Ahpatron: A New Budgeted Online Kernel Learning Machine with Tighter Mistake Bound

Yun Liao, Junfan Li, Shizhong Liao*, Qinghua Hu, Jianwu Dang

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
{yliao,junfli,szliao,huqinghua}@tju.edu.cn, jdang@jaist.ac.jp

Abstract

In this paper, we study the mistake bound of online kernel learning on a budget. We propose a new budgeted online kernel learning model, called Ahpatron, which significantly improves the mistake bound of previous work and resolves an open problem related to upper bounds of hypothesis space constraints. We first present an aggressive variant of Perceptron, named AVP, a model without budget, which uses an active updating rule. Then we design a new budget maintenance mechanism, which removes a half of examples, and projects the removed examples onto a hypothesis space spanned by the remaining examples. Ahpatron adopts the above mechanism to approximate AVP. Theoretical analyses prove that Ahpatron has tighter mistake bounds, and experimental results show that Ahpatron outperforms the state-of-the-art algorithms on the same or a smaller budget.

Introduction

Online kernel methods are popular approaches to solving both online machine learning and offline machine learning problems (Kivinen, Smola, and Williamson 2001; Crammer, Kandola, and Singer 2003; Lu et al. 2016; Koppel et al. 2019). Let $(\mathbf{x}_t, y_t)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$ is an instance, and $y_t \in \{-1, 1\}$ is its label, $t = 1, 2, \ldots, T$. Online kernel learning algorithms process the examples on the fly, and produce a sequence of hypotheses $\{f_t\}_{t=1}^{T+1}$ from a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. At any round t, the algorithms maintain $f_t = \sum_{\tau=1}^{t-1} a_\tau \kappa(\mathbf{x}_\tau, \cdot)$ and thus must store $S_t = \{(\mathbf{x}_\tau, y_\tau), a_\tau \neq 0, \tau \leq t-1\}$ in memory. S_t is called the *active set* (Dekel, Shalev-Shwartz, and Singer 2005). The examples in the active set play a similar role to the support vectors in Support Vector Machine (SVM) (Vapnik 1998). The size of f_t is linear with respect to t, and the computational complexity (space complexity and per-round time complexity) is in $O(dt)$, which hinders the deployment of online kernel learning algorithms on computing devices with bounded memory resources.

For bounded memories, many algorithms that only store $B \geq 1$ examples have been proposed (Crammer, Kandola, and Singer 2003; Weston, Bordes, and Bottou 2005; Dekel, Shalev-Shwartz, and Singer 2005; Cheng et al. 2006; Cesa-Bianchi and Gentile 2006; Wang, Crammer, and Vucetic 2012; Li and Liao 2023), where B is the budget. Although the algorithms guarantee constant memory and per-round running time, it also poses a critical question: *How does the prediction performance of the algorithms vary with the budget?* From the experimental perspective, many results have empirically showed that the larger the budget is, the better the algorithms will perform (Dekel, Shalev-Shwartz, and Singer 2005; Zhao et al. 2012; Wang, Crammer, and Vucetic 2012; He and Kwok 2014; Zhang and Liao 2019). From the theoretical perspective, the algorithms that only store B examples have worse regret bounds or mistake bounds than those that store all of the examples (Dekel, Shalev-Shwartz, and Singer 2005; Cesa-Bianchi and Gentile 2006; Orabona, Keshet, and Caputo 2008; Zhao et al. 2012; Wang, Crammer, and Vucetic 2012; He and Kwok 2014). Let $M_T = \{t \in [T] : \mathbf{x}_t \text{ is misclassfied}\}$, $\mathbb{H} = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq U\}$, $U = \frac{\sqrt{B+1}}{4\sqrt{\ln(B+1)}}$ [1], and $\ell_{\text{Hinge}}(\cdot, \cdot)$ be the hinge loss function. For any $f \in \mathbb{H}$, the Forgetron algorithm (Dekel, Shalev-Shwartz, and Singer 2005) enjoys a mistake bound as follows.

$$\begin{aligned} |M_T| &\leq 4L_T(f) + 2\|f\|_{\mathcal{H}}^2, \\ L_T(f) &= \sum_{t=1}^{T} \ell_{\text{Hinge}}(f(\mathbf{x}_t), y_t). \end{aligned} \tag{1}$$

And given $f \in \mathbb{H}$, the RBP algorithm (Cesa-Bianchi and Gentile 2006) enjoys an expected mistake bound as follows.

$$\mathbb{E}[|M_T|] \leq \gamma L_T(f) + \gamma U B + 0.5\gamma U\sqrt{B}\ln\left(0.5\gamma B^2\right), \tag{2}$$

where $U \leq \frac{\gamma-1}{\gamma+1}\sqrt{B}$ [2]. The mistake bound is far from optimal since it becomes worse while B increases. For instance, if $B = O(T)$, then $\mathbb{E}[|M_T|] = O(T)$. The POMDR algorithm (Li and Liao 2023) enjoys a regret bound

*Corresponding author.

[1] There is an open problem in (Dekel, Shalev-Shwartz, and Singer 2005): whether it is possible to remove the $\ln^{-\frac{1}{2}}(B+1)$ factor in U. We have solved the open problem in this paper.

[2] Although RBP removes the $\ln^{-\frac{1}{2}}(B+1)$ factor in U, the mistake bound is an expected bound that is weaker than the deterministic bound. Thus the problem is still open (Dekel, Shalev-Shwartz, and Singer 2008).

of $O\left(U\sqrt{\mathcal{A}_T T/B} + U\sqrt{\mathcal{A}_T}\right)$, where $\mathcal{A}_T$ is a complexity called kernel alignment. For any $f \in \mathbb{H}$, the above regret bound naturally implies the following mistake bound.

$$|M_T| = L_T(f) + O\left(U\sqrt{\mathcal{A}_T} + \frac{U}{\sqrt{B}}\sqrt{\mathcal{A}_T T}\right). \quad (3)$$

The convergence rate of the mistake bound is $O\left(\frac{1}{\sqrt{B}}\right)$. The larger B is, the smaller the mistake bound will be.

In summary, previous work does not give a satisfied answer to the fundamental question. There are two reasons. (i) The mistake bounds in (1) and (2) do not give the convergence rates with respect to B. (ii) Although the mistake bound in (3) gives a convergence rate of $O\left(\frac{1}{\sqrt{B}}\right)$, the coefficient depends on $\sqrt{\mathcal{A}_T T}$ and is unsatisfied. It can be proved that $\mathcal{A}_T \geq \inf_{f\in\mathbb{H}} L_T(f)$ in certain conditions.

