Improved Kernel Alignment Regret Bound for Online Kernel Learning

Junfan Li, Shizhong Liao*

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
{junfli,szliao}@tju.edu.cn

Abstract

In this paper, we improve the kernel alignment regret bound for online kernel learning in the regime of the Hinge loss function. Previous algorithm achieves a regret of $O((\mathcal{A}_T T \ln T)^{\frac{1}{4}})$ at a computational complexity (space and per-round time) of $O(\sqrt{\mathcal{A}_T T \ln T})$, where $\mathcal{A}_T$ is called *kernel alignment*. We propose an algorithm whose regret bound and computational complexity are better than previous results. Our results depend on the decay rate of eigenvalues of the kernel matrix. If the eigenvalues of the kernel matrix decay exponentially, then our algorithm enjoys a regret of $O(\sqrt{\mathcal{A}_T})$ at a computational complexity of $O(\ln^2 T)$. Otherwise, our algorithm enjoys a regret of $O((\mathcal{A}_T T)^{\frac{1}{4}})$ at a computational complexity of $O(\sqrt{\mathcal{A}_T T})$. We extend our algorithm to batch learning and obtain a $O(\frac{1}{T}\sqrt{\mathbb{E}[\mathcal{A}_T]})$ excess risk bound which improves the previous $O(1/\sqrt{T})$ bound.

Introduction

Online kernel learning is a popular non-parametric method for solving large-scale batch learning and online learning problems. Online kernel learning algorithms only pass the data once and thus are computationally efficient. Specifically, $\forall t = 1, \ldots, T$, an online learning algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^d$. Then it selects a hypothesis $f_t \in \mathbb{H}$ and makes prediction $f_t(\mathbf{x}_t)$. $\mathbb{H} = \{f \in \mathcal{H} | \|f\|_{\mathcal{H}} \leq U\}$, where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS). After that, the algorithm observes y_t and suffers a loss $\ell(f_t(\mathbf{x}_t), y_t)$. In an online learning setting, $\{(\mathbf{x}_t, y_t)_{t=1}^T\}$ may not be i.i.d., and can even be adversarial. We focus on the online prediction performance, and aim to minimize the cumulative losses $\sum_{t=1}^T \ell(f_t(\mathbf{x}_t), y_t)$. We usually use the *regret* to measure the performance, which is defined as follows,

$$\forall f \in \mathbb{H},\ \mathrm{Reg}(f) = \sum_{t=1}^{T} [\ell(f_t(\mathbf{x}_t), y_t) - \ell(f(\mathbf{x}_t), y_t)]. \quad (1)$$

An effective algorithm must ensure $\mathrm{Reg}(f) = o(T)$, which implies the average loss of the algorithm converges to that of the optimal hypothesis. In this paper, $\ell(\cdot, \cdot)$ is the Hinge loss function, and $y_t \in \{-1, 1\}$.

The minimax lower bound on the regret is $\Omega(\sqrt{T})$ for the Hinge loss function (Abernethy et al. 2008). The online gradient descent (OGD) algorithm (Zinkevich 2003) enjoys a $O(\sqrt{T})$ upper bound which is optimal w.r.t. T. OGD still suffers two weaknesses. (i) It nearly stores all of the observed examples, and thus suffers a $O(dt)$ computational complexity (i.e., space and per-round time complexity), which is prohibitive for large-scale datasets. (ii) The $O(\sqrt{T})$ bound is too pessimistic for certain benign environments. In practice, the data might be learnt easily or have some intrinsic structures that can be used to circumvent the $\Omega(\sqrt{T})$ barrier. Most of the previous algorithms only address the first weakness. Such algorithms store limited examples or construct explicit feature mapping. For instance, the BOGD and FOGD algorithm (Zhao et al. 2012; Lu et al. 2016) suffer a $O(dB)$ computational complexity and achieve a $O(\sqrt{T} + \frac{T}{\sqrt{B}})$ [1] regret bound, where B is the size of budget or the number of random features. The NOGD algorithm (Lu et al. 2016) which uses Nyström approach to construct explicit feature mapping, enjoys a regret of $O(\sqrt{T} + \frac{T}{B})$ at a computational complexity of $O(dB + B^2)$. The above algorithms reduce the computational complexity at the expense of regret bound. The SkeGD algorithm (Zhang and Liao 2019) which uses randomized sketching to construct explicit feature mapping, enjoys a regret of $O(\sqrt{TB})$ at a computational complexity of $O(dB\mathrm{poly}(\ln T))$. Although the results are better for a constant B, they become worse in the case of $B = \Theta(T^\mu)$, $0 < \mu < 1$.

The only algorithm that addresses both weaknesses simultaneously is $\mathrm{B(AO)_2KS}$ (Liao and Li 2021). It achieves a regret of $O((\mathcal{A}_T T \ln T)^{\frac{1}{4}})$ at a computational complexity of $O(d\sqrt{\mathcal{A}_T T \ln T})$. The kernel alignment, $\mathcal{A}_T$, measures how well the kernel function matches with the data. If we choose a good kernel function, then $\mathcal{A}_T = o(T)$ is possible. In this case, $\mathrm{B(AO)_2KS}$ circumvents the $\Omega(\sqrt{T})$ barrier and only suffers a $o(T)$ computational complexity. If we choose a bad kernel function, $\mathrm{B(AO)_2KS}$ still nearly matches the results of OGD. A natural question arises: *Is it possible to achieve a regret of $O(\sqrt{\mathcal{A}_T})$ at a computational complexity of $O(\mathrm{poly}(\ln T))$?* An algorithm with such characteristics

*Corresponding author.

[1] The regret bound of BOGD is obtained from Eq.(14) in (Zhao et al. 2012).

Algorithm	Regret bound	Computational complexity
OGD	$O(\sqrt{T})$	$O(dT)$
BOGD	$O(\sqrt{T}+\frac{T}{\sqrt{B}})$	$O(dB)$
NOGD	$O(\sqrt{T}+\frac{T}{B})$	$O(dB+B^2)$
SkeGD	$O(\sqrt{TB})$	$O(dB\mathrm{poly}(\ln T))$
$\mathrm{B(AO)_2KS}$	$O((\mathcal{A}_T T\ln T)^{\frac{1}{4}})$	$O(d\sqrt{\mathcal{A}_T T\ln T})$
POMDR	$O(\sqrt{\mathcal{A}_T})$	$O((d\wedge\ln T)\ln T)$
	$O(\sqrt{\mathcal{A}_T}+\frac{\sqrt{T\mathcal{A}_T}}{\sqrt{B}})$	$O(dB)$

Table 1: Regret bound and computational complexity of online kernel learning algorithms in the regime of the Hinge loss. $B\leq T$ is the size of budget. $\max\{a,b\}=a\wedge b$.

would constitute a significant improvement on the existing algorithms, including OGD, SkeGD and $\mathrm{B(AO)_2KS}$.

In this paper, we propose an algorithm, named POMDR, and affirmatively answer the question under some mild assumption. Our results depend on how fast the eigenvalues of the kernel matrix decay. If the eigenvalues decay exponentially, then POMDR enjoys a regret of $O(\sqrt{\mathcal{A}_T})$ at a computational complexity of $O(d\ln T+\ln^2 T)$. Otherwise, POMDR enjoys a regret of $O(\sqrt{\mathcal{A}_T}+\frac{\sqrt{T\mathcal{A}_T}}{\sqrt{B}})$ at a computational complexity of $O(dB)$. B is a tunable parameter. If $B=\sqrt{\mathcal{A}_T T}$, then POMDR still improves the results of $\mathrm{B(AO)_2KS}$. Table 1 summarizes the related results.

