Neural Networks 60 (2014) 17–24

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Simple randomized algorithms for online learning with kernels

Wenwu He [a,b,*], James T. Kwok [b]

[a] Department of Mathematics and Physics, Fujian University of Technology, Fuzhou, Fujian 350118, China

[b] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 9 April 2014
Received in revised form 7 July 2014
Accepted 17 July 2014
Available online 28 July 2014

Keywords:
Online learning
Kernel methods
Stochastic strategies
Budget

ABSTRACT

In online learning with kernels, it is vital to control the size (budget) of the support set because of the curse of kernelization. In this paper, we propose two simple and effective stochastic strategies for controlling the budget. Both algorithms have an expected regret that is sublinear in the horizon. Experimental results on a number of benchmark data sets demonstrate encouraging performance in terms of both efficacy and efficiency.

1. Introduction

Online learning is a popular and natural approach for solving real-time and life-long learning problems, where instances arrive sequentially. Online learning is also advantageous in large-scale learning as it is often efficient and highly competitive (Shalev-Shwartz, 2007, 2011). At each iteration t, an online learning algorithm produces a function estimate $f_t \in \mathcal{F}$, and then suffers a loss $\ell_t(f_t)$. Here, we assume that the function space $\mathcal{F}$ is closed and convex, and $\ell_t(\cdot)$ is convex. To evaluate the performance of the algorithm, it is customary to measure its regret $R_T = \sum_{t=1}^{T}(\ell_t(f_t) - \ell_t(f))$, where T is the horizon, w.r.t. a competitor $f \in \mathcal{F}$.

A standard online learning algorithm is the gradient descent (GD) (Zinkevich, 2003), which updates f_t as

$$f_{t+1} = \Pi_{\mathcal{F}}(f_t - \eta g_t). \tag{1}$$

Here, g_t is the gradient (or subgradient) of ℓ_t w.r.t. f_t, $\Pi_{\mathcal{F}}$ is the Euclidean projection onto $\mathcal{F}$, and η is the stepsize. Its regret is $O(\sqrt{T})$, and cannot be improved in general (Abernethy, Bartlett, Rakhlin, & Tewari, 2008). To extend linear models for nonlinear function learning, the kernel trick has been widely used (Kivinen, Smola, & Williamson, 2004; Schölkopf & Smola, 2002). An input instance $\mathbf{x}$ is first mapped to $\phi(\mathbf{x})$ in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, where ϕ is a feature map induced by the kernel $\kappa(\cdot,\cdot)$ of $\mathcal{H}$. The inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ between two instances $\mathbf{x}_i, \mathbf{x}_j$ in the linear algorithm is then replaced by $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

A bottleneck for kernelized online learning is that the set S_t of support vectors (SV's) in f_t keeps expanding as learning proceeds. Subsequently, so are the memory and time costs. It is thus necessary to keep $|S_t|$ under control with the use of budget (Crammer, Kandola, & Singer, 2003; Weston, Bordes, & Bottou, 2005). In recent years, various budget algorithms have been proposed, such as the Projectron and its variants (He & Wu, 2012; Orabona, Keshet, & Caputo, 2009), randomized budget perceptron (RBP) (Cesa-Bianchi & Gentile, 2006; Sutskever, 2009) and the simplified Forgetron (Dekel, Shalev-Shwartz, & Singer, 2008; Sutskever, 2009). These algorithms are mainly for classification and only mistake bounds are provided. A framework for algorithms with sublinear regret (as in (1)) is lacking and highly desirable in the budget online learning literature. With such a regret guarantee, we can directly generalize existing budget algorithms to regression and other problems. Moreover, though some budget algorithms (such as the Projectron) have remarkable classification accuracies, they have to update the inverse of a Gram matrix in each iteration. This costs $O(B^2)$ time and memory, where B is the budget. Besides, setting an appropriate budget for a particular learning problem can be difficult.

In this paper, we propose a simple but effective stochastic strategy for online learning with budget. The idea is to keep the excess regret of the budget algorithm over its non-budget counterpart small, while still controlling the growth of the budget. In particular, two algorithms, both with $O(B)$ memory and time, will be presented.

* Corresponding author at: Department of Mathematics and Physics, Fujian University of Technology, Fuzhou, Fujian 350118, China. Tel.: +86 13290930180.
E-mail addresses: hwwhbb@163.com, hwwhbb@gmail.com (W. He), jamesk@cse.ust.hk (J.T. Kwok).

http://dx.doi.org/10.1016/j.neunet.2014.07.006
0893-6080/

Algorithm 1 Online learning with kernels (OLK).

Input: Learning rate sequence $\{\eta_t > 0\}$.
Initialize: $S_1 = \emptyset, f_1 = 0$.
for $t = 1, 2, \ldots, T$ **do**
  receive input $\mathbf{x}_t$;
  suffer loss $\ell_t(f_t)$;
  compute the subgradient $g_t \in \partial \ell_t(f_t)$;
  if $g_t = \mathbf{0}$ **then**
    $f_{t+\frac{1}{2}} = f_t$;
    $S_{t+1} = S_t$;
  else
    $f_{t+\frac{1}{2}} = f_t - \eta_t g_t$;
    $S_{t+1} = S_t \cup \{t\}$;
  end if
  $f_{t+1} = \Pi_{\mathcal{F}}(f_{t+\frac{1}{2}})$.
end for

- The first one is based on dynamically increasing the budget in a stochastic manner. This yields a sublinear expected regret of $O(T^{\frac{1+\gamma}{2}})$ (where $0 < \gamma < 1$), and a budget of (variable) size $O(T^{1-\gamma})$ which is also sublinear in T.
- The second algorithm allows the use of a fixed budget, which may be more convenient in some applications. When the pre-assigned budget is exceeded, the algorithm randomly removes an existing SV. This also yields a sublinear expected regret of $O(\sqrt{T})$ and a budget of size $O(\sqrt{T})$.

The rest of this paper is organized as follows. Section 2 presents a baseline algorithm for non-budget kernelized online learning. The stochastic strategy for (kernelized) online learning with budget, with two specific budget algorithms, is demonstrated in Section 3. A discussion on the related work is in Section 4. Experimental results are reported in Section 5, and the last section gives some concluding remarks.

2. Basic algorithm for Online Learning with Kernels

Given a sequence $\{(\mathbf{x}_t, y_t)\}$, with $\mathbf{x}_t$ coming from the input domain $\mathcal{X} \subseteq \mathbb{R}^d$, y_t from output domain $\mathcal{Y} \subseteq \mathbb{R}$, and $t \in [T] \equiv [1, 2, \ldots, T]$, the learner aims to learn the underlying function $f : \mathcal{X} \to \mathcal{Y}$ in an online manner. Let $\mathcal{H}$ be the RKHS associated with the kernel function κ, such that $\kappa_{ij} \equiv \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi_i, \phi_j \rangle$ and $\phi_t \equiv \phi(\mathbf{x}_t) \in \mathcal{H}$. We assume that

$$\|\phi_t\| \leq 1, \quad \forall t \in [T]. \tag{2}$$

Unless explicitly stated, all the norms are ℓ_2-norms. Moreover, let $\mathbf{K}$ be the Gram matrix with elements κ_{ij}'s,

$$\mathcal{F} = \{f \in \mathcal{H} \mid \|f\| \leq U\} \tag{3}$$

for some given $U > 0$, and $f_t \in \mathcal{F}$ be the function learned at iteration t. Denote the loss of f_t at iteration t by $\ell_t(f_t)$, and $g_t \in \partial \ell_t(f_t)$ be its gradient (or subgradient). For any $f \in \mathcal{H}$, the operation

$$\Pi_{\mathcal{F}}(f) = \arg\min_{g \in \mathcal{F}} \|g - f\| = \min\left\{1, \frac{U}{\|f\|}\right\} f \tag{4}$$

projects f onto $\mathcal{F}$.