In this paper, we will propose a new algorithm, Ahpatron, and answer the question better. We first consider the case of no budget, and propose a variant of Perceptron (Rosenblatt 1958), called AVP, using a more active updating strategy. We prove that AVP improves the mistake bound of Perceptron. Then we propose Ahpatron that approximates AVP and enjoys a $L_T(f) + O\left(\frac{L_T(f)}{\sqrt{B}}\right)$ mistake bound that is better than all of previous results. Ahpatron uses a very simple but effective budget maintaining approach proposed in (Li and Liao 2023), i.e., removing a half of examples. The novelties of Ahpatron include a strategy that selects the removed examples and a projection scheme that keeps the information of the removed examples.

Main Results

Our main results are summarized as follows.

Mistake Bound of AVP For any $f \in \mathcal{H}$, the mistake bounds of AVP and Perceptron are

$$|M_T| \leq 2L_T(f) + \|f\|_{\mathcal{H}}^2 + \Delta_T$$

and

$$|M_T| \leq 2L_T(f) + \|f\|_{\mathcal{H}}^2$$

respectively, where $\Delta_T \leq 0$. AVP improves the mistake bound of Perceptron, and the improvement is non-trivial.

Mistake Bound of Ahpatron Given $f \in \mathbb{H}$, the mistake bound of Ahpatron is

$$|M_T| \leq L_T(f) + \Delta_T + \max\left\{\frac{12U}{\sqrt{B}}L_T(f), \frac{0.9U}{\sqrt{B}}L_T(f) + \frac{\sqrt{B}}{2U}\|f\|_{\mathcal{H}}^2\right\}, \quad (4)$$

where $\Delta_T \leq 0$. We improve the result in (1), since the coefficient on $L_T(f)$ can be smaller. We improve the result in (2), since the dependence on B is better. We also improve the result in (3), since $\min_{f\in\mathbb{H}} L_T(f) = O(\sqrt{\mathcal{A}_T T})$ in certain conditions.

Resolving Open Problem Let $U = \frac{\sqrt{B}}{4}$ in (4). Then the mistake bound of Ahpatron is

$$|M_T| \leq \max\left\{4L_T(f), 1.3L_T(f) + 2\|f\|_{\mathcal{H}}^2\right\} + \Delta_T.$$

Here, we remove the $\ln^{-\frac{1}{2}}(B+1)$ factor in U, and improve the mistake bound in (1). Thus we resolve the open problem posed by Dekel, Shalev-Shwartz, and Singer (2005).

Refined Mistake Bound of Ahpatron We further prove an algorithm-dependent mistake bound for Ahpatron, which can be much better than the mistake bound in (4).

The mistake bound in (4) depends on the selected kernel function and the structure of the examples via $L_T(f)$. In certain benign environments where we have $\min_{f\in\mathbb{H}} L_T(f) \ll T$, Ahpatron performs well using a small budget. For instance, the examples are linearly separable in the feature space induced by the kernel function. In the worst case, i.e, $\min_{f\in\mathbb{H}} L_T(f) \approx T$, our result still coincides with the result in (3).

Related Work

Instead of using a fixed budget, the Projectron and Projectron++ algorithm (Orabona, Keshet, and Caputo 2008) use the approximate linear dependence condition (Engel, Mannor, and Meir 2004) to add the current example into the active set. However, the two algorithms can not precisely control the size of the active set, and may suffer a computational complexity in $O(dT^2)$. The mistake bounds of the two algorithms are also similar to the result in (1).

Besides the budget maintaining technique, there are many other techniques that can keep constant memory for online kernel learning algorithms, such as random features (Rahimi and Recht 2007), Nyström approximation (Williams and Seeger 2001) and matrix sketching (Charikar, Chen, and Farach-Colton 2002). The FOGD algorithm (Wang et al. 2013; Lu et al. 2016) uses random features to approximate kernel function and enjoys a $O\left(\sqrt{T} + \frac{T}{\sqrt{D}}\right)$ regret bound [3], where D is the number of random features. The NOGD algorithm (Wang et al. 2013; Lu et al. 2016) uses the Nyström technique, and enjoys a regret bound of $O\left(\sqrt{T} + \frac{T}{\sqrt{B}}\right)$. The two regret bounds imply a $L_T(f) + O\left(\sqrt{T} + \frac{T}{\sqrt{B}}\right)$ mistake bound, which is much worse than our mistake bound. The SkeGD algorithm (Zhang and Liao 2019) uses randomized sketching and enjoys a regret bound of $O\left(\sqrt{TB}\right)$ under the assumption that the eigenvalues of the kernel matrix decay exponentially. Although the result is better by a constant B, it becomes worse in the case of $B = \Theta\left(T^\mu\right)$, $0 < \mu < 1$.

[3] The original regret bound is $O\left(\|f\|_1\sqrt{T} + \epsilon T\|f\|_1\right)$, where $\|f\|_1 = \sum_{t=1}^T |a_t|$ and $f = \sum_{t=1}^T a_t \kappa(\mathbf{x}_t, \cdot)$, and holds with probability $1 - 2^8\left(\sigma_p/\epsilon\right)^2 \exp\left(-D\epsilon^2/(4d+8)\right)$.

Problem Setting and Preliminaries

Notations

Let $\mathcal{I}_T := \{(\mathbf{x}_t, y_t)\}_{t\in[T]}$ be a sequence of examples, where $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$ is an instance, $y_t \in \{-1, 1\}$ is the label, and $[T] = \{1, \ldots, T\}$. Let $\kappa(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive semidefinite kernel function and $\mathcal{H}$ be the associated RKHS, such that, (i) $\mathcal{H} = \operatorname{span}(\kappa(\mathbf{x}_t, \cdot) \mid t \in [T])$, and (ii) for any $f \in \mathcal{H}, \mathbf{x} \in \mathcal{X}$, it must be $\langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x})$. Denote by $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ the inner product in $\mathcal{H}$. For any $f \in \mathcal{H}$, there exists a weight vector $\mathbf{a} \in \mathbb{R}^T$ such that $f = \sum_{t=1}^{T} a_t\kappa(\mathbf{x}_t, \cdot)$. For another $g = \sum_{t=1}^{T} b_t\kappa(\mathbf{x}_t, \cdot) \in \mathcal{H}$, we define

$$\langle f, g\rangle_{\mathcal{H}} = \sum_{t=1}^{T}\sum_{\tau=1}^{T} a_t b_\tau \kappa\left(\mathbf{x}_t, \mathbf{x}_\tau\right).$$

The inner product induces the norm $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f\rangle_{\mathcal{H}}}$. We further assume that $\max_{\mathbf{x}\in\mathcal{X}} \kappa(\mathbf{x}, \mathbf{x}) \leq 1$. Let $\mathcal{X}$ be bounded. Usual kernel functions, such as polynomial kernels and radial basis kernels, satisfy the assumption. Our results are suitable for any κ such that $\max_{\mathbf{x}\in\mathcal{X}} \kappa(\mathbf{x}, \mathbf{x})$ is bounded. Denote by $\ell_{\text{Hinge}}(u, y) = \max\{0, 1 - uy\}$ and $L_T(f) = \sum_{t=1}^{T} \ell_{\text{Hinge}}\left(f\left(\mathbf{x}_t\right), y_t\right)$, $f \in \mathcal{H}$.

