Second-Order Kernel Online Convex Optimization with Adaptive Sketching

Daniele Calandriello [1] Alessandro Lazaric [1] Michal Valko [1]

Abstract

Kernel online convex optimization (KOCO) is a framework combining the expressiveness of non-parametric kernel models with the regret guarantees of online learning. First-order KOCO methods such as functional gradient descent require only $\mathcal{O}(t)$ time and space per iteration, and, when the only information on the losses is their convexity, achieve a minimax optimal $\mathcal{O}(\sqrt{T})$ regret. Nonetheless, many common losses in kernel problems, such as squared loss, logistic loss, and squared hinge loss posses stronger curvature that can be exploited. In this case, second-order KOCO methods achieve $\mathcal{O}(\log(\operatorname{Det}(\mathbf{K})))$ regret, which we show scales as $\mathcal{O}(d_{\text{eff}} \log T)$, where d_{eff} is the effective dimension of the problem and is usually much smaller than $\mathcal{O}(\sqrt{T})$. The main drawback of second-order methods is their much higher $\mathcal{O}(t^2)$ space and time complexity. In this paper, we introduce *kernel online Newton step* (KONS), a new second-order KOCO method that also achieves $\mathcal{O}(d_{\text{eff}} \log T)$ regret. To address the computational complexity of second-order methods, we introduce a new matrix sketching algorithm for the kernel matrix $\mathbf{K}_t$, and show that for a chosen parameter $\gamma \leq 1$ our Sketched-KONS reduces the space and time complexity by a factor of γ^2 to $\mathcal{O}(t^2\gamma^2)$ space and time per iteration, while incurring only $1/\gamma$ times more regret.

1. Introduction

Online convex optimization (OCO) (Zinkevich, 2003) models the problem of convex optimization over $\mathbb{R}^d$ as a game over $t \in \{1, \ldots, T\}$ time steps between an adversary and the player. In its linear version, that we refer to as linear-OCO (LOCO), the adversary chooses a sequence of arbitrary convex losses ℓ_t and points $\mathbf{x}_t$, and a player chooses weights $\mathbf{w}_t$ and predicts $\mathbf{x}_t^{\mathsf{T}}\mathbf{w}_t$. The goal of the player is to minimize the regret, defined as the difference between the losses of the predictions obtained using the weights played by the player and the best fixed weight in hindsight given all points and losses.

Gradient descent. For this setting, Zinkevich (2003) showed that simple *gradient descent* (GD), combined with a smart choice for the stepsize η_t of the gradient updates, achieves a $\mathcal{O}(\sqrt{dT})$ regret with a $\mathcal{O}(d)$ space and time cost per iteration. When the only assumption on the losses is simple convexity, this upper bound matches the corresponding lower bound (Luo et al., 2016), thus making first-order methods (e.g., GD) essentially unimprovable in a minimax sense. Nonetheless, when the losses have additional curvature properties, Hazan et al. (2006) show that *online Newton step* (ONS), an adaptive method that exploits second-order (second derivative) information on the losses, can achieve a *logarithmic* regret $\mathcal{O}(d \log T)$. The downside of this adaptive method is the larger $\mathcal{O}(d^2)$ space and per-step time complexity, since second-order updates require to construct, store, and invert $\mathbf{H}_t$, a preconditioner matrix related to the Hessian of the losses used to correct the first-order updates.

Kernel gradient descent. For linear models, such as the ones considered in LOCO, a simple way to create more expressive models is to map them in some high-dimensional space, the *feature space*, and then use the *kernel trick* (Schölkopf & Smola, 2001) to avoid explicitly computing their high-dimensional representation. Mapping to a larger space allows the algorithm to better fit the losses chosen by the adversary and reduce its cumulative loss. As a drawback, the Kernel OCO (KOCO) problem[1] is fundamentally harder than LOCO, due to 1) the fact that an infinite parametrization makes regret bounds scaling with the dimension d meaningless and 2) the size of the model, and therefore time and space complexities, scales with t itself, making these methods even less performant than LOCO algorithms. Kernel extensions of LOCO algorithms have been proposed for KOCO, such as functional GD (e.g., NORMA, Kivinen et al., 2004) which achieves a $\mathcal{O}(\sqrt{T})$ regret with a $\mathcal{O}(t)$ space and time cost per iteration. For second-order methods, the Second-Order Per-

[1]SequeL team, INRIA Lille - Nord Europe. Correspondence to: Daniele Calandriello <daniele.calandriello@inria.fr>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

[1]This setting is often referred to as *online kernel learning* or *kernel-based online learning* in the literature.

ceptron (Cesa-Bianchi et al., 2005) or NAROW (Orabona & Crammer, 2010) for generic curved losses and Recursive Kernel Least Squares (Zhdanov & Kalnishkan, 2010) or Kernel AAR (Gammerman et al., 2004) for the specific case of ℓ_2 losses provide bounds that scale with the log-determinant of the kernel-matrix. As we show, this quantity is closely related to the effective dimension d_{eff}^T of the of the points $\mathbf{x}_t$, and scales as $\mathcal{O}(d_{\text{eff}}^T \log T)$, playing a similar role as the $\mathcal{O}(d \log T)$ bound from LOCO.

Approximate GD. To trade off between computational complexity $\big($smaller than $\mathcal{O}(d^2)\big)$ and improved regret (close to $\mathcal{O}(d \log T)$), several methods try approximate second-order updates, replacing $\mathbf{H}_t$ with an approximate $\widetilde{\mathbf{H}}_t$ that can be efficiently stored and inverted. AdaGrad (Duchi et al., 2011) and ADAM (Kingma & Ba, 2015) reweight the gradient updates on a per-coordinate basis using a diagonal $\widetilde{\mathbf{H}}_t$, but these methods ultimately only improve the regret dependency on d and leave the $\sqrt{T}$ component unchanged. Sketched-ONS, by Luo et al. (2016), uses matrix sketching to approximate $\mathbf{H}_t$ with a r-rank sketch $\widetilde{\mathbf{H}}_t$, that can be efficiently stored and updated in $\mathcal{O}(dr^2)$ time and space, close to the $\mathcal{O}(d)$ complexity of diagonal approximations. More importantly, Sketched-ONS achieves a much smaller regret compared to diagonal approximations: When the true $\mathbf{H}_t$ is of low-rank r, it recovers a $\mathcal{O}(r \log T)$ regret bound logarithmic in T. Unfortunately, due to the sketch approximation, a new term appears in the bound that scales with the spectra of $\mathbf{H}_t$, and in some cases can grow much larger than $\mathcal{O}(\log T)$.