Our algorithm combines two techniques, namely optimistic mirror descent (OMD) (Chiang et al. 2012; Rakhlin and Sridharan 2013) and the *approximate linear dependence (ALD) condition* (Engel, Mannor, and Meir 2004), and gives a new budget maintaining approach. OMD achieves the $O(\sqrt{\mathcal{A}_T})$ regret bound. The ALD condition ensures the $O(d\ln T+\ln^2 T)$ computational complexity. The main challenge is to analyze the size of the budget maintained by the ALD condition. Previous analysis assumed that examples are i.i.d, which is commonly violated in an online learning setting. We give a new and cleaner analysis for the size of budget. To be specific, if the eigenvalues of the kernel matrix decay exponentially, then the size is $O(\ln\frac{T}{\alpha})$, where $\alpha>0$ is a threshold parameter. If the eigenvalues decay polynomially with degree $p\geq 1$, then the size is $O((T/\alpha)^{\frac{1}{p}})$.

We extend our algorithm to batch learning, and give a $O(\frac{1}{T}\sqrt{\mathbb{E}[\mathcal{A}_T]})$ excess risk bound in expectation. Such a result may also circumvent the $\Omega(\frac{1}{\sqrt{T}})$ barrier (Srebro, Sridharan, and Tewari 2010).

Related Work

Engel, Mannor, and Meir (2004) first used the ALD condition to control the budget of the kernel recursive least-squares algorithm, and proved the budget is bounded. The Projectron algorithm (Orabona, Keshet, and Caputo 2008) also used the ALD condition to maintain the budget and proved a same result. An significant improvement on the size of the budget was given by Sun, Gomez, and Schmidhuber (2012). They proved similar results with our results. However, their results are not suitable for online learning, since they assume that the examples are i.i.d. Our results do not require such an assumption and are more general.

Our results reveal new trade-offs between regret bound and computational costs for online kernel learning in the regime of the Hinge loss function. Previous work only focus on loss functions with strong curvature properties, such as smoothness and exp-concave. For exp-concave loss functions, the PROS-N-KONS algorithm (Calandriello, Lazaric, and Valko 2017a) which combines the online Newton step algorithm (Hazan, Agarwal, and Kale 2007) and KORS (Calandriello, Lazaric, and Valko 2017b), achieves a regret of $O(\mathrm{d_{eff}}(\gamma)\ln^2 T)$ at a computational complexity of $O(\mathrm{d^2_{eff}}(\gamma)\ln^4 T)$. $\mathrm{d_{eff}}(\gamma)$ is the effective dimension of kernel matrix. If the eigenvalues of kernel matrix decay exponentially, then $\mathrm{d_{eff}}(\gamma)=O(\ln T)$. For smooth loss functions, the OSKL algorithm (Zhang et al. 2013) achieves a regret of $O(\sqrt{L^*_T})$ at a computational complexity of $O(dL^*_T)$. L^*_T is the cumulative losses of optimal hypothesis. The above two types of regret bound can also circumvent the $\Omega(\sqrt{T})$ barrier, but are not suitable for the Hinge loss function that does not enjoy strong curvature properties.

In batch learning setting, the examples are i.i.d. sampled from a fixed distribution. In this paper, our problem setting captures the classical support vector machines (SVM). Our algorithm solves the SVM in an online approach and outputs an approximate solution $\bar{f}$. We prove that $\bar{f}$ achieves a $O(\frac{1}{T}\sqrt{\mathbb{E}[\mathcal{A}_T]})$ excess risk bound. The Pegasos algorithm (Shalev-Shwartz, Singer, and Srebro 2007) can only achieves a $O(\frac{1}{\sqrt{T}})$ excess risk bound. If the eigenvalues of the kernel matrix decay exponentially, then the time complexity of our algorithm is $O(T\cdot(d\ln T+\ln^2 T))$, while the time complexity of Pegasos is $O(dT^2)$.

Problem Setting

Let $\mathcal{X}=\{\mathbf{x}\in\mathbb{R}^d|\|\mathbf{x}\|_2<\infty\}$ and $\mathcal{I}_T=\{(\mathbf{x}_t,y_t)_{t\in[T]}\}$ be a sequence of examples, where $[T]=\{1,\ldots,T\}$, $\mathbf{x}_t\in\mathcal{X}, y_t\in\{-1,1\}$. Let $\kappa(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\to[A,D]$ be a positive semidefinite kernel function, where $A>0$. Denote by $\mathcal{H}$ the RKHS associated with κ, such that (i) $\langle f,\kappa(\mathbf{x},\cdot)\rangle_{\mathcal{H}}=f(\mathbf{x})$; (ii) $\mathcal{H}=\overline{\mathrm{span}(\kappa(\mathbf{x}_t,\cdot):t\in[T])}$. We define $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ as the inner product in $\mathcal{H}$, which induces the norm $\|f\|_{\mathcal{H}}=\sqrt{\langle f,f\rangle_{\mathcal{H}}}$. Denote by $\mathbb{H}=\{f\in\mathcal{H}|\|f\|_{\mathcal{H}}\leq U\}$. The hinge loss function is $\ell(f(\mathbf{x}),y)=\max\{0,1-yf(\mathbf{x})\}$.

Let $\psi(f)$ be a strongly convex regularizer defined on $f\in\mathcal{H}$. Denote by $\mathcal{D}_\psi(f,g)$ the Bregman divergence,

$$\mathcal{D}_\psi(f,g)=\psi(f)-\psi(g)-\langle\nabla\psi(g),f-g\rangle.$$

The protocol of online learning in $\mathbb{H}$ is as follows: at any round t, an adversary first sends an instance $\mathbf{x}_t\in\mathcal{X}$. An learner uses an algorithm to choose a hypothesis $f_t\in\mathbb{H}$, and makes the prediction $\mathrm{sign}(f_t(\mathbf{x}_t))$. Then the adversary reveals the label y_t. We aim to minimize the regret w.r.t. any $f\in\mathbb{H}$, denoted by $\mathrm{Reg}(f)$ which is defined in (1).

OGD achieves $\mathrm{Reg}(f)=O(U\sqrt{T})$ which is optimal in the worst-case. We will prove a data-dependent regret

bound, that is, $\mathrm{Reg}(f) = O(U\sqrt{A_T})$, where $\mathcal{A}_T$ is called *kernel alignment* (Liao and Li 2021) defined as follows

$$\mathcal{A}_T = \sum_{t=1}^{T} \kappa(\mathbf{x}_t, \mathbf{x}_t) - \frac{1}{T}\mathbf{Y}_T^\top \mathbf{K}_T \mathbf{Y}_T.$$

$\mathbf{K}_T$ is the kernel matrix on $\mathcal{I}_T$ and $\mathbf{Y}_T = (y_1, \ldots, y_T)^\top$. $\mathbf{Y}_T^\top \mathbf{K}_T \mathbf{Y}_T$ is called *kernel polarization* (Baram 2005), a classical kernel selection criterion. If $\mathbf{K}_T$ is the ideal kernel matrix $\mathbf{Y}_T \mathbf{Y}_T^\top$, then $\mathcal{A}_T = \Theta(1)$. More generally, if κ matches well with the data, then we expect that $\mathcal{A}_T \ll T$. In this case, the $O(U\sqrt{A_T})$ bound circumvents the $\Omega(U\sqrt{T})$ barrier. In the worse case, i.e., $\mathcal{A}_T \approx T$, we still have $O(U\sqrt{A_T}) = O(U\sqrt{T})$.