Budgeted Online Kernel Learning

The protocol of online kernel learning can be defined as a game between a learner and an adversary. At any round t, the adversary sends an instance $\mathbf{x}_t \in \mathcal{X}$. Then the learner selects a hypothesis $f_t \in \mathcal{H}$ and makes a prediction $\hat{y}_t = \operatorname{sign}(f_t(\mathbf{x}_t))$. After that the adversary gives the label y_t. The game proceeds to the next round. We rewrite $M_T = \{t \in [T] : \hat{y}_t \neq y_t\}$. The learner aims to minimize $|M_T|$. For any $f \in \mathcal{H}$, we use an upper bound on the mistakes to measure the performance of the learner,

$$|M_T| \leq h\left(\mathcal{I}_T, f\right),$$

where $h(\mathcal{I}_T, f)$ depends on $\mathcal{I}_T$ and f.

At any round t, $f_t = \sum_{\tau=1}^{t-1} a_\tau\kappa\left(\mathbf{x}_\tau, \cdot\right)$. To store f_t in memory, the learner must store $S_t = \{(\mathbf{x}_\tau, y_\tau), a_\tau \neq 0, \tau \leq t-1\}$. The memory cost may be unbounded. To keep the memory bounded, we would limit $|S_t| \leq B$. We call B the budget. The budget also weakens the performance of the learner. It is impossible to provide a non-trivial mistake bound for any $f \in \mathcal{H}$ without additional assumptions on the problem, so we maintain a competitive hypothesis space (Dekel, Shalev-Shwartz, and Singer 2005),

$$\mathbb{H} = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq U\},$$

where U is a constant.

The state-of-the-art mistake bound is (Li and Liao 2023)

$$h\left(\mathcal{I}_T, f\right) = L_T(f) + O\left(U\sqrt{\frac{\mathcal{A}_T T}{B}}\right).$$

Our goal is to design an algorithm that achieves

$$h\left(\mathcal{I}_T, f\right) = L_T(f) + O\left(\frac{U}{\sqrt{B}} L_T(f)\right).$$

Such an upper bound is tighter than previous results.

Perceptron

We first consider online kernel learning without budget. A classical algorithm with non-trivial mistake bounds is Perceptron (Rosenblatt 1958). At any round t, Perceptron predicts $\hat{y}_t = \operatorname{sign}(f_t(\mathbf{x}_t))$. The key is the update rule,

$$f_{t+1} = f_t + y_t\kappa\left(\mathbf{x}_t, \cdot\right) \cdot \mathbb{I}_{\hat{y}_t \neq y_t}.$$

Theorem 1 (Dekel, Shalev-Shwartz, and Singer (2005))**.** *For any sequence of examples $\mathcal{I}_T$ and for any $f \in \mathcal{H}$, the mistake bound of Perceptron satisfies*

$$|M_T| \leq 2L_T(f) + \|f\|_{\mathcal{H}}^2.$$

Theorem 2 (Shalev-Shwartz (2007))**.** *For any $\mathcal{I}_T$ and for any $f \in \mathcal{H}$, the mistake bound of Perceptron satisfies*

$$|M_T| \leq L_T(f) + \|f\|_{\mathcal{H}}\sqrt{L_T(f)} + \|f\|_{\mathcal{H}}^2.$$

If $L_T(f) \leq \|f\|_{\mathcal{H}}^2$, then the mistake bound in Theorem 1 is better than that in Theorem 2. Perceptron stores all of the misclassified examples. Its space complexity is in $O\left(d|M_T|\right)$, which would be $O(dT)$ in the worst case. Many algorithms approximate Perceptron via maintaining a fixed budget, such as Forgetron (Dekel, Shalev-Shwartz, and Singer 2005) and RBP (Cesa-Bianchi and Gentile 2006).

AVP

Perceptron uses a passive updating rule. A more active updating rule is that f_t makes a prediction at a low confidence, i.e., $y_t f_t(\mathbf{x}_t) < \beta_t$, $\beta_t \in [0, 1]$. The updating rule is adopted by ALMA$_p$ (Gentile 2001). If $\beta_t = 0$, then it follows Perceptron. If $\beta_t = 1$, it follows Projectron++ (Orabona, Keshet, and Caputo 2008) and the gradient descent family of algorithms (Zhao et al. 2012; Lu et al. 2016; Zhang and Liao 2019). However, no mistake bound shows that the active updating rule is better than that of Perceptron. There are two technical challenges,

C 1 how to set the value of β_t;

C 2 how to give a tighter analysis.

We will carefully design β_t, and give a novel and solid theoretical analysis of mistake bounds.

Let $\beta_t = 1 - \varepsilon_t$. We redefine the updating rule as follows.

$$f_{t+1} = \operatorname{Proj}_{\mathbb{H}}\left(f_t + \lambda_t \cdot y_t\kappa\left(\mathbf{x}_t, \cdot\right) \cdot \mathbb{I}_{y_t \cdot f_t(\mathbf{x}_t) < 1-\varepsilon_t}\right),$$
$$\operatorname{Proj}_{\mathbb{H}}(g) = \min\left\{1, U \cdot \|g\|_{\mathcal{H}}^{-1}\right\} \cdot g,$$

where $\varepsilon_t \in [0, 1]$ and λ_t is the learning rate. We will prove that setting $\frac{\lambda_t}{2} < \varepsilon_t < 1$ will give smaller mistake bounds. We name this algorithm AVP (Aggressive Variant of Perceptron), and give the pseudo-code in Algorithm 1.

Theorem 3. *Let*

$$M_T' = \{t \in [T] : y_t f_t\left(\mathbf{x}_t\right) \leq 0\},$$
$$N_T = \{t \in [T] : 0 < y_t f_t\left(\mathbf{x}_t\right) < 1 - \varepsilon_t\}.$$

Set $U > 0$, $\lambda_t < 2$, $\delta_t = \|f_t - f\|_{\mathcal{H}}^2 - \|f_{t+1} - f\|_{\mathcal{H}}^2$, and

$$\frac{\lambda_t}{2} < \varepsilon_t < 1. \tag{5}$$

Algorithm 1: AVP

Input: $\{\lambda_t, \varepsilon_t\}_{t=1}^T, U$
Initialization: $f_1 = 0, S_1 = \emptyset$
for $t = 1,\ldots,T$ **do**
  Receive $\mathbf{x}_t$;
  Compute $\hat{y}_t = \text{sign}\left(f_t(\mathbf{x}_t)\right)$;
  if $y_t \cdot f_t(\mathbf{x}_t) < 1 - \varepsilon_t$ **then**
    Update $f_{t+1} = \text{Proj}_{\mathbb{H}}\left(f_t + \lambda_t y_t \kappa\left(\mathbf{x}_t, \cdot\right)\right)$
    $S_{t+1} = S_t \cup \{(\mathbf{x}_t, y_t)\}$
  end if
end for

For any $f \in \mathbb{H}$, the mistake bound of AVP satisfies

$$|M_T'| - L_T(f) \leq \sum_{t \in M_T' \cup N_T} \frac{\delta_t}{2\lambda_t} + \sum_{t \in M_T'} \frac{\lambda_t}{2} + \sum_{t \in N_T} \left(\frac{\lambda_t}{2} - \varepsilon_t\right).$$

Note that $U = +\infty$ implies $\mathbb{H} = \mathcal{H}$. Recalling the definition of M_T, we have $M_T' = M_T \cup \{t \in [T] : \hat{y}_t = y_t, f_t(\mathbf{x}_t) = 0\}$. Thus $|M_T| \leq |M_T'|$.