Algorithm 1 shows a basic kernel extension of GD in (1), and will be called *Online Learning with Kernels* (OLK) in the sequel. Let $\{f_t \in \mathcal{F}\}_{t \in [T]}$ be a sequence obtained by OLK. The following regret can be easily obtained from (Zinkevich, 2003).

Theorem 1. (i) *Using a constant stepsize $\eta > 0$, the regret of OLK is bounded as*

$$R_T \leq \frac{\|f_1 - f\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|g_t\|^2. \tag{5}$$

Assume that $\max_{t \in [T]} \|g_t\|^2 \leq G$. We have $R_T \leq 2U\sqrt{GT}$ on setting $\eta = 2UG^{-\frac{1}{2}} T^{-\frac{1}{2}}$.

(ii) *With a dynamic stepsize*

$$\eta_t = \eta t^{-\frac{1}{2}}, \tag{6}$$

we have

$$R_T \leq \frac{\sqrt{T} \max_{t \in [T]} \|f_t - f\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \frac{\|g_t\|^2}{\sqrt{t}}. \tag{7}$$

On setting $\eta = \sqrt{2}UG^{-\frac{1}{2}}$, we have $R_T \leq \sqrt{2}\left(2U\sqrt{GT}\right)$.

The constant stepsize η can be difficult to set unless T is known. The OGD algorithm (Kivinen et al., 2004; Zhao, Wang, Wu, Jin, & Hoi, 2012), which uses a constant stepsize, also suffers from the same problem. Moreover, OGD has a regret of

$$R_T \leq \frac{\lambda T + \eta^{-1}}{2} \|f_1 - f\|^2 + \eta \sum_{t=1}^{T} \|g_t\|^2,$$

where $\lambda > 0$ is a regularization parameter. It is worse than the bound in (5) by a factor of $\frac{1}{2}$ on setting $\lambda T = \eta^{-1}$ (Zhao et al., 2012).

On the other hand, with the dynamic stepsize scheme in (6), η_t does not depend on T, and the learner can tune η for optimal performance based on a subset of samples. The price to pay is that its regret is only $\sqrt{2}$-competitive with that of the fixed stepsize.

We assume that at iteration t, the change which may be added to f_t can be written as $\alpha_t \phi_t$ for some $\alpha_t \in \mathbb{R}$, i.e.,

$$-\eta_t g_t = \alpha_t \phi_t. \tag{8}$$

This holds for many commonly used loss functions. For example, for the hinge loss $\ell_t(f_t) = \max\{0, 1 - y_t f_t(\phi_t)\}$, we have $g_t = -y_t \phi_t$ when $\ell_t(f_t) > 0$. Similarly, for the square loss $\ell_t(f_t) = \frac{1}{2}(y_t - f_t(\phi_t))^2$, we have $g_t = -(y_t - f_t(\phi_t))\phi_t$. Hence, f_t can be written as $f_t = \sum_{\tau \in S_t} \alpha_\tau \phi_\tau$. On non-separable problems or noisy data, the number of support vectors involved in f_t may thus increase with t, and both the update and prediction time will become unlimited. This hinders the application of online learning with kernels to large-scale problems.

When a new SV is to be added, $f_{t+\frac{1}{2}} = \sum_{\tau \in S_t} \alpha_\tau \phi_\tau + \alpha_t \phi_t$. Note that the projection $\Pi_{\mathcal{F}}(f_{t+\frac{1}{2}})$ (defined in (4)) involves computing $\|f_{t+\frac{1}{2}}\|^2$, which can be easily obtained as:

$$\begin{aligned} \left\|f_{t+\frac{1}{2}}\right\|^2 &= \|f_t\|^2 + \alpha_t^2 \kappa_{tt} + 2\alpha_t \sum_{\tau \in S_t} \alpha_\tau \kappa_{\tau t} \\ &= \|f_t\|^2 + \alpha_t^2 \kappa_{tt} + 2\alpha_t f_t(\mathbf{x}_t). \end{aligned} \tag{9}$$

Here, $f_t(\mathbf{x}_t)$ is the prediction at iteration t and needs to be computed anyway. By storing $\|f_t\|^2$ in each iteration, computing (9) is inexpensive in terms of both time and memory.

3. Randomized strategies for online learning with budget

In online learning with budget, we restrict each f_t to have a maximum of B SV's, where $B > 0$. The budget version of OLK can be obtained by replacing $f_{t+\frac{1}{2}}$ in Theorem 1 with $f^B_{t+\frac{1}{2}}$, whose expression is to be specified.

Proposition 1. *For any stepsize sequence $\{\eta_t\}$ (with $\eta_t > 0$), the regret for the budget version of OLK is bounded by*

$$R^B_T \leq R^{\sim B}_T + \sum_{t=1}^{T} \frac{1}{2\eta_t} \left(\|e_t\|^2 + 2\left\langle f_{t+\frac{1}{2}} - f, e_t \right\rangle\right), \tag{10}$$

where $e_t = f^B_{t+\frac{1}{2}} - f_{t+\frac{1}{2}}$, and $R^{\sim B}_T = \sum_{t=1}^{T} \frac{\|f_t - f\|^2 - \|f_{t+1} - f\|^2}{2\eta_t} + \frac{\eta_t \|g_t\|^2}{2}$.

Proof. By convexity, $\ell_t(f_t) - \ell_t(f) \le \langle f_t - f, g_t\rangle$. We base our analysis on the reduction $\ell_t(f) \mapsto \langle g_t, f\rangle$. Recall that $\|\Pi_{\mathcal{F}}(h) - f\| \le \|h - f\|$ for all $h, f \in \mathcal{F}$. From Algorithm 1, we have

$$\begin{aligned}\|f_{t+1} - f\|^2 &\le \left\|f_{t+\frac{1}{2}} - f\right\|^2 \\ &= \|f_t - \eta_t g_t - f\|^2 \\ &= \|f_t - f\|^2 + \|\eta_t g_t\|^2 - 2\eta_t\langle f_t - f, g_t\rangle.\end{aligned}$$

Consequently,

$$\begin{aligned}R_T &\le \sum_{t=1}^{T}\langle f_t - f, g_t\rangle \\ &\le \sum_{t=1}^{T}\left[\frac{1}{2\eta_t}(\|f_t - f\|^2 - \|f_{t+1} - f\|^2) + \frac{\eta_t}{2}\|g_t\|^2\right] = R_T^{\sim B}.\end{aligned}$$