Algorithm

Our algorithm consists of two phases. The first one is named Projected Optimistic Mirror Descent (POMD). The second one is Optimistic Mirror Descent with Removing (OMDR).

POMD

POMD is based on the optimistic mirror descent framework (OMD) (Chiang et al. 2012; Rakhlin and Sridharan 2013). For all $t \geq 1$, OMD maintains $f'_{t-1}, f_t \in \mathcal{H}$. Let S_t be a budget storing a subset of $\{(\mathbf{x}_\tau, y_\tau)_{\tau=1}^{t-1}\}$ and $\nabla_t = \nabla \ell(f_t(\mathbf{x}_t), y_t)$. OMD is defined as follows,

$$f_t = \arg\min_{f \in \mathcal{H}} \left\{ \langle f, \bar{\nabla}_t \rangle + \mathcal{D}_{\psi_t}(f, f'_{t-1}) \right\}, \tag{2}$$

$$f'_t = \arg\min_{f \in \mathcal{H}} \left\{ \langle f, \nabla_t \rangle + \mathcal{D}_{\psi_t}(f, f'_{t-1}) \right\}, \tag{3}$$

where $\bar{\nabla}_t$ is an optimistic estimator of ∇_t. When receiving $\mathbf{x}_t$, we first compute f_t and then predict $\hat{y}_t = \mathrm{sign}(f_t(\mathbf{x}_t))$. After observing y_t, we compute the gradient ∇_t and execute the update (3). It is obvious that, if $\nabla_t = 0$, then $f'_t = f'_{t-1}$. In this case, we do not add $(\mathbf{x}_t, y_t)$ into S_t. If $\nabla_t = -y_t \kappa(\mathbf{x}_t, \cdot)$, then $f'_t \neq f'_{t-1}$ and we must add $(\mathbf{x}_t, y_t)$ into S_t which increases the memory cost. To address this issue, we use the ALD condition (Engel, Mannor, and Meir 2004) to maintain the budget S_t.

We consider any round t at which $\nabla_t = -y_t \kappa(\mathbf{x}_t, \cdot)$. The ALD condition measures whether $\kappa(\mathbf{x}_t, \cdot)$ is approximate linear dependence with the instances in S_t. Let $\mathbf{\Phi}_{S_t} = (\kappa(\mathbf{x}, \cdot)_{\mathbf{x} \in S_t})$. We first compute the projection error

$$\left(\min_{\boldsymbol{\beta} \in \mathbb{R}^{|S_t|}} \|\mathbf{\Phi}_{S_t} \boldsymbol{\beta} - \kappa(\mathbf{x}_t, \cdot)\|_{\mathcal{H}}^2 \right) =: \alpha_t. \tag{4}$$

The solution [2] is

$$\boldsymbol{\beta}_t^* = \mathbf{K}_{S_t}^{-1} \mathbf{\Phi}_{S_t}^\top \kappa(\mathbf{x}_t, \cdot),$$

where $\mathbf{K}_{S_t}$ is the kernel matrix defined on S_t. If $\alpha_t = 0$, then $\kappa(\mathbf{x}_t, \cdot)$ is linear dependence with the instances in S_t. Thus we do not add $(\mathbf{x}_t, y_t)$ into S_t. Linear dependence is a strong condition. We introduce a threshold for α_t and define the ALD condition as follows

$$\mathrm{ALD}_t : \sqrt{\alpha_t} \leq \sqrt{D} \cdot T^{-\zeta},\ \zeta \in (0, 1], \tag{5}$$

[2] If $S_t = \emptyset$, then we set $\boldsymbol{\beta}_t^* = 0$ and $\alpha_t = D$.

where $\sup_{\mathbf{x} \in \mathcal{X}} \kappa(\mathbf{x}, \mathbf{x}) \leq D$. If ALD_t holds, then $\kappa(\mathbf{x}_t, \cdot)$ is approximate linear dependence with the instances in S_t. We can safely replace $\kappa(\mathbf{x}_t, \cdot)$ with $\mathbf{\Phi}_{S_t} \boldsymbol{\beta}_t^*$, and do not add $(\mathbf{x}_t, y_t)$ into S_t. We replace (3) with (6),

$$\begin{cases} f'_t = \arg\min_{f \in \mathbb{H}} \left\{ \langle f, \hat{\nabla}_t \rangle + \mathcal{D}_{\psi_t}(f, f'_{t-1}) \right\}, \\ \hat{\nabla}_t = -y_t \mathbf{\Phi}_{S_t} \boldsymbol{\beta}_t^*. \end{cases} \tag{6}$$

If ALD_t does not hold, that is, $\kappa(\mathbf{x}_t, \cdot)$ can not be approximated by $\mathbf{\Phi}_{S_t} \boldsymbol{\beta}_t^*$, then we add $(\mathbf{x}_t, y_t)$ into S_t and replace (3) with (7),

$$f'_t = \arg\min_{f \in \mathbb{H}} \left\{ \langle f, \nabla_t \rangle + \mathcal{D}_{\psi_t}(f, f'_{t-1}) \right\}. \tag{7}$$

We set $\psi_t(f) = \frac{1}{2\lambda_t} \|f\|_{\mathcal{H}}^2$. Then (2), (6) and (7) become gradient descent. The learning rate λ_t is defined as follows

$$\lambda_t = \frac{U}{\sqrt{3 + \sum_{\tau=1}^{t-1} \max\{\|\tilde{\nabla}_\tau - \bar{\nabla}_\tau\|_{\mathcal{H}}^2 - \|\bar{\nabla}_\tau\|_{\mathcal{H}}^2, 0\}}},$$

$$\tilde{\nabla}_\tau = \hat{\nabla}_\tau, \text{ if } \mathrm{ALD}_\tau \text{ holds},$$

$$\tilde{\nabla}_\tau = \nabla_\tau, \text{ otherwise}.$$

Note that POMD allows $f_t \in \mathcal{H}$ and $f'_t \in \mathbb{H}$, while OMD requires both f_t and f'_t belong to $\mathcal{H}$ (or $\mathbb{H}$). The slight modification gives improved regret bound. More precisely, there will be a negative term, $-\|\bar{\nabla}_t\|_2^2$, in the regret bound. We use projection to ensure $f'_t \in \mathbb{H}$, and thus name this procedure POMD (Projected OMD).

OMDR

At any round t, we can compute $\mathbf{K}_{S_t}^{-1}$ incrementally. Thus the space and per-round time complexity of POMD are $O(d|S_t| + |S_t|^2)$. We will prove that $|S_T|$ depends on how fast the eigenvalues of the kernel matrix decay. If the eigenvalues decay exponentially, then $|S_T| = O(\ln T)$. More rigorous results are given in Theorem 1. Otherwise, $|S_T|$ may be large and POMD suffers high computational costs. To address this issue, we set a threshold on $|S_t|$.