Eq. (5) solves the first challenge **C** 1. To rise to the second challenge **C** 2, our theoretical analysis makes use of the gap between mistake bounds and cumulative hinge losses, which is given as follows.

$$\begin{aligned} &\sum_{t \in M_T' \cup N_T} \ell_{\text{Hinge}}\left(f_t\left(\mathbf{x}_t\right), y_t\right) - |M_T'| \\ \geq &\sum_{t \in N_T} 1 - y_t f_t(\mathbf{x}_t) \\ \geq &\sum_{t \in N_T} \varepsilon_t. \end{aligned}$$

We will use the non-positive term $-\sum_{t \in N_T} \varepsilon_t$ to counteract the increase of $\sum_{t \in N_T} \frac{\lambda_t}{2}$ on the upper bound. This non-positive term naturally improves the mistake bound of Perceptron and solves **C** 2.

Corollary 1. *Let $U = +\infty$, $\lambda_t = 1$, $\varepsilon_t = \varepsilon \in (\frac{1}{2}, 1)$. For any $f \in \mathcal{H}$, the mistake bound of AVP satisfies*

$$|M_T| \leq 2L_T(f) + \|f\|_{\mathcal{H}}^2 + (1 - 2\varepsilon)\,|N_T|.$$

Note that $(1 - 2\varepsilon)\,|N_T| \leq 0$. Compared with Theorem 1, AVP improves the mistake bound of Perceptron. It is worth mentioning that $|N_T|$ depends on ε. It seems that setting $\varepsilon = 1$ minimizes $(1 - 2\varepsilon)\,|N_T|$. However, if $\varepsilon = 1$, then $|N_T| = 0$. It is hard to give the exact value of $|N_T|$. The only information is that $|N_T| \geq 0$ for all $\varepsilon \in [0, 1)$. According to the upper bound, it is better to set $\frac{1}{2} < \varepsilon < 1$. In this case, $|N_T| > 0$ unless all instances are classified with a high confidence. We will empirically verify that $|N_T| \gg 1$.

Corollary 2. *Let $U < +\infty$, $\lambda_t = \frac{U}{\sqrt{U^2 + \sum_{\tau=1}^t \mathbb{I}_{y_\tau f_\tau(\mathbf{x}_\tau) \leq 0}}}$, $t \geq 1$ and $\frac{\lambda_t}{2} < \varepsilon_t < 1$. For any $f \in \mathbb{H}$, the mistake bound of AVP satisfies*

$$\begin{aligned} |M_T| \leq &\max\left\{L_T(f) + 2U^2 + \Delta_T, 0\right\} + 9U^2 + \\ &3U\sqrt{\max\left\{L_T(f) + 2U^2 + \Delta_T, 0\right\}}, \end{aligned}$$

where $\Delta_T = \sum_{t \in N_T} \left(\frac{\lambda_t}{2} - \varepsilon_t\right)$.

Compared with Theorem 2, the non-positive term Δ_T makes that AVP improves the mistake bound of Perceptron. AVP stores $|M_T'| + |N_T|$ instances in memory. In the worst case, the memory cost of AVP is also in $O(dT)$. In the next section, we will approximate AVP by limiting $|S_t| \leq B$.

Remark 1. *The updating rule of AVP is similar to* ALMA_2 *(Gentile 2001). The essential differences are the values of β_t and λ_t.* ALMA_2 *sets*

$$\beta_t = \frac{(1-\alpha)B}{\|\mathbf{x}_t\|_2^{-1}\sqrt{k}}, \lambda_t = \frac{Ck^{-\frac{1}{2}}}{\|\mathbf{x}_t\|_2}, k = 1 + \sum_{\tau=1}^{t-1} \mathbb{I}_{y_\tau f_\tau(\mathbf{x}_\tau) \leq \beta_\tau},$$

where α, B and C are tunable parameters. In Corollary 2, AVP sets $\beta_t = 1 - \varepsilon_t$ and $\lambda_t = \frac{U}{\sqrt{U^2 + \sum_{\tau=1}^t \mathbb{I}_{y_\tau f_\tau(\mathbf{x}_\tau) \leq 0}}}$.

The values of β_t and λ_t in ALMA_2 *make its mistake bound be similar to Theorem 2 (see Theorem 3 in Gentile (2001)). Thus* ALMA_2 *did not prove that more active updating rule is better than Perceptron. AVP also significantly improves the mistake bound of* ALMA_2.

Ahpatron

Forgetron and RBP approximate Perceptron by limiting $|S_t| \leq B$. If $|S_t| = B$ and we will add $(\mathbf{x}_t, y_t)$ into S_t, then Forgetron removes the oldest example, and RBP randomly removes an example. A recent algorithm POMDR (Li and Liao 2023) adopts a very simple technique that removes a half of examples. We will approximate AVP by removing a half of examples. There are still two technical challenges. (1) How to select the examples that will be removed. (2) How to keep the information of the removed examples. We will propose a heuristic example selecting strategy and a novel projection scheme to address the two challenges.

At any round t, if $|S_t| = B$ and $y_t f_t(\mathbf{x}_t) < 1 - \varepsilon$, then we will remove $\frac{B}{2}$ examples from S_t. Let $f_t = \sum_{i=1}^B \alpha_{r_i} \kappa(\mathbf{x}_{r_i}, \cdot)$, where $(\mathbf{x}_{r_i}, y_{r_i}) \in S_t$. Denote by $S_t = S_{t,1} \cup S_{t,2}$ satisfying (i) $S_{t,1} \cap S_{t,2} = \emptyset$; (ii) $|S_{t,1}| = \frac{B}{2}$; (iii) $\forall (\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,2}$ and $\forall (\mathbf{x}_{r_j}, y_{r_j}) \in S_{t,1}$, $|\alpha_{r_i}| \geq |\alpha_{r_j}|$. We rewrite $f_t = f_{t,1} + f_{t,2}$, where

$$\begin{aligned} f_{t,1} &= \sum_{(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,1}} \alpha_{r_i} \kappa\left(\mathbf{x}_{r_i}, \cdot\right), \\ f_{t,2} &= \sum_{(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,2}} \alpha_{r_i} \kappa\left(\mathbf{x}_{r_i}, \cdot\right). \end{aligned}$$