Similarly, for the budget version, $\|f_{t+1} - f\|^2 \le \|f^B_{t+\frac{1}{2}} - f\|^2$. Using the fact that

$$\begin{aligned}\left\|f^B_{t+\frac{1}{2}} - f\right\|^2 = \left\|f_{t+\frac{1}{2}} - f\right\|^2 + \left\|f^B_{t+\frac{1}{2}} - f_{t+\frac{1}{2}}\right\|^2 \\ + 2\left\langle f_{t+\frac{1}{2}} - f, f^B_{t+\frac{1}{2}} - f_{t+\frac{1}{2}}\right\rangle,\end{aligned}$$

we have

$$\begin{aligned}\langle f_t - f, g_t\rangle \le \frac{1}{2\eta_t}\left(\|f_t - f\|^2 - \|f_{t+1} - f\|^2 + \|\eta_t g_t\|^2\right) \\ + \frac{1}{2\eta_t}\left(\|e_t\|^2 + 2\left\langle f_{t+\frac{1}{2}} - f, e_t\right\rangle\right),\end{aligned}$$

which leads to (10). □

Note that $R_T^{\sim B}$ is upper-bounded by (5) for fixed stepsize, and (7) for dynamic stepsize. Hence, the budget algorithm's regret exceeds its non-budget counterpart by a summation term. To ensure that R_T^B is still sublinear in T (e.g., as $O(\sqrt{T})$ in Theorem 1), we have to keep this sum at least within the same order. Apparently, this is possible if e_t is small. However, a direct analysis of e_t is difficult without extra information on f. In the following, we propose a stochastic strategy which maintains

$$\mathbf{E}[e_t] = 0. \tag{11}$$

In other words, $f^B_{t+\frac{1}{2}}$ is an unbiased estimator of $f_{t+\frac{1}{2}}$. The expectation of the regret in (10) then reduces to

$$\mathbf{E}[R_T^B] \le R_T^{\sim B} + \sum_{t=1}^{T}\frac{1}{2\eta_t}\mathbf{E}[\|e_t\|^2]. \tag{12}$$

The issue becomes how to control $\sum_{t=1}^{T}\frac{1}{2\eta_t}\mathbf{E}[\|e_t\|^2]$ with a proper budget B.

3.1. Online Learning with Random Updating (OLRU)

In this section, we propose a simple but effective algorithm called *Online Learning with Random Updating* (OLRU), shown in Algorithm 2. Recall that at the tth iteration of OLK, whenever $g_t \neq 0$, a new component $\alpha_t\phi_t = -\eta_t g_t$ is added to f_t to form $f_{t+\frac{1}{2}}$. For learning with budget, we set

$$f^B_{t+\frac{1}{2}} = f_t + \widetilde{\alpha}_t\phi_t,$$

where $\widetilde{\alpha}_t \equiv (1-\theta_t)a_t$ for some a_t to be determined, and $\theta_t \in \{0, 1\}$ is a random variable that takes the value 0 with probability p_t. When $\theta_t = 0$, a new SV is added; whereas when $\theta_t = 1$ (with probability $1 - p_t$), no SV is added and $f^B_{t+\frac{1}{2}} = f_t$. To determine a_t,

Algorithm 2 Online learning with random updating (OLRU).

Input: Learning rate sequence $\{\eta_t > 0\}$.
Initialize: $S_1 = \emptyset, f_1^B = 0$.
for $t = 1, 2, \ldots, T$ **do**
  receive input $\mathbf{x}_t$; suffer loss $\ell_t(f_t)$ and compute its subgradient g_t;
  set the indicator θ_t using probability p_t in (13);
  if $g_t = 0$ or $\theta_t = 1$ **then**
    $f^B_{t+\frac{1}{2}} = f_t, S_{t+1} = S_t$;
  else
    $\alpha_t\phi_t = -\eta_t g_t, a_t = \frac{\alpha_t}{p_t}$;
    $f^B_{t+\frac{1}{2}} = f_t + a_t\phi_t, S_{t+1} = S_t \cup \{t\}$;
  end if
  $f_{t+1} = \Pi_{\mathcal{F}}(f^B_{t+\frac{1}{2}})$.
end for

recall that we try to maintain $\mathbf{E}[e_t] = 0$. Here, $e_t = f^B_{t+\frac{1}{2}} - f_{t+\frac{1}{2}} = (\widetilde{\alpha}_t - \alpha_t)\phi_t$. $\mathbf{E}[e_t] = 0$ thus implies $\mathbf{E}_t[\widetilde{\alpha}_t] = \alpha_t$, where $\mathbf{E}_t[\cdot]$ is the shorthand for $\mathbf{E}[\cdot|p_t]$. Hence,

$$a_t = \frac{\alpha_t}{p_t}.$$

Let $B_T = \sum_{t=1}^{T}(1-\theta_t)$ be an upper bound on the total number of SVs added in the T iterations (this is an upper bound because an SV is added at iteration t only when $g_t \neq 0$ is also satisfied). Thus, $\mathbf{E}[B_T] = \sum_{t=1}^{T} p_t$. The goal is to control $\sum_{t=1}^{T}\frac{1}{2\eta_t}\mathbf{E}_t[\|e_t\|^2]$ in (12) with a reasonable value of $\mathbf{E}[B_T]$.

Consider setting

$$p_t = \begin{cases} cT^{-\gamma} & \text{when a constant stepsize } \eta_t = \eta \text{ is used} \\ ct^{-\gamma} & \text{when a dynamic stepsize } \eta_t = \eta t^{-\frac{1+\gamma}{2}} \text{ is used,}\end{cases} \tag{13}$$

where $\gamma \in (0, 1)$ and c is a constant. In both cases, $\mathbf{E}[B_T] = O(T^{1-\gamma})$. As for $\sum_{t=1}^{T}\frac{1}{2\eta_t}\mathbf{E}_t[\|e_t\|^2]$, it is easy to see that

$$\mathbf{E}_t[\|e_t\|^2] = \mathbf{E}_t[\widetilde{\alpha}_t^2]\kappa_{tt} - \alpha_t^2\kappa_{tt} = \frac{(1-p_t)\alpha_t^2\kappa_{tt}}{p_t}.$$

Thus, on using (8),

$$\begin{aligned}\sum_{t=1}^{T}\frac{1}{2\eta_t}\mathbf{E}_t[\|e_t\|^2] &= \sum_{t=1}^{T}\frac{(1-p_t)\alpha_t^2\kappa_{tt}}{2\eta_t p_t} \\ &= \sum_{t=1}^{T}\frac{(1-p_t)\eta_t\|g_t\|^2}{2p_t} \\ &= \frac{1}{2}\sum_{t=1}^{T}\frac{\eta_t\|g_t\|^2}{p_t} - \frac{1}{2}\sum_{t=1}^{T}\eta_t\|g_t\|^2.\end{aligned}$$

Combining this with (5) and (12), we obtain the following regret bound.

Theorem 2. *Let $b \equiv \frac{1+\gamma}{2}$, and use the update probability p_t in (13).*

(i) *The expected regret of OLRU (with a constant stepsize $\eta > 0$) can be bounded as*

$$\mathbf{E}[R_T^B] \le \frac{\|f_1 - f\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\frac{\|g_t\|^2}{p_t}.$$

Assume that

$$\max_{t\in[T]}\|g_t\|^2 \le G. \tag{14}$$

We obtain $\mathbf{E}[R_T^B] \le 2UG^{\frac{1}{2}}c^{-\frac{1}{2}}T^b$ on setting $\eta = 2Uc^{\frac{1}{2}}G^{-\frac{1}{2}}T^{-b}$.

(ii) *With a dynamic stepsize* $\eta_t = \eta t^{-b}$, *we have*

$$\mathbf{E}[R_T^B] \le \frac{T^b \max_{t\in[T]} \|f_t - f\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \frac{\|g_t\|^2}{t^b p_t}.$$

Using $\eta = 2U(cb)^{\frac{1}{2}} G^{-\frac{1}{2}}$, *we obtain* $\mathbf{E}[R_T^B] \le 2UG^{\frac{1}{2}}(cb)^{-\frac{1}{2}} T^b$.

Thus, for both stepsize schemes, OLRU has an expected regret of $O(T^{\frac{1+\gamma}{2}})$ and a support set of size $O(T^{1-\gamma})$. Note that γ trades off the regret and support set size. As in OLK, the use of a dynamic stepsize frees η from T and allows it to be tuned on a sample subset without knowing T in advance, while its expected regret is only $\sqrt{\frac{2}{1+\gamma}}$-competitive with that for constant stepsize.