We first execute POMD. Let $B_0 = \Theta(\ln T)$ and $\bar{t} = \min_{t \in [T]}\{t : |S_t| = B_0\}$. If $\bar{t} < T$, then the eigenvalues of the kernel matrix may not decay exponentially. In this case, we stop executing POMD for $t \geq \bar{t} + 1$. To be specific, we always execute the following two steps,

$$f_t = \arg\min_{f \in \mathcal{H}} \left\{ \langle f, \bar{\nabla}_t \rangle + \mathcal{D}_{\psi_t}(f, f'_{t-1}) \right\},$$

$$f'_t = \arg\min_{f \in \mathbb{H}} \left\{ \langle f, \nabla_t \rangle + \mathcal{D}_{\psi_t}(f, f'_{t-1}) \right\}. \tag{8}$$

If $\nabla_t \neq 0$, we add $(\mathbf{x}_t, y_t)$ into S_t. Let $B > B_0$ be another threshold. If $|S_t| = B$, then we remove $B/2$ examples from S_t. Denote by $\{t_i \in [\bar{t}+1, T]\}_{i=1}^N$ the time instances at which $|S_{t_i}| = B$. Let $S_{t_i} = \{(\mathbf{x}_{\tau_j}, y_{\tau_j})_{j=1}^B\}$, where $\tau_j < t_i$. We will delete $\{(\mathbf{x}_{\tau_j}, y_{\tau_j})_{j=B/2+1}^B\}$ from S_{t_i}. According to the representer theorem, we can rewrite $f'_{t_i} = \sum_{j=1}^B a_{\tau_j} \kappa(\mathbf{x}_{\tau_j}, \cdot)$. For each $j \geq B/2+1, \ldots, B$, we define $\mathbf{x}_{\tau_{\underline{j}}}$ as follows

$$\mathbf{x}_{\tau_{\underline{j}}} = \arg\max_{i=1,\ldots,B/2} \kappa(\mathbf{x}_{\tau_i}, \mathbf{x}_{\tau_j}).$$

To keep more information as possible, we construct $\underline{f}'_{t_i}$ by

$$\underline{f}'_{t_i} = \sum_{j=1}^{B/2} a_{\tau_j} \kappa(\mathbf{x}_{\tau_j}, \cdot) + \sum_{j=B/2+1}^{B} a_{\tau_j} \kappa(\mathbf{x}_{\tau_j}, \cdot). \quad (9)$$

Now we redefine $f'_{t_i} = \underline{f}'_{t_i} \cdot U \cdot \|\underline{f}'_{t_i}\|_{\mathcal{H}}^{-1}$. Next we reset the learning rate after the removing operation

$$\lambda_t = \frac{U}{\sqrt{\max_{t\in[t_j+1,t_{j+1}]} \delta_t + \sum_{\tau=t_j+1}^{t-1} \delta_\tau}}, \quad (10)$$
$$\delta_\tau = \max\{\|\nabla_\tau - \bar{\nabla}_\tau\|_{\mathcal{H}}^2 - \|\bar{\nabla}_\tau\|_{\mathcal{H}}^2, 0\}.$$

We name this procedure OMDR. The space and average per-round time complexity of OMDR is $O(dB)$.

Optimistic Estimator

$\bar{\nabla}_t$ depends on the desired regret bound. We will prove that the regret bound depends on the following term,

$$\sqrt{\sum_{t\in[T]} \delta_t} = \sqrt{\sum_{t\in[T]} ([\|\nabla_t - \bar{\nabla}_t\|_{\mathcal{H}}^2 - \|\bar{\nabla}_t\|_{\mathcal{H}}^2] \wedge 0)}.$$

To achieve the kernel alignment regret bound, the optimal value of $\bar{\nabla}_t$ is $\sum_{\tau=1}^{t-1} \frac{-y_\tau}{t-1} \kappa(\mathbf{x}_\tau, \cdot)$. However, such a value needs to store the past $t-1$ examples. To avoid this issue, we define $\bar{\nabla}_t = -\sum_{r=1}^{M_t} \frac{y_{t-r}}{M_t} \kappa(\mathbf{x}_{t-r}, \cdot)$ and $\bar{\nabla}_1 = 0$. Let $M \geq 1$ be a small constant. Then we define $M_t = t-1$ for $2 \leq t \leq M$ and $M_t = M$ for $t > M$. Our $\bar{\nabla}_t$ only needs to store the past M examples, and does not increases the computational costs. We name the algorithm POMDR and give the pseudo-code in Algorithm 1.

Algorithm 1: POMDR

Input: U, ζ, B, B_0, M.
Initialization: $f'_0 = \bar{\nabla}_1 = 0$, $\bar{t} = +\infty$, $S_1 = \emptyset$.
for $t = 1, \ldots, T$ **do**
  Receive $\mathbf{x}_t$
  Compute λ_t
  Compute f_t following (2)
  Make prediction $\hat{y}_t = \mathrm{sign}(f_t(\mathbf{x}_t))$
  Receive y_t and compute loss $\ell(f_t(\mathbf{x}_t), y_t)$
  if $\ell(f_t(\mathbf{x}_t), y_t) > 0$ **then**
    if $\bar{t} \geq T$ **then**
      Compute $\boldsymbol{\beta}_t^* = \mathbf{K}_{S_t}^{-1} \boldsymbol{\Phi}_{S_t}^\top \kappa(\mathbf{x}_t, \cdot)$
      Compute α_t following (4)
      if ALD_t holds **then**
        Compute $\hat{\nabla}_t$
        Update f'_t following (6)
      else
        Update f'_t following (7)
        $S_{t+1} = S_t \cup \{(\mathbf{x}_t, y_t)\}$
        if $|S_{t+1}| == B_0$, **then** $\bar{t} = t+1$
      end if
    else
      Update f'_t following (8)
      $S_{t+1} = S_t \cup \{(\mathbf{x}_t, y_t)\}$
      if $|S_{t+1}| == B$, **then** compute $\underline{f}'_t$ (see (9)) and f'_t
    end if
  end if
end for

Main Results

In this section, we give the size of the budget generated by the ALD condition and the kernel alignment regret bound.

The Size of Budget

Theorem 1. *Let $S_1 = \emptyset$ and ALD_t be defined in (5) where $\alpha = D \cdot T^{-2\zeta}$ for a certain $\zeta \in (0, 1]$. For all $t \leq T-1$, if ALD_t does not hold, then $S_{t+1} = S_t \cup \{(\mathbf{x}_t, y_t)\}$. Otherwise, $S_{t+1} = S_t$. Let $\{\lambda_i\}_{i=1}^T$ be the eigenvalues of $\mathbf{K}_T$ sorted in decreasing order. If $\{\lambda_i\}_{i=1}^T$ decay exponentially, that is, there is a constant $R_0 > 0$ and $0 < r < 1$ such that $\lambda_i \leq R_0 r^i$, then $|S_T| \leq 2\frac{\ln(\frac{C_1 R_0}{\alpha})}{\ln r^{-1}}$. If $\{\lambda_i\}_{i=1}^T$ decay polynomially, that is, there is a constant $R_0 > 0$ and $p \geq 1$, such that $\lambda_i \leq R_0 i^{-p}$, then $|S_T| \leq \mathrm{e}(\frac{C_2 R_0}{\alpha})^{\frac{1}{p}}$. In both cases, C_1 and C_2 are constants, and $R_0 = \Theta(T)$.*

It is worth mentioning that we use the ALD condition to update S_t only if $\nabla_t \neq 0$. In this case, Theorem 1 still holds, since our proof is independent of the condition $\nabla_t \neq 0$. Initial analyses of the ALD condition only proved that the budget is bounded (Engel, Mannor, and Meir 2004). An improved result was given by Sun, Gomez, and Schmidhuber (2012). They proved similar results with Theorem 1. There are three main differences between their results (Sun, Gomez, and Schmidhuber 2012) and our results. (i) Their results require that $\{\mathbf{x}_t\}_{t=1}^T$ are sampled i.i.d. from a fixed distribution, which may not hold in an online learning setting. (ii) Their results hold in a high-probability, while our results are deterministic. (iii) Our analyses are simpler.

