + \frac{4}{3}RG\ln\frac{m}{\delta} \qquad (11)\\ \le&\frac{1}{4}\gamma\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) + \frac{16}{\eta}R^2\ln\frac{m}{\delta} + \frac{4}{3}RG\ln\frac{m}{\delta}.\end{aligned}$$

Combining the bounds in (9), (10) and (11), we have, with probability at least $1-2\delta$,

$$\begin{aligned}&\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) - \sum_{t=1}^{T}\ell(y_t, f_*(\mathbf{x}_t))\\ \le&\frac{1}{2\eta}\left(\|f_*\|^2_{\mathcal{H}_\kappa} + 32R^2\ln\frac{m}{\delta}\right) + \gamma\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t)) +\\ &\frac{4}{3}\eta G^2\ln\frac{m}{\delta} + \frac{4}{3}RG\ln\frac{m}{\delta},\end{aligned}$$

which implies

$$\begin{aligned}&\sum_{t=1}^{T}\ell(y_t, f_t(\mathbf{x}_t))\\ \le&\frac{1}{1-\gamma}\sum_{t=1}^{T}\ell(y_t, f_*(\mathbf{x}_t)) + \frac{4\eta G^2}{3(1-\gamma)}\ln\frac{m}{\delta} + \qquad (12)\\ &\frac{R^2}{2(1-\gamma)\eta}\left(1 + 32\ln\frac{m}{\delta}\right) + \frac{4RG}{3(1-\gamma)}\ln\frac{m}{\delta}.\end{aligned}$$

Table 2. Data statistics

Data Sets	# Training	# Testing	# Features
Magic	15,216	3,804	10
Adult	32,561	16,281	123
Covtype	464,809	116, 203	54

Using the definition of c in (4), we obtain (2) by combining (5) and (12), and (3) by combining (6) and (7).

5. Experiment

In this section, we perform classification experiments to demonstrate the advantage of the proposed method. We use 3 benchmark data sets which are summarized in Table 2. Magic is available at UCI Machine Learning Repository (Frank & Asuncion, 2010), while the others are downloaded from LIBSVM data sets (Chang & Lin, 2011). For Magic and Covtype, we randomly select 80% data for training and repeat the experiments 5 times with different training and testing splits. For Adult, we use the training and testing splits provided by the authors, and also repeat the experiments 5 times by processing the training data in different random permutations. For unnormalized data sets, we linearly scale each feature to the range $[0, 1]$. We choose the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$, and set the kernel width σ to the 20-th percentile of the pairwise distances (Mallapragada et al., 2009). We choose the logit loss for our Online Sparse Kernel Learning (OSKL) algorithm.

5.1. Comparison with online sparse kernel learning

We first compare the proposed OSKL with three online sparse kernel learning algorithms:

- **Margin** and **Auxiliary**, two online learning algorithms designed for sparse kernel logistic regression (Zhang et al., 2012);
- **Pegasos**, an online learning method for kernel SVM, which produce sparse classifiers by using hinge loss (Shalev-Shwartz et al., 2007).

Besides, we also report the result of **Baseline**, which applies stochastic gradient descent to solve kernel logistic regression. The regularization parameter λ in Pegasos is searched in the range of $\{1e{-}10, \ldots, 1e{-}1\}$, the parameter R in other four algorithms is searched

in $\{1, 1e1, \ldots, 1e5\}$, the step size η in Baseline, Margin, and Auxiliary is searched in $\{1e{-}2, 1e{-}1, 1\}$, and their values are determined by cross validation. The parameters γ and η in OSKL are set to be 0.9 and γ/G, respectively. In the experiments, we formally defined sparsity as the ratio between the number of non-support vectors and the number of received training examples (Zhang et al., 2012).