Moreover,

$$\left\|f_{t+\frac{1}{2}}^B\right\|^2 = \|f_t\|^2 + a_t^2\kappa_{tt} + 2a_t f_t(\mathbf{x}_t),$$

which is very similar to (9) for OLK. Thus, the projection $\Pi_{\mathcal{F}}(f_{t+\frac{1}{2}}^B)$ in step 12 can also be efficiently computed.

As will be empirically demonstrated in Section 5, this simple algorithm is effective. The addition of new SVs is sparse (B_T times out of T iterations), and both the memory cost and regret are sublinear with T. However, it does not utilize $|\alpha_t|$ in designing p_t, and doing so may lead to improved performance.

3.2. Online Learning with Random Discarding (OLRD)

In OLRU, though $\mathbf{E}[B_T]$ is sublinear with T, it is not fixed. In some applications, it may be more convenient to directly specify a fixed budget B. In this section, we propose an algorithm called *Online Learning with Random Discarding* (OLRD) (Algorithm 3). It is based on the idea that when $|S_t| > B$ at iteration t, one SV will be randomly selected and discarded.

Specifically, in forming $f_{t+\frac{1}{2}}^B$, consider the case where the budget is exceeded on adding a new component $-\eta_t g_t$ to $f_t = \sum_{i\in[B]} \alpha_i^t \phi_i^t$, where $\phi_i^t \equiv \phi(\mathbf{x}_i^t)$. In OLRD, one SV will be selected and discarded. Let $\theta_i^t \in \{0, 1\}$ be the random variable such that $\theta_i^t = 1$ when the ith SV is selected (with probability q_i^t); and $\theta_i^t = 0$ otherwise. Clearly, $\sum_{i=1}^B \theta_i^t = 1$. We then update $f_{t+\frac{1}{2}}^B$ as

$$f_{t+\frac{1}{2}}^B = f_t^{B-1} - \eta_t g_t, \tag{15}$$

where $f_t^{B-1} = \sum_{i\in[B]} \widetilde{\alpha}_i^t \phi_i^t$, $\widetilde{\alpha}_i^t = (1-\theta_i^t)a_i^t$, and a_i^t's are coefficients to be determined. Recall that

$$e_t = f_{t+\frac{1}{2}}^B - f_{t+\frac{1}{2}} = (f_t^{B-1} - \eta_t g_t) - (f_t - \eta_t g_t) = f_t^{B-1} - f_t, \tag{16}$$

and (11) requires

$$\begin{aligned}\mathbf{E}[e_t] = 0 &\Leftrightarrow \mathbf{E}[f_t^{B-1}] = f_t \\ &\Leftrightarrow \mathbf{E}_t[\widetilde{\alpha}_i^t] = \alpha_i^t,\end{aligned}$$

where $\mathbf{E}_t$ is the shorthand for $\mathbf{E}[\cdot|\boldsymbol{q}^t]$, and $\boldsymbol{q}^t \equiv (q_1^t, \ldots, q_B^t)$. Solving, we obtain

$$a_i^t = \frac{\alpha_i^t}{1 - q_i^t}. \tag{17}$$

Thus,

$$f_t^{B-1} = \sum_{i\in[B]\setminus i^*} a_i^t \phi_i^t,$$

where $i^* = \{i \in [B] \mid \theta_i^t = 1\}$ contains indices of the removed SVs.

Algorithm 3 Online learning with random discarding (OLRD).

1: **Input:** Learning rate sequence $\{\eta_t > 0\}$; $c > 0$; B_t set in (23) when $\eta_t = \eta$, and B_t set in (24) when $\eta_t = \eta t^{-\frac{1}{2}}$.
2: **Initialize:** $S_1 = \emptyset, f_1^B = 0$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: receive input $\mathbf{x}_t$; suffer loss $\ell_t(f_t)$ and compute its subgradient g_t;
5: **if** $g_t = 0$ **then**
6: $f_{t+\frac{1}{2}}^B = f_t, S_{t+1} = S_t$;
7: **else if** $|S_t| < B_t$ **then**
8: $f_{t+\frac{1}{2}}^B = f_t - \eta_t g_t, S_{t+1} = S_t \cup \{t\}$;
9: **else**
10: set the indicator $\boldsymbol{\theta}^t$ using $(q_1^t, \ldots, q_B^t)$ in (19);
11: $i^* = \{i \mid \theta_i^t = 1, i \in [B_t]\}$;
12: $f_t^{B-1} = \sum_{i\in[B_t]\setminus i^*} \frac{\alpha_i^t}{1-q_i^t}\phi_i^t$;
13: $f_{t+\frac{1}{2}}^B = f_t^{B-1} - \eta_t g_t, S_{t+1} = (S_t \setminus \{i^*\text{th element in } S_t\}) \cup \{t\}$;
14: **end if**
15: $f_{t+1} = \Pi_{\mathcal{F}}(f_{t+\frac{1}{2}}^B)$.
16: **end for**

As in Section 3.1, we have to bound the term $\sum_{t=1}^T \frac{1}{2\eta_t}\mathbf{E}_t[\|e_t\|^2]$ in (12). Now, from (16), (17),

$$\begin{aligned}\mathbf{E}_t[\|e_t\|^2] &= \mathbf{E}_t[\|f_t^{B-1} - f_t\|^2] \\ &= \mathbf{E}_t\left[\left\|\sum_{i\in[B]}(q_i^t - \theta_i^t)a_i^t\phi_i^t\right\|^2\right] \\ &= \sum_{i,j\in[B]} a_i^t a_j^t \kappa_{ij}\left(\mathbf{E}_t[\theta_i^t\theta_j^t] - q_i^t q_j^t\right) \\ &= \sum_{i\in[B]}(a_i^t)^2 q_i^t \kappa_{ii} - \left\|\sum_{i\in[B]} a_i^t q_i^t \phi_i^t\right\|^2 \\ &\le \sum_{i\in[B]}(a_i^t)^2 q_i^t \kappa_{ii} \\ &= \sum_{i\in[B]} \frac{q_i^t(\alpha_i^t)^2\kappa_{ii}}{(1-q_i^t)^2}.\end{aligned} \tag{18}$$

Thus, the issue becomes how to minimize (18) with an appropriate $\mathbf{q}^t$.

In the following, we consider using the simple uniform distribution,

$$q_i^t = \frac{1}{B}, \quad i = 1, 2, \ldots, B. \tag{19}$$

In other words, every SV has the same probability of being discarded. Using the assumption in (2), (18) then reduces to

$$\mathbf{E}_t[\|e_t\|^2] \le \frac{B\sum_{i\in[B]}\|\alpha_i^t\phi_i^t\|^2}{(B-1)^2}. \tag{20}$$

However, the fact that $f_t \in \mathcal{F}$ only yields a bound on $\|f_t\|^2 = \|\sum_{i\in[B]}\alpha_i^t\phi_i^t\|^2$, but not on $\sum_{i\in[B]}\|\alpha_i^t\phi_i^t\|^2$ in the RHS of (20). In the following, we will show that this problem can be alleviated by performing an approximate (instead of exact) projection of $f_{t+\frac{1}{2}}^B$ onto $\mathcal{F}$.