Approximate kernel GD. Existing approximate GD methods for KOCO focus only on first-order updates, trying to reduce the $\mathcal{O}(t)$ per-step complexity. Budgeted methods, such as Budgeted-GD (Wang et al., 2012) and budgeted variants of the perceptron (Cavallanti et al., 2007; Dekel et al., 2008; Orabona et al., 2008) explicitly limit the size of the model, using some destructive budget maintenance procedure (e.g., removal, projection) to constrain the natural model growth over time. Alternatively, functional approximation methods in the primal (Lu et al., 2016) or dual (Le et al., 2016) use non-linear embedding techniques, such as random feature expansion (Le et al., 2013), to reduce the KOCO problem to a LOCO problem and solve it efficiently. Unfortunately, to guarantee $\mathcal{O}(\sqrt{T})$ regret using less than $\mathcal{O}(t)$ space and time per round w.h.p., all of these methods require additional assumptions, such as points $\mathbf{x}_t$ coming from a distribution or strong convexity on the losses. Moreover, as approximate first-order methods, they can at most hope to match the $\mathcal{O}(\sqrt{T})$ regret of exact GD, and among second-order kernel methods, no approximation scheme has been proposed that can provably maintain the same $\mathcal{O}(\log T)$ regret as exact GD. In addition, approximating $\mathbf{H}_t$ is harder for KOCO, since we cannot directly access the matrix representation of $\mathbf{H}_t$ in the feature-space, making diagonal approximation impossible, and low-rank sketching harder.

Contributions In this paper, we introduce Kernel-ONS, an extension to KOCO of the ONS algorithm. As a second-order method, KONS achieves a $\mathcal{O}(d_{\text{eff}}^t \log T)$ regret on a variety of curved losses, and runs in $\mathcal{O}(t^2)$ time and space. To alleviate the computational complexity, we propose SKETCHED-KONS, the first approximate second-order KOCO methods, that approximates the kernel matrix with a low-rank sketch. To compute this sketch we propose a new online kernel dictionary learning, *kernel online row sampling*, based on *ridge leverage scores*. By adaptively increasing the size of its sketch, SKETCHED-KONS provides a favorable regret-performance trade-off, where for a given factor $\gamma \leq 1$, we can increase the regret by a linear $1/\gamma$ factor to $\mathcal{O}(d_{\text{eff}}^t \log(T)/\gamma)$ while obtaining a quadratic γ^2 improvement in runtime, thereby achieving $\mathcal{O}(t^2\gamma^2)$ space and time cost per iteration.

2. Background

In this section, we introduce linear algebra and RKHS notation, and formally state the OCO problem in an RKHS (Schölkopf & Smola, 2001).

Notation. We use upper-case bold letters $\mathbf{A}$ for matrices, lower-case bold letters $\mathbf{a}$ for vectors, lower-case letters a for scalars. We denote by $[\mathbf{A}]_{ij}$ and $[\mathbf{a}]_i$ the (i, j) element of a matrix and i-th element of a vector respectively. We denote by $\mathbf{I}_T \in \mathbb{R}^{T\times T}$, the identity matrix of dimension T and by $\operatorname{Diag}(\mathbf{a}) \in \mathbb{R}^{T\times T}$, the diagonal matrix with the vector $\mathbf{a} \in \mathbb{R}^T$ on the diagonal. We use $\mathbf{e}_{T,i} \in \mathbb{R}^T$ to denote the indicator vector of dimension T for element i. When the dimension of $\mathbf{I}$ and $\mathbf{e}_i$ is clear from the context, we omit the T. We also indicate with $\mathbf{I}$ the identity operator. We use $\mathbf{A} \succeq \mathbf{B}$ to indicate that $\mathbf{A} - \mathbf{B}$ is a positive semi-definite (PSD) matrix. With $\|\cdot\|$ we indicate the *operator* ℓ_2-norm. Finally, the set of integers between 1 and T is denoted by $[T] := \{1, \ldots, T\}$.

Kernels. Given an arbitrary input space $\mathcal{X}$ and a positive definite kernel function $\mathcal{K} : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, we indicate the *reproducing kernel Hilbert space* (RKHS) associated with $\mathcal{K}$ as $\mathcal{H}$. We choose to represent our Hilbert space $\mathcal{H}$ as a feature space where, given $\mathcal{K}$, we can find an associated feature map $\varphi : \mathcal{X} \to \mathcal{H}$, such that $\mathcal{K}(\mathbf{x}, \mathbf{x}')$ can be expressed as an inner product $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle\varphi(\mathbf{x}), \varphi(\mathbf{x}')\rangle_{\mathcal{H}}$. With a slight abuse of notation, we represent our feature space as an high-dimensional vector space, or in other words $\mathcal{H} \subseteq \mathbb{R}^D$, where D is very large or potentially infinite. With this notation, we can write the inner product simply as $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^\mathsf{T}\varphi(\mathbf{x}')$, and for any function $f_\mathbf{w} \in \mathcal{H}$, we can represent it as a (potentially infinite) set of weights $\mathbf{w}$ such that $f_\mathbf{w}(\mathbf{x}) = \varphi(\mathbf{x})^\mathsf{T}\mathbf{w}$. Given points $\{\mathbf{x}_i\}_{i=1}^t$, we shorten $\varphi(\mathbf{x}_i) = \boldsymbol{\phi}_i$ and define the feature matrix $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_t] \in \mathbb{R}^{D\times t}$. Finally, to denote the inner product between two arbitrary subsets a and b of columns of $\boldsymbol{\Phi}_T$ we use $\mathbf{K}_{a,b} = \boldsymbol{\Phi}_a^\mathsf{T}\boldsymbol{\Phi}_b$. With this notation, we can write the empirical kernel matrix as $\mathbf{K}_t = \mathbf{K}_{[t],[t]} = \boldsymbol{\Phi}_t^\mathsf{T}\boldsymbol{\Phi}_t$, the vector with all the similarities between a new point and the old ones as $\mathbf{k}_{[t-1],t} = \boldsymbol{\Phi}_{t-1}^\mathsf{T}\boldsymbol{\phi}_t$, and the kernel evaluated at a specific point as $k_{t,t} = \boldsymbol{\phi}_t^\mathsf{T}\boldsymbol{\phi}_t$. Throughout the rest of the paper, we assume that $\mathcal{K}$ is normalized and $\boldsymbol{\phi}_t^\mathsf{T}\boldsymbol{\phi}_t = 1$.

Kernelized online convex optimization. In the general OCO framework with linear prediction, the optimization process is a game where at each time step $t \in [T]$ the player

1 receives an input $\mathbf{x}_t \in \mathcal{X}$ from the adversary,

2 predicts $\widehat{y}_t = f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{w}_t = \boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t$,

3 incurs loss $\ell_t(\widehat{y}_t)$, with ℓ_t a convex and differentiable function chosen by the adversary,

4 observes the derivative $\dot{g}_t = \ell_t'(\widehat{y}_t)$.