Table 3 shows the average as well as the standard deviation of the classification accuracy, the sparsity (SP), the number of support vectors (SVs), and the training time on each data set[3]. Compared to Baseline, all the other fours algorithms are able to generate sparse classifiers, that dramatically reduce the training time, and at the same time still maintain high classification accuracy. We can see the classifier generated by OSKL achieves the highest sparsity among all the methods in comparison. Based on the results of OSKL on different data sets, we also observe that the easier the classification task, the higher the sparsity. This is consistent with our analysis, i.e., the number of support vectors is bounded by the loss of the optimal classifier. Finally, we observe that increasing the value of G improves the sparsity measure, but at the cost of small reduction in classification accuracy.

5.2. Comparison with budget online learning

In this section, we compare OSKL with Forgetron (Dekel et al., 2008), which is a classic budget online learning algorithm, and bounded online gradient descent using nonuniform sampling (BOGD++) (Zhao et al., 2012), which has shown to be superior to other budget online learning algorithms, such as Projectron (Orabona et al., 2008). To make the result comparable, we set the budget size to be the number of support vectors used by OSKL.

In the first comparison, we evaluate the performance of the last classifier f_T generated by the two budget online learning algorithms, and summarize the classification performance in Table 4. We omit the sparsity and the number of support vector of f_T from Table 4, because they are the same as OSKL and can be found from Table 3. Since the budget is set as the number of support vectors used by OSKL, it is not surprising that the training time of Forgetron and BOGD++ is overall comparable to that of OSKL. However, the classification accuracy of the last classifiers f_T generated by the budget online learning algorithms is significantly worse than that of OSKL in most cases. Besides, the performance of the last classifier is unstable, as indicated by the large standard deviation.

Second, we evaluate the performance of the average classifier $\hat{f} = \frac{1}{T}\sum_{t=1}^{T} f_t$, a common approach that converts online learning solutions into a batch learning solution. Table 5 summarizes the classification accuracy, the sparsity and the number of support vectors for the average classifier. The training time was omitted from Table 5 as it is already listed in Table 4. We observe that, using the average classifier, both Forgetron and BOGD++ achieve similar classification accuracy as OSKL. Compared to the results in Table 4, we observer that the performance of the average classifier is significantly more stable than the last classifier generated by the budget online learning. However, compared to the results in Table 3, the average classifier is significantly denser in the number of support vectors than the solution returned by OSKL, making it less efficient in testing. In general, the number of support vectors increases as the budget size decreases. That is because using a smaller budget size, the online learning algorithm tends to make more mistakes.

6. Conclusion

In this paper, we developed an algorithm for online sparse kernel learning. The key idea is to reduce the number of support vectors by performing stochastic updating. By setting the sampling probability to be proportional to the derivative of a smooth loss function, we are able to show theoretically that the sparsity bound achieved by the proposed algorithm is near optimal. Experimental results show that the proposed algorithm is very effective in finding a both accurate and sparse classifier, and thus reduces the computational cost dramatically.

In the future, we plan to combine the strength of the proposed approach with methods for budget online learning to further improve the sparsity of online kernel learning.

Acknowledgments

This work is partially supported by Office of Navy Research (ONR Award N00014-09-1-0663 and N000141210431), National Basic Research Program of China (973 Program) under Grant 2009CB320801, and National Natural Science Foundation of China (Grant No: 61125203).

[3]Due to the space limit, we didn't report the standard deviation of the number of support vectors, as it can be inferred from that of the sparsity.