The KORS algorithm (Calandriello, Lazaric, and Valko 2017b) uses the ridge leverage scores to construct sampling probability, and then randomly adds examples. KORS ensures $|S_T| = O(\mathrm{d}_{\mathrm{eff}}(\gamma) \ln^2 T)$, where $\mathrm{d}_{\mathrm{eff}}(\gamma) = \mathrm{tr}(\mathbf{K}_T(\mathbf{K}_T + \gamma\mathbf{I})^{-1})$ is the effective dimension. If $\lambda_i \leq R_0 r^i$, then $\mathrm{d}_{\mathrm{eff}}(\gamma) = O(\ln(R_0/\gamma))$. If $\lambda_i \leq R_0 i^{-p}$, then $\mathrm{d}_{\mathrm{eff}}(\gamma) = O((R_0/\gamma)^{\frac{1}{p}})$. KORS is worse than the ALD condition by a factor of order $O(\ln^2 T)$.

Proof Sketch of Theorem 1. We first state a key lemma which proves that the eigenvalues of a Hermitian matrix $\mathbf{A} \in \mathbb{C}^{T\times T}$ are interlaced with those of any principal submatrix $\mathbf{B} \in \mathbb{C}^{m\times m}$, $m \leq T-1$.

Lemma 1 (Theorem 4.3.28 in Horn and Johnson (2012) [3])**.** *Let $\mathbf{A}$ be a Hermitian matrix of order T, and let $\mathbf{B}$ be a principle submatrix of $\mathbf{A}$ of order m. If $\lambda_T \leq \lambda_{T-1} \leq \ldots \leq \lambda_2 \leq \lambda_1$ lists the eigenvalues of $\mathbf{A}$ and $\mu_m \leq \mu_{m-1} \leq \ldots \leq \mu_1$ lists the eigenvalues of $\mathbf{B}$, then*

$$\lambda_{i+T-m} \leq \mu_i \leq \lambda_i, \ i = 1, 2, \ldots, m.$$

[3]The reviewer brings Horn and Johnson (2012) to our attention. Theorem 4.3.28 in (Horn and Johnson 2012) makes our proof cleaner. Our initial manuscripts uses Theorem 1 in Hwang (2004).

The kernel matrix $\mathbf{K}_T$ which is positive semidefinite and real symmetric, satisfies Lemma 1. Observing that

$$\begin{aligned}\alpha_t =&\kappa(\mathbf{x}_t,\mathbf{x}_t) - (\mathbf{\Phi}_{S_t}^\top \kappa(\mathbf{x}_t,\cdot))^\top \mathbf{K}_{S_t}^{-1}\mathbf{\Phi}_{S_t}^\top \kappa(\mathbf{x}_t,\cdot)\\ =&\frac{\det(\mathbf{K}_{S_t\cup\{(\mathbf{x}_t,y_t)\}})}{\det(\mathbf{K}_{S_t})}.\end{aligned}$$

Let $|S_T| = k$ and $\{t_j\}_{j=1}^k$ be the set of time index at which ALD_{t_j} does not hold. $\mathbf{K}_{S_{t_j}}$ is a j-order principle submatrix of $\mathbf{K}_T$. Let $\{\lambda_i^{(j)}\}_{i=1}^j$ be the eigenvalues of $\mathbf{K}_{S_{t_j}}$, where $j = 1,\dots,k$. Let $\{\lambda_j\}_{j=1}^T$ be the eigenvalues of $\mathbf{K}_T$.

If $k \le 2$, Theorem 1 always holds. Next we assume that $k > 3$. We focus on the eigenvalues of $\mathbf{K}_{S_{t_k}}$. We first consider the case, $\lambda_j \le R_0 r^j$, $j = 1,\dots,T$. Lemma 1 gives $\lambda_k^{(k)} \le \lambda_k \le R_0 r^k$.

- **Case 1**: $\lambda_k^{(k)} = R_0 r^k$.
  If ALD_t does not hold, then it must be $\alpha_t > DT^{-2\zeta} =: \alpha$. We have
  $$\frac{\det(\mathbf{K}_{S_{t_k}})}{\det(\mathbf{K}_{S_{t_1}})} = \frac{\lambda_k^{(k)}\cdot\prod_{j=1}^{k-1}\lambda_j^{(k)}}{\lambda_1^{(1)}} > \alpha^{k-1}. \tag{11}$$
  Let $T = k$ and $m = 1$ in Lemma 1. We can obtain $\lambda_1^{(1)} \ge \lambda_k^{(k)}$. Thus it must be
  $$\alpha^{k-1} < \prod_{j=1}^{k-1}\lambda_j^{(k)}. \tag{12}$$
  Lemma 1 gives $\lambda_j^{(k)} \le \lambda_j, j = 1,\dots,k-1$.
  Since $\lambda_j \le R_0 r^j$, we have
  $$R_0^{k-1} r^{k(k-1)} < \prod_{j=1}^{k-1}\lambda_j^{(k)} \le \prod_{j=1}^{k-1}\lambda_j \le R_0^{k-1} r^{\frac{k(k-1)}{2}}.$$
  Solving (12) gives $k < 2\frac{\ln(\frac{R_0}{\alpha})}{\ln r^{-1}}$.
- **Case 2**: $\lambda_k^{(k)} < R_0 r^k$.
  We will prove that if k is large, then there is a contradiction. We start with (11). Rearranging terms gives
  $$\begin{aligned}\lambda_k^{(k)} > \frac{\lambda_1^{(1)}\alpha^{k-1}}{\prod_{j=1}^{k-1}\lambda_j^{(k)}} \ge& A\alpha^{k-1}\cdot R_0^{-k+1}\cdot r^{-\frac{k(k-1)}{2}}\\ =&R_0 r^k\cdot A\alpha^{k-1}\cdot R_0^{-k} r^{-k-\frac{k(k-1)}{2}},\end{aligned}$$
  where the second inequality comes from the fact $\lambda_j^{(k)} \le \lambda_j, j = 1,\dots,k-1$, and $\lambda_1^{(1)} = \kappa(\mathbf{x}_{t_1},\mathbf{x}_{t_1}) \ge A$. Let
  $$\alpha^{k-1}\cdot R_0^{-k} r^{-k-\frac{k(k-1)}{2}} > \frac{1}{A}.$$
  Solving for k yields $k > 2\frac{\ln(\frac{C_1 R_0}{\alpha})}{\ln r^{-1}} - 2$, where C_1 is a constant depending on A. In this case, we further obtain $\lambda_k^{(k)} > R_0 r^k$ which contradicts with the condition $\lambda_k^{(k)} < R_0 r^k$. Thus it must be
  $$k \le 2\frac{\ln(\frac{C_1 R_0}{\alpha})}{\ln r^{-1}} - 2.$$

Combining the two cases, we conclude the first statement. Next we consider the case, $\lambda_j \le R_0 j^{-p}$, $j = 1,\dots,T$.