Since the player uses a linear combination $\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t$ to compute $\widehat{y}_t$, having observed $\dot{g}_t$, we can compute the gradient,

$$\mathbf{g}_t = \nabla\ell_t(\widehat{y}_t) = \dot{g}_t\nabla(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_{t-1}) = \dot{g}_t\boldsymbol{\phi}_t.$$

After t timesteps, we indicate with $\mathcal{D}_t = \{\mathbf{x}_i\}_{i=1}^t$, the dataset containing the points observed so far. In the rest of the paper we consider the problem of *kernelized OCO* (KOCO) where $\mathcal{H}$ is arbitrary and potentially non-parametric. We refer to the special parametric case $\mathcal{H} = \mathbb{R}^d$ and $\boldsymbol{\phi}_t = \mathbf{x}_t$ as *linear OCO* (LOCO).

In OCO, the goal is to design an algorithm that returns a solution that performs almost as well as the best-in-class, thus we must first define our comparison class. We define the feasible set as $\mathcal{S}_t = \{\mathbf{w} : |\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}| \le C\}$ and $\mathcal{S} = \cap_{t=1}^T \mathcal{S}_t$. This comparison class contains all functions f_w whose output is contained (clipped) in the interval $[-C, C]$ on all points $x_1, \dots, x_T$. Unlike the often used constraint on $\|\mathbf{w}\|_\mathcal{H}$ (Hazan et al., 2006; Zhu & Xu, 2015), comparing against clipped functions (Luo et al., 2016; Gammerman et al., 2004; Zhdanov & Kalnishkan, 2010) has a clear interpretation even when passing from $\mathbb{R}^d$ to $\mathcal{H}$. Moreover, $\mathcal{S}$ is invariant to linear transformations of $\mathcal{H}$ and suitable for practical problems where it is often easier to choose a reasonable interval for the predictions $\widehat{y}_t$ rather than a bound on the norm of a (possibly non-interpretable) parametrization $\mathbf{w}$. We can now define the regret as

$$R_T(\mathbf{w}) = \sum\nolimits_{t=1}^T \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}_t) - \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w})$$

and denote with $R_T = R_T(\mathbf{w}^*)$, the regret w.r.t. $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{S}} \sum_{t=1}^T \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w})$, i.e., the best fixed function in $\mathcal{S}$. We work with the following assumptions on the losses.

Algorithm 1 One-shot KONS

```
Input: Feasible parameter $C$, stepsizes $\eta_t$, regulariz. $\alpha$
1: Initialize $\mathbf{w}_0 = \mathbf{0}, \mathbf{g}_0 = \mathbf{0}, b_0 = 0, \mathbf{A}_0 = \alpha\mathbf{I}$
2: for $t = \{1, \dots, T\}$ do
3:     receive $\mathbf{x}_t$
4:     compute $b_s$ as in Lem. 2
5:     compute $\mathbf{u}_t = \mathbf{A}_{t-1}^{-1}(\sum_{s=0}^{t-1} b_s\mathbf{g}_s)$
6:     compute $\overline{y}_t = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{u}_t$
7:     predict $\widehat{y}_t = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{w}_t = \overline{y}_t - h(\overline{y}_t)$
8:     observe $\mathbf{g}_t$, update $\mathbf{A}_t = \mathbf{A}_{t-1} + \eta_t\mathbf{g}_t\mathbf{g}_t^\mathsf{T}$
9: end for
```

Assumption 1. *The loss function ℓ_t satisfies $|\ell_t'(y)| \le L$ whenever $y \le C$.*

Note that this is equivalent to assuming Lipschitzness of the the loss w.r.t. y and it is weaker than assuming something on the norm of the gradient $\|\mathbf{g}_t\|$, since $\|\mathbf{g}_t\| = |\dot{g}_t|\|\boldsymbol{\phi}_t\|$.

Assumption 2. *There exists $\sigma_t \ge 0$ such that for all $\mathbf{u}, \mathbf{w} \in \mathcal{S}$, $l_t(\mathbf{w}) = \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w})$ is lower-bounded by*

$$l_t(\mathbf{w}) \ge l_t(\mathbf{u}) + \nabla l_t(\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \frac{\sigma_t}{2}(\nabla l_t(\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}))^2.$$

This condition is weaker than strong convexity and it is satisfied by all exp-concave losses (Hazan et al., 2006). For example, the squared loss $l_t(\mathbf{w}) = (y_t - \mathbf{x}_t^\mathsf{T}\mathbf{w})^2$ is not strongly convex but satisfies Asm. 2 with $\sigma_t = 1/(8C^2)$ when $\mathbf{w} \in \mathcal{S}$.

3. Kernelized Online Newton Step

The online Newton step algorithm, originally introduced by Hazan et al. (2006), is a projected gradient descent that uses the following update rules

$$\mathbf{u}_t = \mathbf{w}_{t-1} - \mathbf{A}_{t-1}^{-1}\mathbf{g}_{t-1},$$
$$\mathbf{w}_t = \Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(\mathbf{u}_t),$$

where $\Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(u_t) = \arg\min_{\mathbf{w}\in\mathcal{S}_t} \|\mathbf{u}_t - \mathbf{w}\|_{\mathbf{A}_{t-1}}$ is an oblique projection on a set $\mathcal{S}_t$ with matrix $\mathbf{A}_{t-1}$. If $\mathcal{S}_t$ is the set of vectors with bounded prediction in $[-C, C]$ as by Luo et al. (2016), then the projection reduces to

$$\mathbf{w}_t = \Pi_{\mathcal{S}_t}^{\mathbf{A}_{t-1}}(\mathbf{u}_t) = \mathbf{u}_t - \frac{h(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{u}_t)}{\boldsymbol{\phi}_t^\mathsf{T}\mathbf{A}_{t-1}^{-1}\boldsymbol{\phi}_t}\mathbf{A}_{t-1}^{-1}\boldsymbol{\phi}_t, \qquad (1)$$

where $h(z) = \operatorname{sign}(z)\max\{|z| - C,\ 0\}$ computes how much z is above or below the interval $[-C, C]$. When $\mathbf{A}_t = \mathbf{I}/\eta_t$, ONS is equivalent to vanilla projected gradient descent, which in LOCO achieves $\mathcal{O}(\sqrt{dT})$ regret (Zinkevich, 2003). In the same setting, Hazan et al. (2006) shows that choosing $\mathbf{A}_t = \sum_{s=1}^t \eta_s\mathbf{g}_s\mathbf{g}_s^\mathsf{T} + \alpha\mathbf{I}$ makes ONS an efficient reformulation of *follow the approximate* *leader* (FTAL). While traditional follow-the-leader algorithms play the weight $\mathbf{w}_t = \arg\min_{\mathbf{w}\in\mathcal{S}_t}\sum_{s=1}^{t-1} l_t(\mathbf{w})$, FTAL replaces the loss l_t with a convex approximation using Asm. 2, and plays the minimizer of the surrogate function. As a result, under Asm. 1-2 and when $\sigma_t \geq \sigma > 0$, FTAL achieves a logarithmic $\mathcal{O}(d\log T)$ regret. FTAL's solution path can be computed in $\mathcal{O}(d^2)$ time using ONS updates, and further speedups were proposed by Luo et al. (2016) using matrix sketching.