Table 3. Classification results (mean ± std) on several benchmark data sets

Data set	Metric	Baseline	Margin	Auxiliary	Pegasos	OSKL			
						$G = 1$	$G = 2$	$G = 4$	$G = 10$
Magic	AC (%)	85.8±0.4	85.7±0.5	85.7±0.3	84.0±0.7	85.5±0.5	85.0±0.8	84.0±0.5	82.3±0.3
	SP (%)	0±0	41.4±0.4	59.8±0.6	70.2±0.2	77.4±0.2	88.0±0.3	93.7±0.2	97.3±0.1
	# SVs	15,216	8,923	6,113	4,537	3,443	1,821	962	414
	Time (s)	11.8±0.0	9.7±0.3	7.0±0.2	5.8±0.1	4.4±0.0	2.8±0.0	2.0±0.0	1.3±0.0
Adult	AC (%)	85.3±0.0	85.2±0.1	85.2±0.0	84.4±0.2	85.2±0.1	85.0±0.1	84.8±0.1	84.1±0.2
	SP (%)	0±0	42.2±0.3	63.7±0.2	68.4±0.1	78.8±0.1	89.2±0.2	94.5±0.1	97.6±0.1
	# SVs	32,561	18,824	11,824	10,291	6,897	3,525	1,801	773
	Time (s)	597±14	371±1	239±3	201±7	138±1	64.8±0.9	25.4±1.2	7.1±0.2
Covtype	AC (%)	79.2±0.1	79.1±0.1	79.1±0.1	78.2±0.2	79.0±0.1	78.6±0.1	77.9±0.2	77.1±0.1
	SP (%)	0±0	38.9±0.1	52.6±0.1	52.7±0.1	69.6±0.0	84.4±0.1	91.9±0.0	96.6±0.0
	# SVs	464,809	284,006	220,209	219,871	141,317	72,677	37,509	15,696
	Time (m)	761±47	579±31	452±30	462±101	284±20	147±9	78.9±5.5	34.0±2.5

Table 4. Classification results (mean ± std) of two budget learning algorithms, where the *last* classifier is used.

Data set	Metric	Forgetron				BOGD++			
Magic	Budget	3,443	1,821	962	414	3,443	1,821	962	414
	AC (%)	78.2±3.9	76.7±5.4	76.3±1.4	75.5±3.5	83.4±1.3	82.4±1.1	82.4±1.5	80.8±1.6
	Time (s)	4.8±0.0	4.4±0.1	3.1±0.1	2.1±0.0	7.6±0.1	5.7±0.0	3.9±0.1	2.2±0.0
Adult	Budget	6,897	3,525	1,801	773	6,897	3,525	1,801	773
	AC (%)	80.4±2.4	76.4±6.4	76.9±3.2	65.7±20.2	83.4±0.4	81.4±2.3	82.9±1	82.6±1.5
	Time (s)	147±6	118±7	61.1±1.6	15.0±1.7	189±14	120±11	54.4±5.5	12.4±0.2
Covtype	Budget	141,317	72,677	37,509	15,696	141,317	72,677	37,509	15,696
	AC (%)	70.8±2.0	64.8±5.6	67.4±5.7	62.6±6.6	74.5±1.9	74.1±4.0	71.5±3.0	73.3±2.0
	Time (m)	327±42	277±21	172±18	80.8±3.4	465±35	325±30	188±34	76.1±7.6

Table 5. Classification results (mean ± std) of two budget learning algorithms, where the *average* classifier is used.

Data set	Metric	Forgetron				BOGD++			
Magic	AC (%)	85.1±0.6	84.0±0.4	82.9±0.5	81.5±0.7	85.9±0.5	85.3±0.5	84.7±0.5	83.4±0.3
	SP (%)	76.0±0.2	74.6±0.3	72.9±0.2	70.3±0.2	68.6±0.2	68.1±0.2	66.3±0.3	63.3±0.4
	# SVs	3,653	3,864	4,129	4,515	4,779	4,857	5,135	5,584
Adult	AC (%)	85.2±0.1	84.8±0.1	84.2±0.1	83.4±0.1	85.2±0.1	85.2±0.1	85.0±0.1	84.7±0.1
	SP (%)	78.1±0.2	77.0±0.2	75.6±0.2	73.4±0.2	70.0±0.1	70.1±0.2	69.4±0.2	67.6±0.1
	# SVs	7,138	7,485	7,943	8,661	9,754	9,731	9,957	10,556
Covtype	AC (%)	78.9±0.1	78.4±0.1	77.9±0.1	77.0±0.2	78.8±0.1	78.6±0.1	78.4±0.1	77.8±0.1
	SP (%)	69.1±0.1	68.2±0.0	67.1±0.1	65.4±0.1	53.7±0.1	54.5±0.1	56.3±0.1	54.9±0.1
	# SVs	143,697	147,650	152,556	160,927	215,119	211,475	203,114	209,766