- **Case 1**: $\lambda_k^{(k)} = R_0 k^{-p}$.
  We start with (12). First we can obtain
  $$R_0^{k-1} k^{-p(k-1)} < \prod_{j=1}^{k-1}\lambda_j^{(k)} \le R_0^{k-1}\prod_{j=1}^{k-1} j^{-p}.$$
  According to (12), we obtain a necessary condition
  $$((k-1)!)^p < R_0^{k-1}\cdot(\alpha^{-1})^{k-1}.$$
  Using Stirling's formula, i.e., $k! \approx \sqrt{2\pi k}(k/\mathrm{e})^k$, we can obtain $k < \mathrm{e}\cdot\left(\frac{R_0}{\alpha}\right)^{\frac{1}{p}} + 1$.
- **Case 2**: $\lambda_k^{(k)} < R_0 k^{-p}$.
  We start with (11). Similarly, we have
  $$\lambda_k^{(k)} > \frac{\lambda_1^{(1)}\alpha^{k-1}}{\prod_{j=1}^{k-1}\lambda_j^{(k)}} \ge R_0 k^{-p}\cdot A\alpha^{k-1}\cdot R_0^{-k}\cdot(k!)^p.$$
  Let $\alpha^{k-1}\cdot R_0^{-k}\cdot(k!)^p > \frac{1}{A}$. Solving the inequality gives
  $$k > \mathrm{e}\cdot\frac{1}{A^{pk}} R_0^{\frac{1}{p}}(\alpha^{-1})^{\frac{k-1}{kp}}.$$
  In this case, we have $\lambda_k^{(k)} > R_0 k^{-p}$ which contradicts with the condition $\lambda_k^{(k)} < R_0 k^{-p}$. Thus
  $$k \le \mathrm{e}\cdot\frac{1}{A^{pk}} R_0^{\frac{1}{p}}(\alpha^{-1})^{\frac{k-1}{kp}} \le \mathrm{e}\cdot\left(C_2\frac{R_0}{\alpha}\right)^{\frac{1}{p}},$$
  where C_2 is a constant depending on A.

Up to now, we conclude the proof. □

Regret Bound

Theorem 2. *Let $U > 0$ and $B_0 > \frac{2}{\ln r^{-1}}\ln(\frac{C_1 R_0}{\alpha})$ where $\alpha = D\cdot T^{-2\zeta}$. Let $\zeta = 1$. If the eigenvalues of $\mathbf{K}_T$ decay exponentially, then the regret of POMDR satisfies*

$$\forall f\in\mathbb{H},\quad \mathrm{Reg}(f) \le 3U\sqrt{\sum_{\tau=1}^T \delta_\tau} + 8U,$$

and the space and per-round time complexity is $O(d\ln(T) + \ln^2(T))$. Otherwise, the regret of POMDR satisfies

$$\mathrm{Reg}(f) \le 3U\sqrt{\sum_{\tau=1}^T \delta_\tau} + 8U + 3\sqrt{2}U\frac{\sqrt{T\sum_{\tau=1}^T\delta_\tau}}{\sqrt{B}},$$

and the space and per-round time complexity is $O(dB)$. δ_τ is given by (10).

Note that POMDR needs to tune B_0 which must satisfy $B_0 > \frac{2}{\ln r^{-1}}\ln(\frac{C_1 R_0}{\alpha})$. Since $R_0 = \Theta(T)$ and $\alpha = DT^{-2}$, we can empirically set $B_0 = \lceil c\cdot\ln T\rceil$ where $c > 4$.

Our regret bound improves the previous optimistic regret bounds. Both in constrained and unconstrained case, the

initial regret bound of OMD is $O(\sqrt{\sum_{\tau=1}^{T}\|\nabla_\tau-\bar{\nabla}_\tau\|_2^2})$ (Chiang et al. 2012), which is worse than our bound. In the unconstrained case, Cutkosky (2019) proposed an algorithm with a regret bound of order

$$O\left(B_T(\mathbf{w})\sqrt{\left\{1\wedge\sum_{\tau=1}^{T}[\|\nabla_\tau-\bar{\nabla}_\tau\|_2^2-\|\bar{\nabla}_\tau\|_2^2]\right\}}\right),$$

where $B_T(\mathbf{w})=\|\mathbf{w}\|\sqrt{\ln(T\|\mathbf{w}\|)}$. However, their algorithm can not give similar regret bound in the constrained case. Our algorithm can be used to both constrained and unconstrained case. In the constrained case, Bhaskara et al. (2020) proposed an algorithm with a regret bound of order

$$O\left(U\ln(T)\sqrt{1+\sum_{\tau=1}^{T}([\|\nabla_\tau-\bar{\nabla}_\tau\|_2^2-\|\bar{\nabla}_\tau\|_2^2]\wedge 0)}\right),$$

which is worse than our bound by a factor of order $O(\ln T)$.

Corollary 1. *Let $\bar{\nabla}_t=-\sum_{r=1}^{M_t}\frac{y_{t-r}}{M_t}\kappa(\mathbf{x}_{t-r},\cdot)$. Under the condition of Theorem 2, if the eigenvalues of $\mathbf{K}_T$ decay exponentially, then the regret of POMDR satisfies*

$$\forall f\in\mathbb{H},\quad \mathrm{Reg}(f)\leq 6U\sqrt{\mathcal{A}_T+D}+8U.$$

Otherwise, the regret of POMDR satisfies, $\forall f\in\mathbb{H}$,

$$\mathrm{Reg}(f)\leq 6U\sqrt{\mathcal{A}_T+D}+8U+6\sqrt{2}U\frac{\sqrt{T(\mathcal{A}_T+D)}}{\sqrt{B}}.$$

$\mathrm{B(AO)_2KS}$ (Liao and Li 2021) achieves a $O((\mathcal{A}_T T\ln T)^{\frac{1}{4}})$ regret bound with high probability and suffers a $O(d\sqrt{\mathcal{A}_T T\ln T})$ computational complexity. We compare our results with that of $\mathrm{B(AO)_2KS}$.

- **Case 1**: $\{\lambda_i\}_{i=1}^T$ decay exponentially.
  POMDR achieves a regret of $O(\sqrt{\mathcal{A}_T})$ at a computational complexity of $O(d\ln T+\ln^2 T)$. Thus POMDR significantly improves the results of $\mathrm{B(AO)_2KS}$.
- **Case 2**: $\{\lambda_i\}_{i=1}^T$ decay polynomially.
  Let $B=\sqrt{T\mathcal{A}_T}$. Then POMDR achieves a regret of $O((\mathcal{A}_T T)^{\frac{1}{4}})$ at a computational complexity of $O(d\sqrt{\mathcal{A}_T T})$. POMDR improves the regret bound of $\mathrm{B(AO)_2KS}$ by a $O(\ln^{\frac{1}{4}} T)$ factor, and reduces the computational complexity by a $O(\sqrt{\ln T})$ factor.

Next we compare our regret bounds with the worst-case regret bounds. OGD (Zinkevich 2003) achieves a regret of $O(\sqrt{T})$ at a computational complexity of $O(dT)$. Our results are never worse than that of OGD, and can beat the results of OGD in the case of $\mathcal{A}_T\ll T$. SkeGD (Zhang and Liao 2019) which uses randomized sketching to construct explicit feature mapping, achieves a regret of $O(\sqrt{TB})$ at a computational complexity of $O(dB\mathrm{poly}(\ln T))$. The results become worse in the case of $B=\Theta(T^\mu)$, $0<\mu<1$.