In the expansion $f_{t+\frac{1}{2}}^B = \sum_{i=1}^B \alpha_i^{t+\frac{1}{2}}\phi_i^t$, recall from (15) and (17) that $\alpha_i^{t+\frac{1}{2}} = \frac{\alpha_i^t}{1-q_i^t}$ $\forall i \in [B]\setminus i^*$ and $\alpha_B^{t+\frac{1}{2}} = \alpha_t$. Let $\boldsymbol{\alpha}_{t+\frac{1}{2}}^B =$

$[\alpha_1^{t+\frac{1}{2}}, \ldots, \alpha_B^{t+\frac{1}{2}}]'$, and $\mathbf{K}^B_{t+\frac{1}{2}}$ be the kernel submatrix corresponding to SV's in $f^B_{t+\frac{1}{2}}$. We have

$$\begin{aligned}\left\|f^B_{t+\frac{1}{2}}\right\|^2 &= \left(\boldsymbol{\alpha}^B_{t+\frac{1}{2}}\right)' \mathbf{K}^B_{t+\frac{1}{2}} \boldsymbol{\alpha}^B_{t+\frac{1}{2}} \\ &= \mathrm{tr}\left(\mathbf{K}^B_{t+\frac{1}{2}} \boldsymbol{\alpha}^B_{t+\frac{1}{2}} \left(\boldsymbol{\alpha}^B_{t+\frac{1}{2}}\right)'\right) \\ &\le \mathrm{tr}\left(\mathbf{K}^B_{t+\frac{1}{2}}\right) \left\|\boldsymbol{\alpha}^B_{t+\frac{1}{2}}\right\|^2 \\ &\le B \left\|\boldsymbol{\alpha}^B_{t+\frac{1}{2}}\right\|^2 \\ &\le B^2 \left\|\boldsymbol{\alpha}^B_{t+\frac{1}{2}}\right\|^2_\infty, \end{aligned} \tag{21}$$

where the second inequality follows from the assumption in (2), and $\left\|\boldsymbol{\alpha}^B_{t+\frac{1}{2}}\right\|_\infty = \max_i\{|\alpha_i^{t+\frac{1}{2}}|\}$. To obtain f_{t+1}, instead of projecting $f^B_{t+\frac{1}{2}}$ onto $\mathcal{F}$, we require

$$B^2 \|\boldsymbol{\alpha}_{t+1}\|^2_\infty \le U^2, \tag{22}$$

where $\boldsymbol{\alpha}_{t+1} = [\alpha_1^{t+1}, \ldots, \alpha_B^{t+1}]'$ is the vector of expansion coefficients for f_{t+1}. From (21), this can be satisfied by setting

$$\alpha_i^{t+1} = \min\left\{1, \frac{U}{B}\right\} \alpha_i^{t+\frac{1}{2}}, \quad \forall i \in [B].$$

Note that $\|f_{t+1}\|^2 \le B^2\|\boldsymbol{\alpha}_{t+1}\|^2_\infty \le U^2$, and so we still have $f_{t+1} \in \mathcal{F}$.

From (20), we have

$$\mathbf{E}_t[\|e_t\|^2] \le \frac{B \sum_{i\in[B]} \|\alpha_i^t \phi_i^t\|^2}{(B-1)^2} \le \frac{B\|\boldsymbol{\alpha}_t\|^2}{(B-1)^2} \le \frac{B^2\|\boldsymbol{\alpha}_t\|^2_\infty}{(B-1)^2},$$

where $\boldsymbol{\alpha}_t = [\alpha_1^t, \ldots, \alpha_B^t]'$. Combining this with (22), we have

$$\mathbf{E}_t[\|e_t\|^2] \le \frac{U^2}{(B-1)^2}.$$

To keep $\sum_{t=1}^T \frac{1}{2\eta_t}\mathbf{E}_t[\|e_t\|^2]$ in (12) within the order of $O(\sqrt{T})$, we have to set B to be in the same order. Combining this with (5) (or (7)) and (12), we obtain the following regret bound.

Theorem 3. *Using the uniform distribution for $\boldsymbol{q}^t$,*

(i) *the expected regret of OLRD, using a constant stepsize $\eta > 0$ and fixed budget*

$$B = 1 + cT^{\frac{1}{2}} \tag{23}$$

for some $c > 0$, can be bounded as

$$\mathbf{E}[R_T^B] \le \frac{1}{2\eta}\left(\|f_1 - f\|^2 + \frac{TU^2}{(B-1)^2}\right) + \frac{\eta}{2}\sum_{t=1}^T \|g_t\|^2.$$

Assume that $\max_{t\in[T]} \|g_t\|^2 \le G$. We obtain $\mathbf{E}[R_T^B] \le bU\sqrt{GT}$ on setting $\eta = bUG^{-\frac{1}{2}}T^{-\frac{1}{2}}$, where $b = \sqrt{4 + \frac{1}{c^2}}$.

(ii) *With a dynamic stepsize $\eta_t = \eta t^{-\frac{1}{2}}$ and a dynamic budget*

$$B_t = 1 + ct^{\frac{1}{2}} \tag{24}$$

for some $c > 0$, we have

$$\mathbf{E}[R_T^B] \le \frac{\sqrt{T}}{2\eta}\left[\max_{t\in[T]} \|f_t - f\|^2 + \frac{2TU^2}{(B_T-1)^2}\right] + \frac{\eta}{2}\sum_{t=1}^T \frac{\|g_t\|^2}{\sqrt{t}}. \tag{25}$$

Setting $\eta = \tilde{b}UG^{-\frac{1}{2}}$ where $\tilde{b} = \sqrt{2 + \frac{1}{c^2}}$, we obtain $\mathbf{E}[R_T^B] \le \tilde{b}(2U\sqrt{GT})$.

3.3. Discussion

As shown in Theorem 3, OLRD also has an expected regret of $O(\sqrt{T})$, which is the same as OLK (Theorem 1). However, in comparison with the regrets there, the regrets in Theorem 3 have an additional term ($\frac{1}{c^2}$ in b, and $\frac{1}{c^2}$ in $\tilde{b}$). Hence, c controls the budget (via (23) or (24)), and also the gap between the regrets of the budget and non-budget algorithms.

To have optimal performance, we have to tune η, which in general depends on T and B. When budget is not used, one can tune the parameters based on a subset of, say, T_0, samples. However, for budget algorithms with a fixed budget B, the η value tuned from these T_0 samples may not be optimal w.r.t. all the T samples. By specifying c (instead of B), Theorem 3 suggests that we can use $\eta = \eta_0 T_0^{-\frac{1}{2}}$, and tune η_0 for optimal performance on the sample subset. Once η_0 is determined, we can use $\eta = \eta_0 T^{-\frac{1}{2}}$ on the T samples.

When T is not known and a fixed B is used, we cannot obtain optimal performance by tuning η (or η_0) on a sample subset, and the gap may grow with t. So we turn to using a dynamic stepsize. On setting $\eta = \sqrt{2 + \frac{t}{(B_t-1)^2}}UG^{-\frac{1}{2}}$, the regret in (25) can be rewritten as

$$\mathbf{E}[R_t^B] \le \sqrt{2 + \frac{t}{(B_t-1)^2}}(2U\sqrt{Gt}).$$

Obviously, for optimal performance, we need to have η free from t. In other words, we have to introduce a c to balance B and T. This leads to a dynamic budget, i.e., $B_t = O(\sqrt{t})$. Although B_t changes with time, its magnitude is predicable with any horizon t, and the expected regret is $O(\sqrt{t})$.