We will remove the examples in $S_{t,1}$. If we directly reset $f_{t,1} = 0$, then much of information may be lost. To keep as much information as possible, we propose a new projection scheme. The main idea is to project $f_{t,1}$ onto $\mathcal{H}_{t,2} = \overline{\text{span}(\kappa(\mathbf{x}, \cdot) : (\mathbf{x}, y) \in S_{t,2})}$. Let $\mathbb{H}_{t,2} = \{f \in \mathcal{H}_{t,2} : \|f\|_{\mathcal{H}} = c_t U\}$ where $c_t \in (0, 1]$. $\forall f \in \mathcal{H}_{t,2}$, $f = \sum_{(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,2}} \theta_{r_i} \kappa(\mathbf{x}_{r_i}, \cdot)$. The projection scheme is

$$\hat{f}_{t,1} = \arg\min_{f \in \mathcal{H}_{t,2}} \|f - f_{t,1}\|_{\mathcal{H}}^2 + \eta \|\boldsymbol{\theta}\|_2^2, \tag{6}$$

$$\bar{f}_{t,2} = \text{Proj}_{\mathbb{H}_{t,2}}\left(f_{t,2} + \hat{f}_{t,1}\right), \tag{7}$$

Algorithm 2: Ahpatron

Input: $B, U, \lambda, \varepsilon, \eta, \{c_t\}_{t=1}^T$
Initialize $f_0 = 0, S_1 = \emptyset$
for $t = 1, \ldots, T$ **do**
  Receive $\mathbf{x}_t$
  Compute $\hat{y}_t = \operatorname{sign}(f_t(\mathbf{x}_t))$
  if $y_t f_t(\mathbf{x}_t) < 1 - \varepsilon$ **then**
    if $|S_t| < B$ **then**
      Update $f_{t+1} = \operatorname{Proj}_{\mathbb{H}}\left(f_t + \lambda_t y_t \kappa\left(\mathbf{x}_t, \cdot\right)\right)$
      Update $S_{t+1} = S_t \cup \{(\mathbf{x}_t, y_t)\}$
    else
      Divide $S_t = S_{t,1} \cup S_{t,2}$
      Compute $\hat{f}_{t,1}, \bar{f}_{t,2}$ following (6) and (7)
      Remove $S_{t,1}$ and update f_{t+1} following (8)
    end if
  end if
end for

where $\eta\|\boldsymbol{\theta}\|_2^2$ is a regularization term and aims to make (6) solvable. We will explain more in (9).

Let $f_{t,1}^\top = f_{t,1} - \hat{f}_{t,1}$ and $f_{t,2}^\top = (f_{t,2} + \hat{f}_{t,1}) - \bar{f}_{t,2}$. We further rewrite f_t by

$$f_t = f_{t,2} + f_{t,1} = f_{t,2} + \hat{f}_{t,1} + f_{t,1}^\top = \bar{f}_{t,2} + f_{t,2}^\top + f_{t,1}^\top.$$

For the sake of clarity, denote by $\bar{f}_t := \bar{f}_{t,2}$. Then we execute

$$\begin{aligned} \bar{f}_t =& f_t - (f_{t,1}^\top + f_{t,2}^\top), \\ f_{t+1} =& \operatorname{Proj}_{\mathbb{H}}\left(\bar{f}_t + \lambda y_t \kappa(\mathbf{x}_t, \cdot)\right). \end{aligned} \tag{8}$$

It is obvious that $(f_{t,1}^\top + f_{t,2}^\top)$ is removed from f_t. Recalling that $\bar{f}_t$ is a combination of $\frac{B}{2}$ examples. Thus the first step in (8) will remove a half of examples from S_t. We name this algorithm Ahpatron (approximating aggressive Perceptron via halving and projecting) and give the pseudo-code in Algorithm 2.

The projection (6) can be rewritten as follows,

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{\frac{B}{2}}} \left\{\boldsymbol{\theta}^\top (\mathbf{K}_2 + \eta \mathbf{I})\boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \mathbf{K}_{21}\boldsymbol{\alpha}\right\} + \boldsymbol{\alpha}^\top \mathbf{K}_1 \boldsymbol{\alpha},$$

where $\boldsymbol{\alpha} = (\alpha_{r_i})_{(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,1}}$, $\mathbf{K}_2$ is the kernel matrix defined on $S_{t,2}$, $\mathbf{K}_1$ is the kernel matrix defined on $S_{t,1}$, $\mathbf{K}_{21}$ is the kernel matrix define on S_2 and S_1 satisfying $\mathbf{K}_{21}[i,k] = \kappa(\mathbf{x}_{r_i}, \mathbf{x}_{r_k})$, $(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,2}$ and $(\mathbf{x}_{r_k}, y_{r_k}) \in S_{t,1}$. The optimization problem has a closed-form solution,

$$\boldsymbol{\theta}^* = \left(\mathbf{K}_2 + \eta \mathbf{I}\right)^{-1} \mathbf{K}_{21}\boldsymbol{\alpha}, \tag{9}$$

where $\eta > 0$ aims to keep $\left(\mathbf{K}_2 + \eta \mathbf{I}\right)^{-1}$ invertible. We further have

$$\begin{aligned} \hat{f}_{t,1} &= \sum_{(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,2}} \theta_{r_i}^* \kappa\left(\mathbf{x}_{r_i}, \cdot\right), \\ \bar{f}_t &= \frac{c_t U}{\|f_{t,2} + \hat{f}_{t,1}\|_{\mathcal{H}}} \sum_{(\mathbf{x}_{r_i}, y_{r_i}) \in S_{t,2}} \left(\alpha_{r_i} + \theta_{r_i}^*\right) \kappa\left(\mathbf{x}_{r_i}, \cdot\right). \end{aligned}$$

The time complexity of computing $\boldsymbol{\theta}^*$ is $O\left(B^3\right)$ since there is a matrix inversion operation. Note that we solve $\boldsymbol{\theta}^*$ only if $|S_t| = B$, that is, the time interval between two continuous matrix inversion operations is at least $\frac{B}{2}$. Thus the average per-round time complexity is only $O\left(\max\{dB, B^2\}\right)$. The space complexity of Ahpatron is also $O\left(\max\{dB, B^2\}\right)$.

Theoretical Analysis

Lemma 1 upper bounds the times of removing operation, which is critical to the mistake bound analysis.

Lemma 1. *For any sequence $\mathcal{I}_T$, the times of removing operation executed by Ahpatron is at most*

$$\max\left\{\frac{2\left(|M_T'| + |N_T|\right)}{B} - 1, 0\right\},$$

where M_T' and N_T are defined in Theorem 3.