Remark 1. *If we always execute POMD, then $\mathrm{Reg}(f)=O(U\sqrt{\mathcal{A}_T})$. If $\{\lambda_i\}_{i=1}^T$ decay exponentially, then the computational complexity is $O(d\ln T+\ln^2 T)$. If $\{\lambda_i\}_{i=1}^T$ decay polynomially, then the computational complexity is $O(\min\{d(\frac{T}{\alpha})^{\frac{1}{p}}+(\frac{T}{\alpha})^{\frac{2}{p}},T^2\})$. The larger p is, the smaller the computational complexity will be. Since $\alpha=D/T^2$, the computational complexity is larger than that of OGD for $p\leq 3$. This is the reason that we execute OMDR.*

Extension: Batch Learning

Let $\mathcal{D}$ be an unknown distribution over $\mathcal{X}\times\{-1,1\}$. In batch learning setting, $\{(\mathbf{x}_t,y_t)\}_{t=1}^T$ are i.i.d. sampled from $\mathcal{D}$. For any $f\in\mathbb{H}$, let $\mathcal{E}(f)=\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[\ell(f(\mathbf{x}),y)]$ be the risk (or expected loss) of f. The goal of batch learning is to learn a hypothesis $\bar{f}\in\mathbb{H}$ with small risk. Similar with the notation of regret, we focus on the excess risk defined as follows

$$\mathcal{E}(\bar{f})-\min_{f\in\mathbb{H}}\mathcal{E}(f).$$

Classical statistical learning theory analyzes the excess risk of (regularized) empirical risk minimizer, i.e.,

$$\hat{f}=\arg\min_{f\in\mathbb{H}}\left[\frac{1}{T}\sum_{t=1}^{T}\ell(f(\mathbf{x}_t),y_t)+\gamma R(f)\right],\ \gamma\geq 0,$$

where $R(f)$ is a regularization function. Our algorithm computes an approximate minimizer $\bar{f}$ as a proxy of $\hat{f}$. We will prove the excess risk of $\bar{f}$. Our algorithm only passes the data once and thus is computationally efficiently.

We run POMDR on $\{(\mathbf{x}_t,y_t)_{t=1}^T\}$ and output a fixed hypothesis $\bar{f}$ for predicting new instances. To be specific, let $\{f_t\}_{t=1}^T$ be the hypotheses produced by POMDR. Then we sample $r\in\{1,2,\ldots,T\}$ uniformly, and denote $\bar{f}=f_r$. The selection of $\bar{f}$ follows the standard online-to-batch conversion technique (Helmbold and Warmuth 1995).

Let $\phi_\kappa(\cdot):\mathbb{R}^d\to\mathcal{H}$ be the feature mapping induced by κ, and $\mathcal{C}=\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_\mathcal{X}}[\phi_\kappa(\mathbf{x})\otimes\phi_\kappa(\mathbf{x})]$ be the covariance operator, where $\mathcal{D}_\mathcal{X}$ be the marginal distribution on $\mathcal{X}$.

Theorem 3 (Excess risk bound)**.** *Let $\mathcal{I}_T=\{(\mathbf{x}_t,y_t)_{t=1}^T\}$ be i.i.d. sampled from $\mathcal{D}$ and ℓ be the hinge loss function. If the eigenvalues of $\mathcal{C}$ decay exponentially, then w.p. at least $1-\delta$, the excess risk of $\bar{f}$ satisfies, $\forall f\in\mathbb{H}$,*

$$\mathbb{E}_{\mathcal{I}_T,r}[\mathcal{E}(\bar{f})]-\mathcal{E}(f)\leq 6U\frac{\sqrt{\mathbb{E}_{\mathcal{I}_T}[\mathcal{A}_T]+D}}{T}+\frac{8U}{T},$$

and the space and per-round time complexity is $O(d(\ln(T)/\ln(r\delta^{-1}))+(\ln(T)/\ln(r\delta^{-1}))^2)$. Otherwise, w.p. at least $1-\delta$, the excess risk satisfies

$$\forall f\in\mathbb{H},\quad \mathbb{E}_{\mathcal{I}_T,r}[\mathcal{E}(\bar{f})]-\mathcal{E}(f)$$
$$\leq\frac{6U\sqrt{\mathbb{E}_{\mathcal{I}_T}[\mathcal{A}_T]+D}}{T}+\frac{8U}{T}+6\sqrt{2}U\frac{\sqrt{\mathbb{E}_{\mathcal{I}_T}[\mathcal{A}_T]+D}}{\sqrt{TB}},$$

and the space and per-round time complexity is $O(dB)$.

For L-Lipschitz loss functions, Srebro, Sridharan, and Tewari (2010) proved a $\Omega(\frac{LU}{\sqrt{T}})$ lower bound on the excess risk. The Pegasos algorithm (Shalev-Shwartz, Singer, and Srebro 2007) achieves the optimal upper bound $O(\frac{LU}{\sqrt{T}})$. Let $B=T$. Our algorithm gives a data-dependent excess risk bound that can circumvent the $\Omega(1/\sqrt{T})$ bound. For instance, if $\mathbb{E}_{\mathcal{I}_T}[\mathcal{A}_T]=\Theta(T^\mu)$, $0\leq\mu<1$, then our algorithm achieves a $O(\frac{1}{T^{1-\mu/2}})$ excess risk bound. In the worst case, i.e., $\mu=1$, our result is still optimal.

Experiments

In this section, we verify the following two goals.

G 1 If the kernel function is well tuned, then $\mathcal{A}_T \ll T$, or the eigenvalues decay exponentially in some real datasets. In this case, the size of budget is $O(\ln T)$ and our algorithm only executes POMD.

G 2 Within a fixed budget size B, our algorithm shows better or similar prediction performance and running time w.r.t. the state-of-the-art algorithms.

Experimental Setups

We adopt the Gaussian kernel $\kappa(\mathbf{x}, \mathbf{v}) = \exp(-\frac{\|\mathbf{x}-\mathbf{v}\|_2^2}{2\sigma^2})$. We choose three classification datasets (*w8a:49,749, magic04:19,020, mushrooms:8,124*) from UCI machine learning repository [4]. More experimental results are shown in the supplementary materials. We do not compare with OGD, since it suffers $O(dt)$ per-round time complexity which is prohibitive. We compare with some variants of OGD, including FOGD, NOGD (Lu et al. 2016) and SkeGD (Zhang and Liao 2019). We also compare with $\mathrm{B(AO)_2KS}$ (Liao and Li 2021) which is an approximation of OMD. We exclude BOGD (Zhao et al. 2012), since its regret bound is same with FOGD and its performance is worse than FOGD.

For all baseline algorithms, we tune the stepsize of gradient descent from $10^{[-3:1:3]}/\sqrt{T}$. For the other parameters, we follow the original papers. For POMDR, we set $B_0 = \lceil 15 \ln T \rceil$, $B = 400$, $M = 15$, $U = 25$ and multiply by a constant $c \in \{0.05, 0.1\}$ on the learning rate λ_t (see (10)). Such values of parameters do not change our regret bound. For the ALD condition (see (5)), we set the threshold $10T^{-\zeta}$, $\zeta \in \{0.5, 2/3\}$. For all algorithms, we tune the kernel parameter $\sigma \in 2^{[-2:1:6]}$. We randomly permutate the examples in the datasets 10 times and report the average results. All algorithms are implemented with R on a Windows machine with 2.8 GHz Core(TM) i7-1165G7 CPU [5].

Experimental Results

Table 2 shows the average mistake ratio (AMR) and the average running time of all algorithms. The second column in Table 2 gives the budget size or number of random features (for FOGD). As a whole, our algorithm achieves the best prediction performance. The running time of our algorithm is also comparable with all baseline algorithms. Our algorithm performs better than FOGD, NOGD and SkeGD, since our algorithm uses OMD to update hypothesis and adaptive learning rates. Our algorithm performs better than $\mathrm{B(AO)_2KS}$. The reason is that $\mathrm{B(AO)_2KS}$ uses the restart technique when the budget exceeds B, while our algorithm only removes a half of examples which produces a better initial hypothesis than restart. The results verify **G** 2.