Unfortunately, in KOCO, vectors ϕ_t and weights $\mathbf{w}_t$ cannot be explicitly represented, and most of the quantities used in vanilla ONS (Eq. 1) cannot be directly computed. Instead, we derive a closed form alternative (Alg. 1) that can be computed in practice. Using a rescaled variant of our feature vectors ϕ_t, $\overline{\phi}_t = \dot{g}_t\sqrt{\eta_t}\phi_t = \sqrt{\eta_t}\mathbf{g}_t$ and $\overline{\mathbf{\Phi}}_t = [\overline{\phi}_1,\dots,\overline{\phi}_t]$, we can rewrite $\mathbf{A}_t = \overline{\mathbf{\Phi}}_t\overline{\mathbf{\Phi}}_t^{\mathsf{T}} + \alpha\mathbf{I}$ and $\overline{\mathbf{\Phi}}_t^{\mathsf{T}}\overline{\mathbf{\Phi}}_t = \overline{\mathbf{K}}_t$, where the empirical kernel matrix $\overline{\mathbf{K}}_t$ is computed using the rescaled kernel $\overline{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_j) = \dot{g}_i\sqrt{\eta_i}\dot{g}_j\sqrt{\eta_j}\mathcal{K}(\mathbf{x}_i,\mathbf{x}_j)$ instead of the original $\mathcal{K}$, or equivalently $\overline{\mathbf{K}}_t = \mathbf{D}_t\mathbf{K}_t\mathbf{D}_t$ with $\mathbf{D}_t = \mathrm{Diag}(\{\dot{g}_i\sqrt{\eta_i}\}_{i=1}^t)$ the rescaling diagonal matrix. We begin by noting that

$$\begin{aligned}\widehat{y}_t = \phi_t^{\mathsf{T}}\mathbf{w}_t &= \phi_t^{\mathsf{T}}\left(\mathbf{u}_t - \frac{h(\phi_t^{\mathsf{T}}\mathbf{u}_t)}{\phi_t^{\mathsf{T}}\mathbf{A}_{t-1}^{-1}\phi_t}\mathbf{A}_{t-1}^{-1}\phi_t\right)\\ &= \phi_t^{\mathsf{T}}\mathbf{u}_t - h(\phi_t^{\mathsf{T}}\mathbf{u}_t)\frac{\phi_t^{\mathsf{T}}\mathbf{A}_{t-1}^{-1}\phi_t}{\phi_t^{\mathsf{T}}\mathbf{A}_{t-1}^{-1}\phi_t} = \overline{y}_t - h(\overline{y}_t).\end{aligned}$$

As a consequence, if we can find a way to compute $\overline{y}_t$, then we can obtain $\widehat{y}_t$ without explicitly computing $\mathbf{w}_t$. Before that, we first derive a non-recursive formulation of $\mathbf{u}_t$.

Lemma 1. *In Alg. 1 we introduce*

$$b_i = [\mathbf{b}_t]_i = \dot{g}_i\sqrt{\eta_i}\left(\widehat{y}_i - \frac{h(\overline{y}_i)}{\overline{\phi}_i^{\mathsf{T}}\mathbf{A}_{i-1}^{-1}\overline{\phi}_i}\right) - \frac{1}{\sqrt{\eta_i}}$$

and compute $\mathbf{u}_t$ as

$$\mathbf{u}_t = \mathbf{A}_{t-1}^{-1}\overline{\mathbf{\Phi}}_{t-1}\mathbf{b}_{t-1}.$$

Then, $\mathbf{u}_t$ is equal to the same quantity in Eq. 1 and the sequence of predictions $\widehat{y}_t$ is the same in both algorithms.

While the definition of $\mathbf{b}_t$ and $\mathbf{u}_t$ still requires performing operations in the (possibly infinitely dimensional) feature space, in the following we show that $\mathbf{b}_t$ and the prediction $\overline{y}_t$ can be conveniently computed using only inner products.

Lemma 2. *All the components $b_i = [\mathbf{b}_t]_i$ of the vector introduced in Lem. 1 can be computed as*

$$\dot{g}_i\sqrt{\eta_i}\left(\widehat{y}_i - \frac{\alpha h(\overline{y}_i)}{k_{i,i} - \overline{\mathbf{k}}_{[i-1],i}^{\mathsf{T}}(\overline{\mathbf{K}}_{i-1}+\alpha\mathbf{I})^{-1}\overline{\mathbf{k}}_{[i-1],i}} - \frac{1}{\eta_i}\right).$$

Then, we can compute

$$\overline{y}_t = \frac{1}{\alpha}\mathbf{k}_{[t-1],t}^{\mathsf{T}}\mathbf{D}_{t-1}(\mathbf{b}_{t-1} - (\overline{\mathbf{K}}_{t-1}+\alpha\mathbf{I})^{-1}\overline{\mathbf{K}}_{t-1}\mathbf{b}_{t-1}).$$

Since Alg. 1 is equivalent to ONS (Eq. 1), existing regret bounds for ONS directly applies to its kernelized version.