References

Bartlett, P.L., Bousquet, O., and Mendelson, S. Local rademacher complexities. *Ann. Stat.*, 33(4):1497–1537, 2005.

Burges, C.J.C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

Burges, C.J.C. and Schölkopf, B. Improving the accuracy and speed of support vector learning machines. In *NIPS 9*, pp. 375–381, 1997.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. Tracking the best hyperplane with a simple budget perceptron. *Mach. Learn.*, 69(2-3):143–167, 2007.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. On the generalization ability of on-line learning algorithms. *IEEE Trans. Inf. Theory*, 50(9):2050–2057, 2004.

Chang, C. and Lin, C. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, 2011.

Cheng, L., Vishwanathan, S.V.N., Schuurmans, D., Wang, S., and Caelli, T. Implicit online learning with kernels. In *NIPS 19*, pp. 249–256, 2007.

Cotter, A., Shalev-Shwartz, S., and Srebro, N. Learning optimally sparse support vector machines. In *ICML*, 2013.

Crammer, K., Kandola, J., and Singer, Y. Online classification on a budget. In *NIPS 16*, pp. 225–232, 2004.

Dekel, O., Shalev-Shwartz, S., and Singer, Y. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008.

Duchi, J. and Singer, Y. Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934, 2009.

Frank, A. and Asuncion, A. UCI machine learning repository, 2010.

Freund, Y. and Schapire, R.E. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, 1999.

Keerthi, S.S., Chapelle, O., and DeCoste, D. Building support vector machines with reduced classifier complexity. *J. Mach. Learn. Res.*, 7:1493–1515, 2006.

Kivinen, J., Smola, A.J., and Williamson, R.C. Online learning with kernels. In *NIPS 14*, pp. 785–792, 2002.

Langford, J., Li, L., and Zhang, T. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10: 777–801, 2009.

Lee, Y. and Mangasarian, O.L. Rsvm: Reduced support vector machines. In *SDM*, 2001.

Mallapragada, P.K., Jin, R., Jain, A.K., and Liu, Y. Semiboost: Boosting for semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11): 2000–2014, 2009.

Orabona, F., Keshet, J., and Caputo, B. The projectron: a bounded kernel-based perceptron. In *ICML*, pp. 720–727, 2008.

Roth, V. Probabilistic discriminative kernel classifiers for multi-class problems. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pp. 246–253, 2001.

Schölkopf, B. and Smola, A.J. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. Pegasos: primal estimated sub-gradient solver for SVM. In *ICML*, pp. 807–814, 2007.

Srebro, N., Sridharan, K., and Tewari, A. Smoothness, low noise and fast rates. In *NIPS 23*, pp. 2199–2207, 2010.

Wang, Z., Crammer, K., and Vucetic, S. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *J. Mach. Learn. Res.*, 13:3103–3131, 2012.

Wu, M., Schölkopf, B., and Bakır, G. A direct method for building sparse kernel learning algorithms. *J. Mach. Learn. Res.*, 7:603–624, 2006.

Zhang, L., Jin, R., Chen, C., Bu, J., and He, X. Efficient online learning for large-scale sparse kernel logistic regression. In *AAAI'12*, pp. 1219–1225, 2012.

Zhao, P., Wang, J., Wu, P., Jin, R., and Hoi, S.C.H. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *ICML*, pp. 169–176, 2012.

Zhu, J. and Hastie, T. Kernel logistic regression and the import vector machine. In *NIPS 13*, pp. 1081–1088, 2001.