Theorem 4 (Mistake Bound)**.** *Let $c_t = 0.6$ for all $t \in [T]$, $\lambda = \frac{\sqrt{2}U}{\sqrt{B}}$, $B \geq 50$, $U \leq \frac{\sqrt{B}}{4}$ and $\frac{3U}{\sqrt{B}} < \varepsilon < 1$. Given $f \in \mathbb{H}$, for any sequence $\mathcal{I}_T$, the mistake bound of Ahpatron satisfies*

$$|M_T| \leq L_T(f) + \max\left\{\frac{12U}{\sqrt{B}} L_T(f) + \overline{\Delta}, \frac{0.9U}{\sqrt{B}} L_T(f) + \frac{\sqrt{B}}{2U}\|f\|_{\mathcal{H}}^2 + \underline{\Delta}\right\},$$

where

$$\overline{\Delta} = \frac{\frac{3U}{\sqrt{B}} - \varepsilon}{1 - \frac{3U}{\sqrt{B}}}|N_T|, \quad \underline{\Delta} = \frac{\frac{U}{\sqrt{2B}} - \varepsilon}{1 - \frac{U}{\sqrt{2B}}}|N_T|.$$

Note that $\overline{\Delta} \leq 0$ and $\underline{\Delta} \leq 0$. Omitting the constant factors, we obtain a mistake bound as follows.

$$L_T(f) + O\left(\frac{U}{\sqrt{B}} L_T(f) + \frac{\sqrt{B}}{U}\|f\|_{\mathcal{H}}^2 + \max\left\{\overline{\Delta}, \underline{\Delta}\right\}\right).$$

Since $B \ll T$, we can obtain a mistake bound of

$$L_T(f) + O\left(\frac{U}{\sqrt{B}} L_T(f)\right),$$

which explicitly gives the trade-off between mistake bound and budget.

Comparison with Previous Results

In the following, we assume $f \in \mathbb{H}$. The BOGD algorithm (Zhao et al. 2012) enjoys an expected regret bound of $O\left(\frac{UT}{\sqrt{B}} + U\sqrt{T}\right)$. Note that $\mathbb{I}_{\hat{y}_t \neq y_t} \leq \ell_{\text{Hinge}}(f_t(\mathbf{x}_t), y_t)$, we have

$$\mathbb{E}\left[|M_T|\right] = L_T(f) + O\left(\frac{UT}{\sqrt{B}} + U\sqrt{T}\right).$$

Ahpatron significantly improves the mistake bound of BOGD, since $L_T(f) = O(T)$. Besides, our mistake bound is deterministic.

The mistakes of POMDR (Li and Liao 2023) satisfy

$$|M_T| = L_T(f) + O\left(\frac{U\sqrt{\mathcal{A}_T T}}{\sqrt{B}} + \sqrt{\mathcal{A}_T}\right),$$

where $\mathcal{A}_T = \sum_{t=1}^T \kappa(\mathbf{x}_t, \mathbf{x}_t) - \frac{1}{T}\mathbf{Y}_T^\top \mathbf{K}_T \mathbf{Y}_T$ is called "kernel alignment", $\mathbf{Y}_T = (y_1, \ldots, y_T)^\top$ and $\mathbf{K}_T$ is the kernel matrix on $\mathcal{I}_T$.

Theorem 5. *Assume for any* $\mathbf{x} \in \mathcal{X}, \kappa(\mathbf{x}, \mathbf{x}) = 1$. *Let* $U \geq 1$. *Then* $\mathcal{A}_T \geq \min_{f \in \mathbb{H}} L_T(f)$.

Theorem 5 implies $\min_{f \in \mathbb{H}} L_T(f) = O(\sqrt{\mathcal{A}_T T})$, and Ahpatron improves the mistake bound of POMDR.

Let $U = \frac{\sqrt{B+1}}{4\sqrt{\ln(B+1)}}$. Then the mistake bound of Forgetron (Dekel, Shalev-Shwartz, and Singer 2005) satisfies

$$|M_T| \leq 4L_T(f) + 2\|f\|_{\mathcal{H}}^2.$$

Let $U = \frac{\sqrt{B}}{4}$ in Theorem 4. The mistake bound of Ahpatron satisfies

$$|M_T| \leq \max\left\{4L_T(f) + \overline{\Delta}, 1.3L_T(f) + 2\|f\|_{\mathcal{H}}^2 + \underline{\Delta}\right\}. \tag{10}$$

We can see that Ahpatron improves the mistake bound of Forgetron. Note that $U = \frac{\sqrt{B}}{4} > \frac{\sqrt{B+1}}{4\sqrt{\ln(B+1)}}$, where we remove the $\ln^{-\frac{1}{2}}(B+1)$ factor. Thus we solve the open problem posed by Dekel, Shalev-Shwartz, and Singer (2005).

A Refined Result

Now we explain why Ahpatron may obtain smaller mistake bounds via removing a half of examples. Let $\mathcal{J} = \{t \in M_T' \cup N_T : |S_t| = B\}$. At any round $t \in \mathcal{J}$, we remove a half of examples. We can prove that the mistake bound of Ahpatron depends on the following term

$$\frac{U+\lambda}{\lambda} \sum_{t \in \mathcal{J}} \left\|f_t - \bar{f}_t\right\|_{\mathcal{H}} \leq \frac{U+\lambda}{\lambda} \cdot \zeta U \cdot |\mathcal{J}|, \tag{11}$$

where $\zeta \in (0, 2]$. In the worst case, i.e., $\left\|f_t - \bar{f}_t\right\|_{\mathcal{H}} = 2U$, we have $\zeta = 2$. It is natural that if $\bar{f}_t$ is close to f_t for most of $t \in \mathcal{J}$, then $\zeta \ll 2$ is possible. In this case, Ahpatron enjoys smaller mistake bounds.

Theorem 6 (Algorithm-dependent Mistake Bound)**.** *Assume the inequality* (11) *holds with* $0 < \zeta \leq 2$. *Let* $c_t = \frac{\|f_t\|_{\mathcal{H}}}{U}$, $B \geq 16$, $\lambda = \frac{U}{2\sqrt{B}}$. *For any* $\gamma \in (0, 1)$, *let* $U \leq \frac{(1-\gamma)\sqrt{B}}{\frac{1}{4}+\frac{9}{2}\zeta}$ *and* $\left(\frac{1}{4}+\frac{9\zeta}{2}\right)\frac{U}{\sqrt{B}} < \varepsilon < 1$. *Then the mistake bound of Ahpatron satisfies*

$$|M_T| \leq L_T(f) + \frac{1}{\gamma}\left(\frac{1}{4}+\frac{9\zeta}{2}\right)\frac{U}{\sqrt{B}}L_T(f) + \frac{1-2\zeta}{\gamma}\frac{\|f\|_{\mathcal{H}}^2\sqrt{B}}{U} + \Delta_T(\zeta),$$

where

$$\Delta_T(\zeta) = \frac{|N_T|}{1-\left(\frac{1}{4}+\frac{9\zeta}{2}\right)\frac{U}{\sqrt{B}}} \cdot \left(\left(\frac{1}{4}+\frac{9\zeta}{2}\right)\frac{U}{\sqrt{B}} - \varepsilon\right).$$