Proposition 1 (Luo et al., 2016)**.** *For any sequence of losses ℓ_t satisfying Asm. 1-2, the regret R_T of Alg. 1 is bounded by $R_T \leq \alpha\|\mathbf{w}^*\|^2 + R_G + R_D$ with*

$$\begin{aligned}R_G &:= \sum_{t=1}^T \mathbf{g}_t^{\mathsf{T}}\mathbf{A}_t^{-1}\mathbf{g}_t = \sum_{t=1}^T \overline{\phi}_t^{\mathsf{T}}(\overline{\mathbf{\Phi}}_t\overline{\mathbf{\Phi}}_t^{\mathsf{T}}+\alpha\mathbf{I})^{-1}\overline{\phi}_t/\eta_t\\ R_D &:= \sum_{t=1}^T(\mathbf{w}_t-\mathbf{w}^*)^{\mathsf{T}}(\mathbf{A}_t - \mathbf{A}_{t-1} - \sigma_t\mathbf{g}_t\mathbf{g}_t^{\mathsf{T}})(\mathbf{w}_t-\mathbf{w}^*)\\ &= \sum_{t=1}^T(\eta_t-\sigma_t)\dot{g}_t^2(\phi_t^{\mathsf{T}}(\mathbf{w}_t-\mathbf{w}^*))^2.\end{aligned}$$

In the d-dimensional LOCO, choosing a decreasing step-size $\eta_t = \sqrt{d/(C^2L^2t)}$ allows ONS to achieve a $\mathcal{O}(CL\sqrt{dT})$ regret for the cases where $\sigma_t = 0$. When $\sigma_t \geq \sigma > 0$ (e.g., when the functions are exp-concave) we can set $\eta_t = \sigma_t$ and improve the regret to $\mathcal{O}(d\log(T))$. Unfortunately, these quantities hold little meaning for KOCO with D-dimensional features, since a $\mathcal{O}(\sqrt{D})$ regret can be very large or even infinite. On the other hand, we expect the regret of KONS to depend on quantities that are more strictly related to the kernel $\overline{\mathbf{K}}_t$ and its complexity.

Definition 1. *Given a kernel function $\mathcal{K}$, a set of points $\mathcal{D}_t = \{\mathbf{x}_i\}_{i=1}^t$ and a parameter $\alpha > 0$, we define the α-ridge leverage scores (RLS) of point i as*

$$\tau_{t,i} = \mathbf{e}_{t,i}^{\mathsf{T}}\mathbf{K}_t^{\mathsf{T}}(\mathbf{K}_t+\alpha\mathbf{I})^{-1}\mathbf{e}_{t,i} = \phi_i^{\mathsf{T}}(\mathbf{\Phi}_t\mathbf{\Phi}_t^{\mathsf{T}}+\alpha\mathbf{I})^{-1}\phi_i, \quad (2)$$

and the effective dimension of $\mathcal{D}_t$ as

$$d_{\mathit{eff}}^t(\alpha) = \sum_{i=1}^t \tau_{t,i} = \mathrm{Tr}\left(\mathbf{K}_t(\mathbf{K}_t+\alpha\mathbf{I}_t)^{-1}\right). \quad (3)$$

In general, leverage scores have been used to measure the correlation between a point i w.r.t. the other $t-1$ points, and therefore how essential it is in characterizing the dataset (Alaoui & Mahoney, 2015). As an example, if ϕ_i is completely orthogonal to the other points, $\tau_{t,i} = \phi_i^{\mathsf{T}}(\phi_i\phi_i^{\mathsf{T}}+\alpha\mathbf{I})^{-1}\phi_i \leq 1/(1+\alpha)$ and its RLS is maximized, while in the case where all the points $\mathbf{x}_i$ are identical, $\tau_{t,i} = \phi_i^{\mathsf{T}}(t\phi_i\phi_i^{\mathsf{T}}+\alpha\mathbf{I})^{-1}\phi_i \leq 1/(t+\alpha)$ and its RLS is minimal. While the previous definition is provided for a generic kernel function $\mathcal{K}$, we can easily instantiate it on $\overline{\mathcal{K}}$ and obtain the definition of $\overline{\tau}_{t,i}$. By recalling the first regret term in the decomposition of Prop. 1, we notice that

$$R_G = \sum_{t=1}^T \overline{\phi}_t^{\mathsf{T}}(\overline{\mathbf{\Phi}}_t\overline{\mathbf{\Phi}}_t^{\mathsf{T}}+\alpha\mathbf{I})^{-1}\overline{\phi}_t/\eta_t = \sum_{t=1}^T \overline{\tau}_{t,t}/\eta_t,$$

which reveals a deep connection between the regret of KONS and the cumulative sum of the RLS. In other words, the RLS capture how much the adversary can increase the regret by picking orthogonal directions that have not been seen before. While in LOCO, this can happen at most d times (hence the dependency on d in the final regret, which is mitigated by a suitable choice of η_t), in KOCO, R_G can grow linearly with time, since large $\mathcal{H}$ can have infinite near-orthogonal directions. Nonetheless, the actual growth rate is now directly related to the complexity of the sequence of points chosen by the adversary and the kernel function $\mathcal{K}$. While the effective dimension $d_{\text{eff}}^t(\alpha)$ is related to the capacity of the RKHS $\mathcal{H}$ on the points in $\mathcal{D}_t$ and it has been shown to characterize the generalization error in batch linear regression (Rudi et al., 2015), we see that R_G is rather related to the *online* effective dimension $\overline{d}_{\text{onl}}^t(\alpha) = \sum_i \overline{\tau}_{i,i}$. Nonetheless, we show that the two quantities are also strictly related to each other.

Lemma 3. *For any dataset $\mathcal{D}_T$, any $\alpha > 0$ we have*

$$\overline{d}_{onl}^T(\alpha) := \sum_{t=1}^T \overline{\tau}_{t,t} \leq \log(\operatorname{Det}(\overline{\mathbf{K}}_T/\alpha + \mathbf{I}))$$
$$\leq \overline{d}_{eff}^T(\alpha)(1 + \log(\|\overline{\mathbf{K}}_T\|/\alpha + 1)).$$

We first notice that in the first inequality we relate $\overline{d}_{\text{onl}}^T(\alpha)$ to the log-determinant of the kernel matrix $\overline{\mathbf{K}}_T$. This quantity appears in a large number of works on online linear prediction (Cesa-Bianchi et al., 2005; Srinivas et al., 2010) where they were connected to the maximal mutual information gain in Gaussian processes. Finally, the second inequality shows that in general the complexity of online learning is only a factor $\log T$ (in the worst case) away from the complexity of batch learning. At this point, we can generalize the regret bounds of LOCO to KOCO.