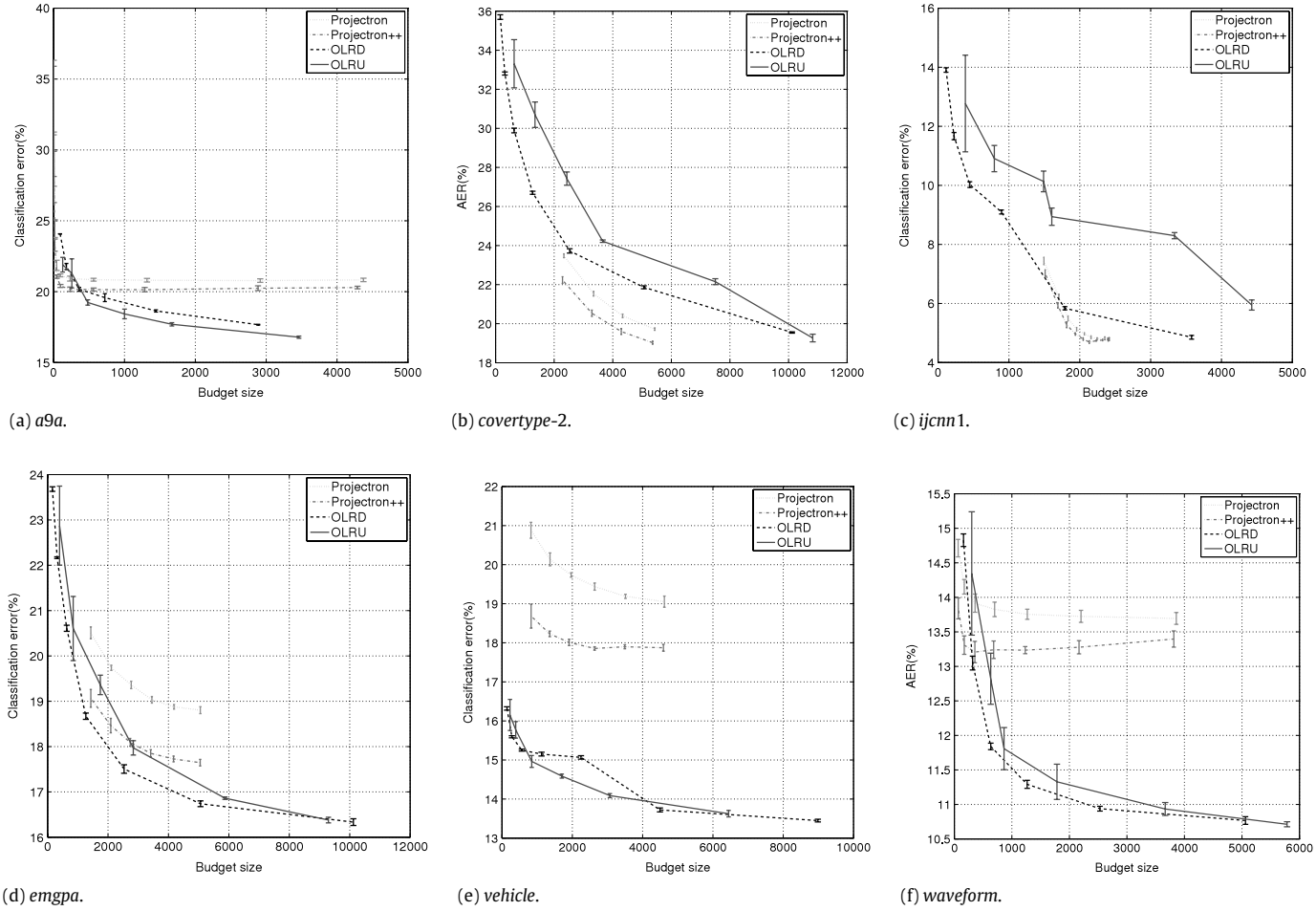
Compared with OLRU, OLRD has an expected regret of $O(\sqrt{T})$ using $O(\sqrt{T})$ SVs. Hence, one may expect OLRD to have better performance when T is large. On the other hand, OLRU is simple, and is particularly useful when updates to the function iterates are expected to be sparse.

4. Related work

A related work is the BOGD algorithm in Zhao et al. (2012). However, (i) BOGD claims to have an expected regret of $O(\sqrt{T})$, which is independent of B. Indeed, their Theorem 3 shows that the regret is $O(\sqrt{T}) + \|f\|^2\sqrt{T}$, where f is the non-budget competitor. In BOGD, $\|f\|^2$ may become $O(T)$, not a constant independent of T. If $\|f\|^2$ and $\|f_t^B\|^2 = O\left(\frac{B^2}{T}\right)$ are required to have the same constant upper bound, we need to have $B \propto \sqrt{T}$. (ii) The parameters used in their BOGD experiments do not match their theoretical analysis. (iii) BOGD is based on the OGD algorithm. As discussed in Section 2, it is worse than the bound in (5) by a factor of $\frac{1}{2}$. (iv) BOGD needs to know the horizon T. (v) Their improved version BOGD++ uses a non-uniform q_i^t distribution utilizing information on $\alpha_i^t\phi_i^t$. Specifically, it sets $1 - q_i^t \propto \|\alpha_i^t\phi_i^t\|$, and $q_i^t = 1 - \frac{(B-1)\|\alpha_i^t\phi_i^t\|}{\sum_{i\in[B]}\|\alpha_i^t\phi_i^t\|}$. However, q_i^t may become negative,[1] and their Theorem 4 will then fail.

In general, the magnitude of $\alpha_i\phi_i$ may be used to design the update/discard probability. In particular, the OSKL algorithm

[1] For example, let $B = 100$, $\|\alpha_i^t\phi_i^t\| = 1$ for $i \in [99]$, and $\|\alpha_{100}^t\phi_{100}^t\| = 1.1$. Then $q_{100}^t = 1 - \frac{108.9}{100.1} < 0$.



(a) *a9a*. (b) *covertype*-2. (c) *ijcnn*1.

(d) *emgpa*. (e) *vehicle*. (f) *waveform*.

Fig. 1. Classification error vs. budget for Projectron, Projectron++ OLRU and OLRD.

Table 1
Data sets used in the experiments.

Data set	#samples	#features
a9a	32,561	123
covertype-2	100,000	54
ijcnn1	49,990	22
emgpa	100,000	8
vehicle	78,823	100
waveform	100,000	21

(Zhang, Yi, Jin, Lin, & He, 2013) considers several special smooth losses and uses their instant derivatives ($|\ell'(y_t, f_t(\mathbf{x}_t))|$) to design the update probability. High probability regret and support set sizes are derived, though only implicitly. However, OSKL is aggressive in that the update probability is completely determined by the scale of the instant derivative and may be sensitive to noise. Besides, OSKL can only insert SV's but not removing them. Thus, it cannot remove noise learned in previous iterations.

Finally, note that RBP also proposed a random method that removes one support vector when the budget is violated (Cesa-Bianchi & Gentile, 2006). It is based on the shifting perceptron, and driven by mistakes. In other words, when $y_t \neq \text{sign}(\langle f_t, \phi(\mathbf{x}_t)\rangle)$, it updates f_t as

$$f_{t+1} = (1 - \lambda_t) f_t + y_t \phi(\mathbf{x}_t),$$

where $\lambda_t > 0$. This can also be viewed as a classifier using the hinge loss but only updates when the loss value is larger than 1. On the other hand, the proposed algorithms are regret-driven, make full use of the loss information, and can be used in applications other than classification.

5. Experiments

In this section, experiments are performed on a number of binary[2] classification data sets from the LIBSVM archive[3] and UCI data repository (Table 1). The hinge loss and Gaussian kernel are used. The kernel parameter, stepsize and value of U in (3) are tuned by running the algorithm on a data subset with 10,000 randomly selected samples.