Theorem 6 clearly shows that the mistake bound of Ahpatron depends on the value of ζ, i.e., the distance between $\bar{f}_t$ and f_t. If $\zeta \ll 2$, then Ahpatron can use a larger value of U and enjoy a smaller mistake bound. For instance, assume $\zeta \leq \frac{1}{16}$, and let $\gamma = \frac{2}{3}$ and $U = \frac{(1-\gamma)\sqrt{B}}{\frac{1}{4}+\frac{9}{2}\zeta} \geq \frac{\sqrt{B}}{1.6}$, we obtain $\frac{1}{3} < \varepsilon < 1$, and the mistake bound of Ahpatron satisfies

$$|M_T| \leq \frac{3}{2}L_T(f) + 2.1\|f\|_{\mathcal{H}}^2 + \Delta_T(\zeta),$$

Datasets	#sample	# Feature	Classes
phishing	11,055	68	2
a9a	48,842	123	2
w8a	49,749	300	2
SUSY	50,000	18	2
ijcnn1	141,691	22	2
cod-rna	271,617	8	2

Table 1: Datasets used in the experiments

which improves the result in (10), including: (i) The value of U is larger, i.e., $U = \frac{\sqrt{B}}{1.6} > \frac{\sqrt{B}}{4}$. The larger the value of U is, the larger $\mathbb{H}$ will be. (ii) The mistake bound is smaller in the case of $L_T(f) > \|f\|_{\mathcal{H}}^2$.

Remark 2. *Theorem 6 also gives a criterion to select* $\bar{f}_t$. *To be specific, the optimal* $\bar{f}_t$ *must be*

$$\operatorname*{arg\,min}_{f \in \mathcal{H}_{t,2}, S_{t,2} \subseteq S_t, |S_{t,2}| = \frac{B}{2}} \|f - f_t\|_{\mathcal{H}}.$$

However, obtaining the optimal solution is computationally infeasible. Our algorithm constructs an approximation solution using a heuristic method.

Experiments

We aim to verify Ahpatron performs better than the state-of-the-art algorithms on the same budget or a smaller budget.

We adopt the Gaussian kernel $\kappa(\mathbf{u}, \mathbf{v}) = \exp(-\frac{\|\mathbf{u}-\mathbf{v}\|_2^2}{2\sigma^2})$, where σ is the width. We download six binary classification datasets from UCI machine learning repository [4] and LIBSVM website [5], as shown in Table 1. We uniformly select 50000 instances from the original *SUSY* dataset. All algorithms are implemented in R on a Windows machine with 2.8 GHz Core(TM) i7-1165G7 CPU [6]. We execute each experiment 10 times with random permutation of all datasets.

The state-of-the-art algorithms we compare with are listed as follows.

- Projectron and Projectron++ (Orabona, Keshet, and Caputo 2008)
  The two algorithms do not remove examples, but use the approximate linear dependence condition (Engel, Mannor, and Meir 2004) to add examples. The two algorithms may maintain a very large budget and thus suffer high memory costs and running time.
- BOGD++ (Zhao et al. 2012), POMDR (Li and Liao 2023)
  When the size of the active set reaches the budget, BOGD++ randomly removes one example, while POMDR removes a half of the examples.
- NOGD (Lu et al. 2016)
  NOGD first selects B examples and constructs the corresponding kernel matrix $\mathbf{K}_B$. Then it constructs a new

[4] http://archive.ics.uci.edu/ml/datasets.php

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

[6] Codes and datasets: https://github.com/alg4ml/Ahpatron.git

Algorithm	phishing, $\sigma = 1$			SUSY, $\sigma = 1$			ijcnn1, $\sigma = 1$		
	AMR (%)	$B\|D$	Time (s)	AMR (%)	$B\|D$	Time (s)	AMR (%)	$B\|D$	Time (s)
Projectron++	7.42 ± 0.15	**571**	3.04	22.05 ± 0.07	**4898**	1330.83	3.01 ± 0.04	**1577**	58.58
Projectron	9.52 ± 0.16	422	1.71	29.10 ± 0.17	2111	108.87	4.14 ± 0.04	1136	28.22
FOGD	7.48 ± 0.16	2000	3.30	26.70 ± 0.60	2000	10.04	3.74 ± 0.23	2000	19.12
BOGD++	10.30 ± 0.20	400	1.01	28.30 ± 0.12	400	3.09	9.21 ± 0.02	600	10.55
NOGD	8.12 ± 0.23	400	3.02	25.49 ± 0.47	400	3.92	4.47 ± 0.06	600	20.71
POMDR	$\mathbf{7.36 \pm 0.16}$	400	**1.29**	25.36 ± 0.25	400	2.79	5.76 ± 0.09	600	9.33
Ahpatron	$\mathbf{7.27 \pm 0.13}$	400	**1.01**	$\mathbf{23.66 \pm 0.17}$	400	**3.62**	$\mathbf{3.72 \pm 0.03}$	600	**8.51**
Algorithm	cod-rna, $\sigma = 1$			w8a, $\sigma = 2$			a9a, $\sigma = 2$		
	AMR (%)	$B\|D$	Time (s)	AMR (%)	$B\|D$	Time (s)	AMR (%)	$B\|D$	Time (s)
Projectron++	10.13 ± 0.03	**5100**	2786.98	2.03 ± 0.03	**1279**	94.37	$\mathbf{16.45 \pm 0.07}$	152	4.22
Projectron	14.43 ± 0.03	2686	453.49	2.90 ± 0.06	306	14.15	21.13 ± 0.17	135	3.46
FOGD	12.72 ± 1.05	2000	47.41	4.18 ± 0.23	300	6.28	16.69 ± 0.11	300	3.71
BOGD++	14.01 ± 0.04	600	12.78	2.76 ± 0.08	300	14.25	19.91 ± 0.12	150	3.34
NOGD	15.85 ± 0.72	600	30.61	2.69 ± 0.04	300	15.04	16.56 ± 0.15	150	3.56
POMDR	$\mathbf{11.76 \pm 0.11}$	600	**11.58**	2.59 ± 0.11	300	15.00	17.09 ± 0.15	150	4.33
Ahpatron	12.33 ± 0.08	600	14.85	$\mathbf{2.23 \pm 0.06}$	300	**11.68**	$\mathbf{16.42 \pm 0.06}$	150	3.14

Table 2: Comparison with the state-of-the-art algorithms

representation in $\mathbb{R}^k$ for each instance by the best rank-k approximation of $\mathbf{K}_B$, where $k \leq B$. The space complexity of NOGD is in $O\left(\max\left\{B^2, Bd\right\}\right)$.

- FOGD (Lu et al. 2016)
  FOGD uses random features (Rahimi and Recht 2007) to approximate kernel functions, and runs online gradient descent in $\mathbb{R}^D$. D is the number of random features. The computational complexity of FOGD is in $O(Dd)$.