Theorem 1. *For any sequence of losses ℓ_t satisfying Asm. 1-2, let $\sigma = \min_t \sigma_t$. If $\eta_t \geq \sigma \geq 0$ for all t and $\alpha \leq \sqrt{T}$, the regret of Alg. 1 is upper-bounded as*

$$R_T \leq \alpha\|\mathbf{w}^*\|^2 + d_{onl}^T(\alpha)/\eta_T + 4C^2L^2\sum_{t=1}^T(\eta_t - \sigma).$$

In particular, if for all t we have $\sigma_t \geq \sigma > 0$, setting $\eta_t = \sigma$ we obtain

$$R_T \leq \alpha\|\mathbf{w}^*\|^2 + 2d_{eff}^T\left(\alpha/(\sigma L^2)\right)\frac{\log(2\sigma L^2 T)}{\sigma},$$

otherwise, $\sigma = 0$ and setting $\eta_t = 1/(LC\sqrt{t})$ we obtain

$$R_T \leq \alpha\|\mathbf{w}^*\|^2 + 4LC\sqrt{T}d_{eff}^T(\alpha/L^2)\log(2L^2T).$$

Comparison to LOCO algorithms. We first notice that the effective dimension $d_{\text{eff}}^T(\alpha)$ can be seen as a soft rank for $\overline{\mathbf{K}}_T$ and that it is smaller than the rank r for any α.[2] For exp-concave functions (i.e., $\sigma > 0$), we slightly improve over the bound of Luo et al. (2016) from $\mathcal{O}(d\log T)$ down to $\mathcal{O}(d_{\text{eff}}^T(\alpha)\log T) \leq \mathcal{O}(r\log T)$, where r is the (unknown) rank of the dataset. Furthermore, when $\sigma = 0$, setting $\eta_t = \sqrt{1/(L^2C^2t)}$ gives us a regret $\mathcal{O}(\sqrt{T}d_{\text{eff}}^T(\alpha)) \leq \mathcal{O}(\sqrt{T}r)$, which is potentially much smaller than $\mathcal{O}(\sqrt{T}d)$. Furthermore, if an oracle provided us in advance with $d_{\text{eff}}^T(\alpha)$, setting $\eta_t = \sqrt{d_{\text{eff}}^T(\alpha)/(L^2C^2t)}$ gives a regret $\mathcal{O}(\sqrt{d_{\text{eff}}^T(\alpha)T}) \leq \mathcal{O}(\sqrt{rT})$.

Comparison to KOCO algorithms. Simple functional gradient descent (e.g., NORMA, Kivinen et al., 2004) achieves a $\mathcal{O}(\sqrt{T})$ regret when properly tuned (Zhu & Xu, 2015), regardless of the loss function. For the special case of squared loss, Zhdanov & Kalnishkan (2010) show that Kernel Ridge Regression achieves the same $\mathcal{O}(\log(\operatorname{Det}(\overline{\mathbf{K}}_T/\alpha + \mathbf{I})))$ regret as achieved by KONS for general exp-concave losses.

Algorithm 2 Kernel Online Row Sampling (KORS)

```
Input: Regularization $\alpha$, accuracy $\varepsilon$, budget $\beta$
1: Initialize $\mathcal{I}_0 = \emptyset$
2: for $t = \{0, \dots, T-1\}$ do
3:     receive $\overline{\phi}_t$
4:     construct temporary dictionary $\overline{\mathcal{I}}_t := \mathcal{I}_{t-1} \cup (t, 1)$
5:     compute $\widetilde{p}_t = \min\{\beta\widetilde{\tau}_{t,t}, 1\}$ using $\overline{\mathcal{I}}_t$ and Eq. 4
6:     draw $z_t \sim \mathcal{B}(\widetilde{p}_t)$ and if $z_t = 1$, add $(t, 1/\widetilde{p}_t)$ to $\mathcal{I}_t$
7: end for
```

4. Kernel Online Row Sampling

Although KONS achieves a low regret, storing and inverting the $\overline{\mathbf{K}}$ matrix requires $\mathcal{O}(t^2)$ space and $\mathcal{O}(t^3)$ time, which becomes quickly unfeasible as t grows. To improve space and time efficiency, we replace $\overline{\mathbf{K}}_t$ with an accurate low-rank approximation $\widetilde{\mathbf{K}}_t$, constructed using a carefully chosen dictionary $\mathcal{I}_t$ of points from $\mathcal{D}_t$. We extend the *online row sampling* (ORS) algorithm of Cohen et al. (2016) to the kernel setting and obtain Kernel-ORS (Alg. 2). There are two main obstacles to overcome in the adaptation of ORS: From an algorithmic perspective we need to find a computable estimator for the RLS, since ϕ_t cannot be accessed directly, while from an analysis perspective we must prove that our space and time complexity does not scale with the dimension of ϕ_t (as Cohen et al. 2016), as it can potentially be infinite.

We define a dictionary $\mathcal{I}_t$ as a collection of *(index, weight)* tuples $(i, 1/\widetilde{p}_i)$ and the associated selection matrix $\mathbf{S}_t \in$

[2]This can be easily seen as $d_{\text{eff}}^T(\alpha) = \sum_t \lambda_t/(\lambda_t + \alpha)$, where λ_t are the eigenvalues of $\overline{\mathbf{K}}_T$.

$\mathbb{R}^{t\times t}$ as a diagonal matrix with $1/\sqrt{\widetilde{p}_i}$ for all $i \in \mathcal{I}_t$ and 0 elsewhere. We also introduce $\mathbf{A}_t^{\mathcal{I}_t} = \overline{\mathbf{\Phi}}_t\mathbf{S}_t\mathbf{S}_t^\mathsf{T}\overline{\mathbf{\Phi}}_t^\mathsf{T} + \alpha\mathbf{I}$ as an approximation of $\mathbf{A}_t$ constructed using the dictionary $\mathcal{I}_t$. At each time step, KORS temporarily adds t with weight 1 to the dictionary $\mathcal{I}_{t-1}$ and constructs the temporary dictionary $\mathcal{I}_{t,*}$ and the corresponding selection matrix $\mathbf{S}_{t,*}$ and approximation $\mathbf{A}_t^{\mathcal{I}_{t,*}}$. This augmented dictionary can be effectively used to compute the RLS estimator,

$$\begin{aligned}\widetilde{\tau}_{t,i} &= (1+\varepsilon)\overline{\phi}_t\big(\mathbf{A}_t^{\mathcal{I}_{t,*}}\big)^{-1}\overline{\phi}_t \qquad (4)\\ &= \tfrac{1+\varepsilon}{\alpha}\big(\overline{k}_{t,t} - \overline{\mathbf{k}}_{[t],t}^\mathsf{T}\mathbf{S}_{t,*}(\mathbf{S}_{t,*}^\mathsf{T}\overline{\mathbf{K}}_t\mathbf{S}_{t,*} + \alpha\mathbf{I})^{-1}\mathbf{S}_{t,*}^\mathsf{T}\overline{\mathbf{k}}_{[t],t}\big).\end{aligned}$$

While we introduced a similar estimator before (Calandriello et al., 2017), here we modified it so that $\widetilde{\tau}_{t,i}$ is an overestimate of the actual $\overline{\tau}_{t,i}$. Note that all rows and columns for which $\mathbf{S}_{t,*}$ is zero (all points outside the temporary dictionary $\mathcal{I}_{t,*}$) do not influence the estimator, so they can be excluded from the computation. As a consequence, denoting by $|\mathcal{I}_{t,*}|$ the size of the dictionary, $\widetilde{\tau}_{t,i}$ can be efficiently computed in $\mathcal{O}(|\mathcal{I}_{t,*}|^2)$ space and $\mathcal{O}(|\mathcal{I}_{t,*}|^2)$ time (using an incremental update of Eq. 4). After computing the RLS, KORS randomly chooses whether to include a point in the dictionary using a coin-flip with probability $\widetilde{p}_t = \min\{\beta\widetilde{\tau}_{t,t}, 1\}$ and weight $1/\widetilde{p}_t$, where β is a parameter. The following theorem gives us at each step guarantees on the accuracy of the approximate matrices $\mathbf{A}_t^{\mathcal{I}_t}$ and of estimates $\widetilde{\tau}_{t,t}$, as well as on the size $|\mathcal{I}_t|$ of the dictionary.