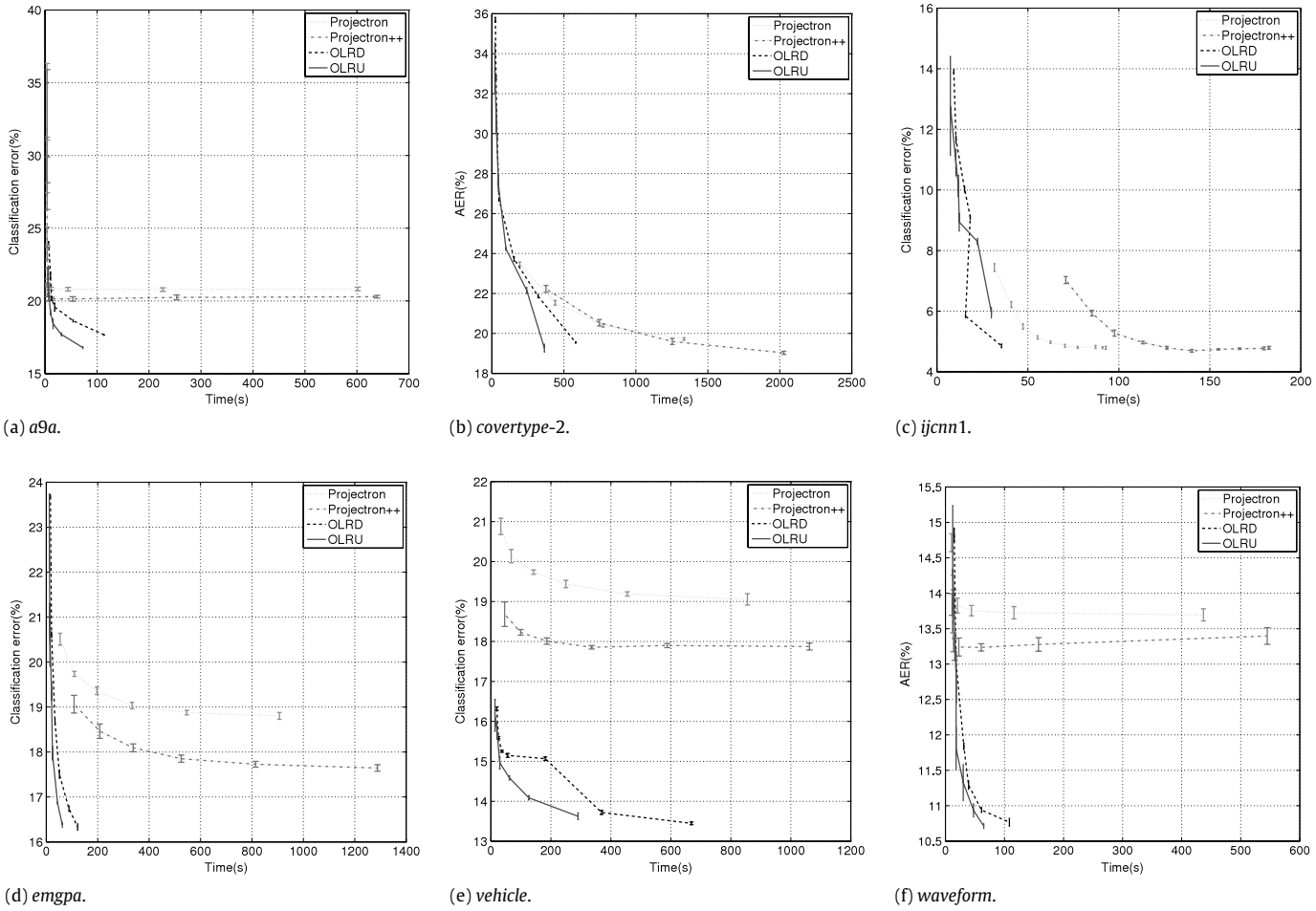
5.1. Algorithms using a fixed budget

In this experiment, we compare the following online learning algorithms that require a pre-defined fixed budget B: (i) RBP (Cesa-Bianchi & Gentile, 2006); (ii) Forgetron (Dekel et al., 2008); and (iii) OLRD (with constant stepsize specified in Theorem 2). Both RBP and Forgetron take $O(B)$ memory and time. The budget is set using (23), with T equal to the number of samples and c varied in $\{2^{-1}, 2^1, 2^3\}$ (leading to three budget sizes). To reduce statistical variability, results are averaged over 10 permuted versions of the data set.

Results on the classification error rates are shown in Table 2. As can be seen, OLRD achieves the lowest error almost all the time. Table 3 shows results on the running time. As can be seen, RBP is slightly faster because it is mistake-driven and no projection

[2] Two of the data sets (*vehicle* and *waveform*) are originally multiclass data sets. They are converted to binary classification by merging some of the classes.

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.



(a) *a9a*. (b) *covertype*-2. (c) *ijcnn*1.

(d) *emgpa*. (e) *vehicle*. (f) *waveform*.

Fig. 2. Classification error (%) vs. time (seconds) for Projectron, Projectron++ OLRU and OLRD.

Table 2
Classification errors (%) of RBP, Forgetron and OLRD on the various data sets. The best results and those that are not significantly worse (according to the t-test with a p-value less than 0.05) are in bold.

Data set	Budget	RBP	Forgetron	OLRD
a9a	91	26.23 ± 0.09	25.88 ± 0.15	$\mathbf{24.07} \pm 0.01$
	362	23.45 ± 0.23	23.00 ± 0.17	$\mathbf{20.16} \pm 0.13$
	1445	21.71 ± 0.27	21.70 ± 0.05	$\mathbf{18.64} \pm 0.07$
covertype-2	159	38.18 ± 0.16	38.11 ± 0.07	$\mathbf{35.69} \pm 0.12$
	633	32.58 ± 0.08	32.67 ± 0.06	$\mathbf{29.89} \pm 0.12$
	2531	26.55 ± 0.11	26.47 ± 0.11	$\mathbf{23.72} \pm 0.11$
ijcnn1	113	19.26 ± 0.11	18.71 ± 0.13	$\mathbf{13.90} \pm 0.07$
	448	12.18 ± 0.13	11.89 ± 0.10	$\mathbf{10.02} \pm 0.09$
	1790	$\mathbf{5.63} \pm 0.07$	$\mathbf{5.70} \pm 0.11$	5.83 ± 0.05
emgpa	159	27.09 ± 0.04	26.98 ± 0.08	$\mathbf{23.68} \pm 0.05$
	633	24.01 ± 0.07	23.83 ± 0.09	$\mathbf{20.61} \pm 0.07$
	2531	21.38 ± 0.14	21.28 ± 0.12	$\mathbf{17.50} \pm 0.09$
vehicle	141	22.73 ± 0.15	22.57 ± 0.15	$\mathbf{16.31} \pm 0.04$
	563	21.27 ± 0.11	21.20 ± 0.14	$\mathbf{15.25} \pm 0.02$
	2247	20.22 ± 0.13	19.98 ± 0.06	$\mathbf{15.06} \pm 0.04$
waveform	159	17.35 ± 0.08	16.86 ± 0.11	$\mathbf{14.83} \pm 0.09$
	633	15.15 ± 0.09	14.96 ± 0.09	$\mathbf{11.84} \pm 0.05$
	2531	14.26 ± 0.09	14.17 ± 0.10	$\mathbf{10.93} \pm 0.03$

Table 3
Time (in seconds) of RBP, Forgetron and OLRD on the various data sets. The best results and those that are not significantly worse (according to the t-test with a p-value less than 0.05) are in bold.