We do not compare with Forgetron and RBP, since their performances are much worse than FOGD and NOGD. For BOGD++, NOGD, and FOGD, we choose the stepsize of gradient descent from $\left\{\frac{10^{[-3:1:3]}}{\sqrt{T}}\right\}$. The other parameters of BOGD++ and NOGD follow the original paper. All parameters of POMDR also follow the original paper. For Projectron and Projectron++, there is a parameter $0 < \eta < 1$ balancing the memory costs and prediction performance. We choose $\eta \in \{0.1, 0.9\}$. The smaller η is, the larger the size of the active set is and the better the prediction performance is. For Ahpatron, we set the parameters following Theorem 6, that is, $\eta = 0.0005, \lambda = \frac{U}{\sqrt{4B}}$ and $U = \frac{\sqrt{B}}{2}$. We choose the best $\varepsilon \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ in hindsight, and set $\sigma = 1$ for all datasets. If the per-round running time of Projectron++ is larger than 1 hour, then we set $\sigma = 2$.

Table 2 shows the results. We record the average mistake rates (AMR) and $\mathrm{AMR} := \frac{|M_T|}{T}$. As a whole, Ahpatron performs best on most of datasets on a small budget. Although there are some datasests on which Projectron++ enjoys a lower AMR than Ahpatron, the budget maintained by Projectron++ is very large, such as on the *SUSY*, *ijcnn1*, *cod-rna* and *w8a* datasets. The main reason is that Projectron and Projectron++ only add examples. In practice, it must be careful to choose Projectron++ and Projectron. In contrast, Ahpatron can precisely control the budget and show a comparable performance on all datasets. Specially, Ahpatron even performs better than Projection++ on the *phishing* datasets using a smaller budget. In the supplementary material, we further compare with Projectron++ on the *SUSY*, *ijcnn1*, *cod-rna* and *w8a* dataset. We aim to show that Ahpatron performs better than Projectron++ on the same or a smaller budget.

	phishing	SUSY	cod-rna	w8a	ijcnn1	a9a
$\|N_T\|$	1429	23045	129497	1525	5204	9234

Table 3: The value of $|N_T|$ in Ahpatron

Ahpatron performs better than BOGD++ on all datsets. BOGD++ approximates gradient descent and removes one example, while Ahpatron approximates AVP and removes a half of examples. The results prove that AVP is better than gradient descent and our budget maintaining approach is also better. Ahpatron also performs better than POMDR on most of datasets, since it adopts a different strategy to remove the examples and projection scheme to keep the information of the removed examples. Ahpatron also performs better than NOGD and FOGD on all datasets.

We also report the value of $|N_T|$ in Ahpatron. Table 3 shows the results. It is obvious that $|N_T| \gg 1$ on all datasets. Thus the negative terms in Theorem 4 and Theorem 6 can significantly reduce the mistake bound.

Conclusion

In this paper, we have proposed a new budgeted online kernel learning model, called Ahpatron, which has a tighter mistake bound, and resolved the open problem posed by Dekel, Shalev-Shwartz, and Singer (2005). The key of Ahpatron is the half removing and half projecting budget mechanism, which keeps as more information of the removed examples as possible. We also have proved that the active updating rule can improve the mistake bound.

The mistake bound of Ahpatron explicitly gives the tradeoff between the mistake bound and the budget, which is important for online learning, and is left for future research.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No.62076181). We thank all anonymous reviewers for their valuable comments and suggestions.

References

Cesa-Bianchi, N.; and Gentile, C. 2006. Tracking the best hyperplane with a simple budget perceptron. In *Proceedings of the 19th Annual Conference on Learning Theory*, 483–498.

Charikar, M.; Chen, K. C.; and Farach-Colton, M. 2002. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, 693–703.

Cheng, L.; Vishwanathan, S. V. N.; Schuurmans, D.; Wang, S.; and Caelli, T. 2006. Implicit online learning with kernels. *Advances in Neural Information Processing Systems*, 19: 249–256.

Crammer, K.; Kandola, J. S.; and Singer, Y. 2003. Online classification on a budget. *Advances in Neural Information Processing Systems*, 16: 225–232.

Dekel, O.; Shalev-Shwartz, S.; and Singer, Y. 2005. The Forgetron: A kernel-based perceptron on a fixed budget. *Advances in Neural Information Processing Systems*, 18: 259–266.

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

Engel, Y.; Mannor, S.; and Meir, R. 2004. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8): 2275–2285.

Gentile, C. 2001. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2: 213–242.

He, W.; and Kwok, J. T. 2014. Simple randomized algorithms for online learning with kernels. *Neural Networks*, 60: 17–24.

Kivinen, J.; Smola, A. J.; and Williamson, R. C. 2001. Online learning with kernels. *Advances in Neural Information Processing Systems*, 14: 785–792.

Koppel, A.; Warnell, G.; Stump, E.; and Ribeiro, A. 2019. Parsimonious online learning with kernels via sparse projections in function space. *Journal of Machine Learning Research*, 20(3): 1–44.

Li, J.; and Liao, S. 2023. Improved kernel alignment regret bound for online kernel kearning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, 8597–8604.

Lu, J.; Hoi, S. C. H.; Wang, J.; Zhao, P.; and Liu, Z. 2016. Large scale online kernel learning. *Journal of Machine Learning Research*, 17(47): 1–43.

Orabona, F.; Keshet, J.; and Caputo, B. 2008. The Projectron: A bounded kernel-based Perceptron. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning*, 720–727.

Rahimi, A.; and Recht, B. 2007. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20: 1177–1184.

Rosenblatt, F. 1958. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65: 386–408.

Shalev-Shwartz, S. 2007. *Online learning: Theory, algorithms, and applications*. Ph.D. thesis, The Hebrew University of Jerusalem.

Vapnik, V. N. 1998. *Statistical learning theory*, volume 1. New York: Wiley & Sons.

Wang, J.; Hoi, S. C. H.; Zhao, P.; Zhuang, J.; and Liu, Z. 2013. Large scale online kernel classification. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 1750–1756.

Wang, Z.; Crammer, K.; and Vucetic, S. 2012. Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale SVM training. *Journal of Machine Learning Research*, 13(1): 3103–3131.

Weston, J.; Bordes, A.; and Bottou, L. 2005. Online (and offline) on an even tighter budget. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 413–420.

Williams, C. K. I.; and Seeger, M. 2001. Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 13: 682–688.

Zhang, X.; and Liao, S. 2019. Incremental randomized sketching for online kernel learning. In *Proceedings of the 36th International Conference on Machine Learning*, 7394–7403.

Zhao, P.; Wang, J.; Wu, P.; Jin, R.; and Hoi, S. C. H. 2012. Fast bounded online gradient descent algorithms for scalable kernel-based online learning. In *Proceedings of the 29th International Conference on Machine Learning*, 1075–1082.