Theorem 2. *Given parameters $0 < \varepsilon \leq 1$, $0 < \alpha$, $0 < \delta < 1$, let $\rho = \frac{1+\varepsilon}{1-\varepsilon}$ and run Algorithm 2 with $\beta \geq 3\log(T/\delta)/\varepsilon^2$. Then w.p. $1-\delta$, for all steps $t \in [T]$,*

(1) $(1-\varepsilon)\mathbf{A}_t \preceq \mathbf{A}_t^{\mathcal{I}_t} \preceq (1+\varepsilon)\mathbf{A}_t$.

(2) The dictionary's size $|\mathcal{I}_t| = \sum_{s=1}^t z_s$ is bounded by

$$\sum_{s=1}^t z_s \leq 3\sum_{s=1}^t \widetilde{p}_s \leq d_{onl}^t(\alpha)\frac{3\rho\beta}{\varepsilon^2} \leq d_{eff}^t(\alpha)\frac{6\rho\log^2\left(\frac{2T}{\delta}\right)}{\varepsilon^2}.$$

(3) Satisfies $\tau_{t,t} \leq \widetilde{\tau}_{t,t} \leq \rho\tau_{t,t}$.

Moreover, the algorithm runs in $\mathcal{O}(d_{eff}^t(\alpha)^2\log^4(T))$ space, and $\widetilde{\mathcal{O}}(d_{eff}^t(\alpha)^2)$ time per iteration.

The most interesting aspect of this result is that the dictionary $\mathcal{I}_t$ generated by KORS allows to accurately approximate the $\mathbf{A}_t = \overline{\mathbf{\Phi}}_t\overline{\mathbf{\Phi}}_t^\mathsf{T} + \alpha\mathbf{I}$ matrix up to a small $(1\pm\varepsilon)$ multiplicative factor with a small time and space complexity, which makes it a natural candidate to sketch KONS.

5. Sketched ONS

Building on KORS, we now introduce a sketched variant of KONS that can efficiently trade off between computational performance and regret. Alg. 3 runs KORS as a black-box estimating RLS $\widetilde{\tau}_t$, that are then used to sketch the original matrix $\mathbf{A}_t$ with a matrix $\widetilde{\mathbf{A}}_t = \sum_{s=1}^t \eta_t z_t \mathbf{g}_t\mathbf{g}_t^\mathsf{T}$, where at each step we add the current gradient $\mathbf{g}_t\mathbf{g}_t^\mathsf{T}$ only if the coin flip z_t succeeded. Unlike KORS, the elements added to $\widetilde{\mathbf{A}}_t$ are not weighted, and the probabilities $\widetilde{p}_t$ used for the coins z_t are chosen as the maximum between $\widetilde{\tau}_{t,t}$, and a parameter $0 \leq \gamma \leq 1$. Let $\mathbf{R}_t$ be the unweighted counterpart of $\mathbf{S}_t$, that is $[\mathbf{R}_t]_{i,j} = 0$ if $[\mathbf{S}_t]_{i,j} = 0$ and $[\mathbf{R}_t]_{i,j} = 1$ if $[\mathbf{S}_t]_{i,j} \neq 0$. Then we can efficiently compute the coefficients $\widetilde{b}_t$ and predictions $\widetilde{y}_t$ as follows.

Algorithm 3 SKETCHED-KONS

```
Input: Feasible parameter $C$, stepsizes $\eta_t$, regulariz. $\alpha$
 1: Initialize $\mathbf{w}_0 = \mathbf{0}, \mathbf{g}_0 = \mathbf{0}, b_0 = 0, \widetilde{\mathbf{A}}_0 = \alpha\mathbf{I}$
 2: Initialize independent run of KORS
 3: for $t = \{1, \ldots, T\}$ do
 4:     receive $\mathbf{x}_t$
 5:     compute $\widetilde{\mathbf{u}}_t = \widetilde{\mathbf{A}}_{t-1}^{-1}(\sum_{s=0}^{t-1}\widetilde{b}_s\mathbf{g}_s)$
 6:     compute $\breve{y}_t = \varphi(\mathbf{x}_t)^\mathsf{T}\widetilde{\mathbf{u}}_t$
 7:     predict $\widetilde{y}_t = \varphi(\mathbf{x}_t)^\mathsf{T}\widetilde{\mathbf{w}}_t = \breve{y}_t - h(\breve{y}_t)$, observe $\mathbf{g}_t$
 8:     compute $\widetilde{\tau}_{t,t}$ using KORS (Eq. 4)
 9:     compute $\widetilde{p}_t = \max\{\min\{\beta\widetilde{\tau}_{t,t}, 1\}, \gamma\}$
10:     draw $z_t \sim \mathcal{B}(\widetilde{p}_t)$
11:     update $\widetilde{\mathbf{A}}_t = \widetilde{\mathbf{A}}_{t-1} + \eta_t z_t \mathbf{g}_t\mathbf{g}_t^\mathsf{T}$
12: end for
```

Lemma 4. *Let $\mathbf{E}_t = \mathbf{R}_t^\mathsf{T}\overline{\mathbf{K}}_t\mathbf{R}_t + \alpha\mathbf{I}$ be an auxiliary matrix, then all the components $\widetilde{b}_i = [\widetilde{\mathbf{b}}_t]_i$ used in Alg. 3 can be computed as*

$$\dot{g}_i\sqrt{\eta_i}\left(\widetilde{y}_i - \frac{\alpha h(\breve{y}_i)}{k_{i,i} - \overline{\mathbf{k}}_{[i-1],i}^\mathsf{T}\mathbf{R}_{i-1}\mathbf{E}_{i-1}^{-1}\mathbf{R}_{i-1}\overline{\mathbf{k}}_{[i-1],i}} - \frac{1}{\eta_i}\right).$$