Data set	Budget	RBP	Forgetron	OLRD
a9a	91	8.36 ± 0.61	10.53 ± 0.39	$\mathbf{6.55} \pm 0.16$
	362	$\mathbf{12.45} \pm 1.40$	16.07 ± 0.95	$\mathbf{12.65} \pm 1.20$
	1445	$\mathbf{50.25} \pm 0.99$	60.58 ± 1.30	53.84 ± 0.80
covertype-2	159	$\mathbf{18.98} \pm 2.0$	27.57 ± 1.3	24.46 ± 0.48
	633	$\mathbf{39.49} \pm 5.1$	46.79 ± 5.7	$\mathbf{38.98} \pm 2.1$
	2531	$\mathbf{134.4} \pm 3.8$	175.6 ± 2.6	155.74 ± 1.1
ijcnn1	113	$\mathbf{6.44} \pm 0.28$	8.07 ± 0.07	9.27 ± 0.06
	448	$\mathbf{10.12} \pm 0.16$	12.63 ± 0.43	15.23 ± 0.36
	1790	$\mathbf{15.87} \pm 0.90$	18.73 ± 0.75	$\mathbf{15.69} \pm 0.73$
emgpa	159	$\mathbf{11.56} \pm 0.03$	17.79 ± 0.05	15.93 ± 0.15
	633	$\mathbf{16.00} \pm 0.07$	23.80 ± 0.19	20.37 ± 0.06
	2531	$\mathbf{35.48} \pm 1.60$	52.22 ± 1.70	50.87 ± 1.70
vehicle	141	$\mathbf{17.62} \pm 0.85$	24.15 ± 1.20	19.45 ± 0.32
	563	$\mathbf{32.90} \pm 3.80$	43.73 ± 4.70	36.26 ± 3.70
	2247	$\mathbf{162.90} \pm 3.70$	201.00 ± 4.70	181.17 ± 2.80
waveform	159	$\mathbf{12.28} \pm 0.13$	17.01 ± 0.22	14.94 ± 0.08
	633	$\mathbf{26.94} \pm 0.60$	34.73 ± 0.85	32.02 ± 0.58
	2531	$\mathbf{30.52} \pm 3.50$	69.32 ± 2.50	65.42 ± 2.60

operation is involved. This is then followed by OLRD, and the Forgetron is the slowest. Moreover, as expected, a smaller budget means f_t^B is easier (faster) to compute, and thus the running time is also shorter. However, a small budget may also lead to more mistakes and thus more updates (which slows the algorithm). Hence, the trend between budget and running time is not always monotonic. For example, on the *ijcnn1* data set, when the budget is increased to 1790, the classification error of OLRD decreases significantly, and the corresponding increase in running time is very small.

5.2. Algorithms with variable budgets

In the second experiment, we compare (i) Projectron (Orabona et al., 2009); (ii) Projectron++ (Orabona et al., 2009); (iii) OLRD (with constant stepsize specified in Theorem 2); and (iv) OLRU (with constant stepsize specified in Theorem 3). Except for OLRD, all the other algorithms do not have a fixed budget in advance. For OLRU (with $\gamma = \frac{1}{4}$) and OLRD, we vary the budget by changing their c values in (13) and (23), respectively. For Projectron and Projectron++ we vary the budget by changing their η values in the range $\{0.9, 0.8, \ldots, 0.1, 0.05\}$, as long as the number of SVs obtained is below 6000 (otherwise, the training time will be excessive). To reduce statistical variability, results are averaged over 10 permuted versions of the data set.

Figs. 1 and 2 show the classification errors versus budget and time, respectively. As can be seen, OLRU/OLRD usually require a smaller budget than Projectron/Projectron++ for comparable or better classification accuracy, except on the *covertype-2* and *ijcnn1* data sets. However, even in those cases, OLRU/OLRD decrease the classification error much faster, as they only take $O(B)$ memory and time, while Projectron/Projectron++ take $O(B^2)$ memory and time. Besides, note that Projectron and Projectron++ being mistake-driven, are usually less accurate.

Recall that OLRU has an expected regret of $O(T^{\frac{1+\gamma}{2}})$ and a support set size of $O(T^{1-\gamma})$; while OLRD has an $O(\sqrt{T})$ expected regret and a support set of size $O(\sqrt{T})$. With the current setting of $\gamma = \frac{1}{4}$, OLRU thus needs a larger budget than OLRD for comparable accuracy, though OLRU is usually faster than OLRD as OLRU adopts a simple sparse update.

6. Conclusion

In this paper, we proposed a general framework for online learning with budget. It is based on a simple stochastic strategy with sublinear regret guarantee. Two specific algorithms, based on random updating and random discarding respectively, are presented and both can be realized efficiently. The parameter tuning problem is also discussed and a dynamic budget strategy is proposed to obtain optimal performance for online learning with budget. Experiments on a number of benchmark data sets demonstrate encouraging performance.

The proposed randomized budget algorithms are somewhat conservative and do not utilize the expansion coefficients' magnitudes in designing the update/discard probabilities. Robust probability distributions may be used to further improve performance. Online learning with adaptive stepsize using subgradient information may also allow the design of more aggressive algorithms. Moreover, OLRU suggests an interesting way for online learning with partially labeled instances. These will be further investigated in the future.

Acknowledgments

We thank the editors and anonymous reviewers for their valuable comments and suggestions. This research has been supported by the NSF of Fujian Province (grant number 2012J01247), and partly supported by NSFC (grant numbers 61304199 and 61304210) and the Research Grants Council of the Hong Kong Special Administrative Region under grant 614513.

References

Abernethy, J., Bartlett, P.L., Rakhlin, A., & Tewari, A. (2008). Optimal strategies and minimax lower bounds for online convex games. In *Proceedings of the conference on learning theory*. Helsinki, Finland (pp. 415–424).

Cesa-Bianchi, N., & Gentile, C. (2006). Tracking the best hyperplane with a simple budget perceptron. In *Proceedings of the conference on learning theory*. Pittsburgh, PA, USA (pp. 483–498).

Crammer, K., Kandola, J., & Singer, Y. (2003). Online classification on a budget. In *Advances in neural information processing systems 16* (pp. 225–232).

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2008). The forgetron: a kernel-based perceptron on a budget. *SIAM Journal on Computing*, *37*, 1342–1372.

He, W., & Wu, S. (2012). A kernel-based perceptron with dynamic memory. *Neural Networks*, *25*, 106–113.

Kivinen, J., Smola, A., & Williamson, R. (2004). Online learning with kernels. *IEEE Transactions on Signal Processing*, *52*, 2165–2176.

Orabona, F., Keshet, J., & Caputo, B. (2009). Bounded kernel-based online learning. *Journal of Machine Learning Research*, *10*, 2643–2666.

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

Shalev-Shwartz, S. (2007). *Online learning: theory, algorithms, and applications* (Ph.D. thesis). Hebrew University.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, *4*, 107–194.

Sutskever, I. (2009). A simpler unified analysis of budget perceptrons. In *Proceedings of the international conference on machine learning*. Montreal, Canada (pp. 985–992).

Weston, J., Bordes, A., & Bottou, L. (2005). Online (and offline) on an even tighter budget. In *Proceedings of the international conference on artificial intelligence and statistics*. Barbados (pp. 413–420).

Zhang, L., Yi, J., Jin, R., Lin, M., & He, X. (2013). Online kernel learning with a near optimal sparsity bound. In *Proceedings of the international conference on machine learning*. Atlanta, Georgia, USA (pp. 621–629).

Zhao, P., Wang, J., Wu, P., Jin, R., & Hoi, S. (2012). Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *Proceedings of the international conference on machine learning*. Edinburgh, Scotland.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the international conference on machine learning*. Washington, DC, USA (pp. 928–935).