Next we verify **G** 1. We change the kernel parameter, σ, and record $\mathcal{A}_T$ and $\bar{t}$. Table 3 gives the results. Theorem 2 proves our regret bound depends on $\sum_{\tau=1}^{T} \delta_\tau \leq 4(\mathcal{A}_T + D)$. $\mathcal{A}_T$ is a coarse approximation of $\sum_{\tau=1}^{T} \delta_\tau$. We use $\sum_{\tau=1}^{T} \delta_\tau$ as a proxy of $\mathcal{A}_T$ and show it in Table 3. We can find that if σ is well tuned, then $\mathcal{A}_T \ll T$, such as the *w8a* and *mushrooms* datasets. In this case, our data-dependent regret bound is smaller than the worst-case bound. Besides, the last row of Table 3 gives the value of $\bar{t}$. If σ is well tuned, then $\bar{t} = +\infty$ (see the **Initialization** of Algorithm 1). In this case, our algorithm only executes POMD and the size of budget is smaller than $B_0 = \lceil 15 \ln T \rceil$. The results verify **G** 1. It is worth mentioning that the kernel function whose eigenvalues decay exponentially may not be good. The third row in Table 3 verifies this claim. How to choose a good kernel function is left to future work.

Algorithm	B	magic04, $\sigma = 0.5$, $\zeta = 2/3$	
		AMR (%)	Time (s)
FOGD	400	16.88 ± 0.15	0.74 ± 0.01
NOGD	400	17.31 ± 0.20	1.57 ± 0.08
SkeGD	400	24.17 ± 0.95	1.07 ± 0.06
$\mathrm{B(AO)_2KS}$	400	22.04 ± 0.43	0.95 ± 0.05
POMDR	400	$\mathbf{16.17 \pm 0.21}$	0.91 ± 0.07
Algorithm	B	w8a, $\sigma = 4$, $\zeta = 0.5$	
		AMR (%)	Time (s)
FOGD	400	2.25 ± 0.04	8.29 ± 0.05
NOGD	400	2.89 ± 0.09	23.65 ± 0.17
SkeGD	400	3.04 ± 0.01	10.72 ± 0.15
$\mathrm{B(AO)_2KS}$	400	2.91 ± 0.02	14.08 ± 0.70
POMDR	400	$\mathbf{2.12 \pm 0.06}$	18.36 ± 0.11
Algorithm	B	mushrooms, $\sigma = 2$, $\zeta = 2/3$	
		AMR (%)	Time (s)
FOGD	400	0.31 ± 0.03	0.70 ± 0.02
NOGD	400	1.71 ± 0.33	1.87 ± 0.13
SkeGD	400	0.26 ± 0.02	0.51 ± 0.03
$\mathrm{B(AO)_2KS}$	400	1.67 ± 0.25	0.76 ± 0.05
POMDR	400	$\mathbf{0.21 \pm 0.03}$	0.44 ± 0.03

Table 2: Average online mistake ratio (AMR) of algorithms.

-	magic04		w8a		mushrooms	
σ	0.5	4	4	64	0.5	2
AMR	**16.17**	23.26	**2.12**	3.12	0.83	**0.21**
$\mathcal{A}_T$	6385	10337	2594	4256	8122	68
$\bar{t}$	469	$+\infty$	3582	$+\infty$	135	$+\infty$

Table 3: The values of $\mathcal{A}_T$ and $\bar{t}$ on different σ.

Conclusion

In this paper, we proposed a new online kernel learning algorithm and improved the previous kernel alignment regret bound and computational complexity simultaneously. Our algorithm combines the OMD framework and the ALD condition, and also invents a new budget maintaining approach. Our main theoretical contribution is that we proved the size of budget maintained by the ALD condition. Our result does not require the examples satisfying i.i.d. and implies further application of the ALD condition on online kernel learning.

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

[5] The codes: https://github.com/JunfLi-TJU/OKL-Hinge

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grants No. 62076181. We thank all anonymous reviewers for their valuable comments and suggestions, especially for bringing Horn and Johnson (2012) to our attention.

References

Abernethy, J. D.; Bartlett, P. L.; Rakhlin, A.; and Tewari, A. 2008. Optimal Stragies and Minimax Lower Bounds for Online Convex Games. In *Proceedings of the 21st Annual Conference on Learning Theory*, 415–424.

Baram, Y. 2005. Learning by Kernel Polarization. *Neural Computation*, 17(6): 1264–1275.

Bhaskara, A.; Cutkosky, A.; Kumar, R.; and Purohit, M. 2020. Online Learning with Imperfect Hints. In *Proceedings of the 37th International Conference on Machine Learning*, 822–831.

Calandriello, D.; Lazaric, A.; and Valko, M. 2017a. Efficient Second-Order Online Kernel Learning with Adaptive Embedding. *Advances in Neural Information Processing Systems*, 30: 6140–6150.

Calandriello, D.; Lazaric, A.; and Valko, M. 2017b. Second-Order Kernel Online Convex Optimization with Adaptive Sketching. In *Proceedings of the 34th International Conference on Machine Learning*, 645–653.

Chiang, C.; Yang, T.; Lee, C.; Mahdavi, M.; Lu, C.; Jin, R.; and Zhu, S. 2012. Online Optimization with Gradual Variations. In *Proceedings of the 25th Annual Conference on Learning Theory*, 6.1–6.20.

Cutkosky, A. 2019. Combining Online Learning Guarantees. In *Proceedings of the 32nd Conference on Learning Theory*, 895–913.

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

Hazan, E.; Agarwal, A.; and Kale, S. 2007. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2): 169–192.

Helmbold, D. P.; and Warmuth, M. K. 1995. On Weak Learning. *Journal of Computer and System Sciences*, 50(3): 551–573.

Horn, R. A.; and Johnson, C. R. 2012. *Matrix Analysis*. Cambridge University Press, 2nd edition.

Hwang, S.-G. 2004. Cauchys Interlace Theorem for Eigenvalues of Hermitian Matrices. *The American Mathematical Monthly*, 111(2): 157–59.

Liao, S.; and Li, J. 2021. High-Probability Kernel Alignment Regret Bounds for Online Kernel Selection. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 67–83.

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.

Rakhlin, A.; and Sridharan, K. 2013. Online Learning with Predictable Sequences. In *Proceedings of the 26th Annual Conference on Learning Theory*, 993–1019.

Shalev-Shwartz, S.; Singer, Y.; and Srebro, N. 2007. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, 807–814.

Srebro, N.; Sridharan, K.; and Tewari, A. 2010. Smoothness, Low Noise and Fast Rates. *Advances in Neural Information Processing Systems*, 23: 2199–2207.

Sun, Y.; Gomez, F. J.; and Schmidhuber, J. 2012. On the Size of the Online Kernel Sparsification Dictionary. In *Proceedings of the 29th International Conference on Machine Learning*, 329–336.

Zhang, L.; Yi, J.; Jin, R.; Lin, M.; and He, X. 2013. Online Kernel Learning with a Near Optimal Sparsity Bound. In *Proceedings of the 30th International Conference on Machine Learning*, 621–629.

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.

Zinkevich, M. 2003. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 928–936.