Then we can compute

$$\begin{aligned}\breve{y}_t = \frac{1}{\alpha}\big(&\mathbf{k}_{[t-1],t}^\mathsf{T}\mathbf{D}_{t-1}\mathbf{b}_{t-1}\\ &- \mathbf{k}_{[t-1],t}^\mathsf{T}\mathbf{D}_{t-1}\mathbf{R}_{t-1}\mathbf{E}_{t-1}^{-1}\mathbf{R}_{t-1}\overline{\mathbf{K}}_{t-1}\mathbf{b}_{t-1}\big).\end{aligned}$$

Note that since the columns in $\mathbf{R}_t$ are selected without weights, $(\mathbf{R}_t^\mathsf{T}\overline{\mathbf{K}}_t\mathbf{R}_t + \alpha\mathbf{I})^{-1}$ can be updated efficiently using block inverse updates, and only when $\widetilde{\mathbf{A}}_t$ changes. While the specific reason for choosing the unweighted sketch $\widetilde{\mathbf{A}}_t$ instead of the weighted version $\mathbf{A}_t^{\mathcal{I}_t}$ used in KORS is discussed further in Sect. 6, the following corollary shows that $\widetilde{\mathbf{A}}_t$ is as accurate as $\mathbf{A}_t^{\mathcal{I}_t}$ in approximating $\mathbf{A}_t$ up to the smallest sampling probability $\widetilde{p}_t^\gamma$.

Corollary 1. *Let $\widetilde{p}_{\min}^\gamma = \min_{t=1}^T \widetilde{p}_t^\gamma$. Then w.h.p., we have*

$$(1-\varepsilon)\widetilde{p}_{\min}\mathbf{A}_t \preceq \widetilde{p}_{\min}\mathbf{A}_t^{\mathcal{I}_t} \preceq \widetilde{\mathbf{A}}_t.$$

We can now state the main result of this section. Since for SKETCHED-KONS we are interested not only in regret minimization, but also in space and time complexity, we do not consider the case $\sigma = 0$, because when the function does not have any curvature, standard GD already achieves the optimal regret of $\mathcal{O}(\sqrt{T})$ (Zhu & Xu, 2015) while requiring only $\mathcal{O}(t)$ space and time per iteration.

Theorem 3. *For any sequence of losses ℓ_t satisfying Asm. 1-2, let $\sigma = \min_t \sigma_t$ and $\overline{\tau}_{\min} = \min_{t=1}^{T} \overline{\tau}_{t,t}$. When $\eta_t \geq \sigma > 0$ for all t, $\alpha \leq \sqrt{T}$, $\beta \geq 3\log(T/\delta)/\varepsilon^2$, if we set $\eta_t = \sigma$ then w.p. $1-\delta$ the regret of Alg. 3 satisfies*

$$\widetilde{R}_T \leq \alpha\|\mathbf{w}^*\|^2 + 2\frac{d_{\textit{eff}}^T\left(\alpha/(\sigma L^2)\right)\log(2\sigma L^2 T)}{\sigma\max\{\gamma, \beta\overline{\tau}_{\min}\}}, \tag{5}$$

and the algorithm runs in $\mathcal{O}(d_{\textit{eff}}^t(\alpha)^2 + t^2\gamma^2)$ time and $\mathcal{O}(d_{\textit{eff}}^t(\alpha)^2 + t^2\gamma^2)$ space complexity for each iteration t.

Proof sketch: Given these guarantees, we need to bound R_G and R_D. Bounding R_D is straightforward, since by construction SKETCHED-KONS adds at most $\eta_t \mathbf{g}_t\mathbf{g}_t^\mathsf{T}$ to $\widetilde{\mathbf{A}}_t$ at each step. To bound R_G instead, we must take into account that an unweighted $\widetilde{\mathbf{A}}_t = \overline{\mathbf{\Phi}}_t\mathbf{R}_t\mathbf{R}_t^\mathsf{T}\overline{\mathbf{\Phi}}_t^\mathsf{T} + \alpha\mathbf{I}$ can be up to $\widetilde{p}_{\min}$ distant from the weighted $\overline{\mathbf{\Phi}}_t\mathbf{S}_t\mathbf{S}_t^\mathsf{T}\overline{\mathbf{\Phi}}_t^\mathsf{T}$ for which we have guarantees. Hence the $\max\{\gamma, \beta\overline{\tau}_{\min}\}$ term appearing at the denominator.

6. Discussion

Regret guarantees. From Eq. 5 we can see that when $\overline{\tau}_{\min}$ is not too small, setting $\gamma = 0$ we recover the guarantees of exact KONS. Since usually we do not know $\overline{\tau}_{\min}$, we can choose to set $\gamma > 0$, and as long as $\gamma \geq 1/\operatorname{polylog} T$, we preserve a (poly)-logarithmic regret.

Computational speedup. The time required to compute $\mathbf{k}_{[t-1],t}$, $k_{t,t}$, and $\mathbf{k}_{[t-1],t}^\mathsf{T}\mathbf{D}_{t-1}\mathbf{b}_{t-1}$ gives a minimum $\mathcal{O}(t)$ per-step complexity. Note that $\overline{\mathbf{K}}_{t-1}\mathbf{b}_{t-1}$ can also be computed incrementally in $\mathcal{O}(t)$ time. Denoting the size of the dictionary at time t as $B_t = \widetilde{\mathcal{O}}(d_{\text{eff}}(\alpha)_t + t\gamma)$, computing $[\widetilde{\mathbf{b}}_t]_i$ and $\mathbf{k}_{[t-1],t}^\mathsf{T}\mathbf{D}_{t-1}\mathbf{R}_{t-1}\mathbf{E}_{t-1}^{-1}\mathbf{R}_{t-1}\overline{\mathbf{K}}_{t-1}\mathbf{b}_{t-1}$ requires an additional $\mathcal{O}(B_t^2)$ time. When $\gamma \leq d_{\text{eff}}^t(\alpha)/t$, each iteration takes $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2)$ to compute $\widetilde{\tau}_{t,t}$ incrementally using KORS, $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2)$ time to update $\widetilde{\mathbf{A}}_t^{-1}$ and $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2)$ time to compute $[\mathbf{b}_t]_t$. When $\gamma > d_{\text{eff}}^t(\alpha)/t$, each iteration still takes $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2)$ to compute $\widetilde{\tau}_{t,t}$ using KORS and $\mathcal{O}(t^2\gamma^2)$ time to update the inverse and compute $[\mathbf{b}_t]_t$. Therefore, in the case when $\overline{\tau}_{\min}$ is not too small, our runtime is of the order $\mathcal{O}(d_{\text{eff}}^t(\alpha)^2 + t)$, which is almost as small as the $\mathcal{O}(t)$ runtime of GD but with the advantage of a second-order method logarithmic regret. Moreover, when $\overline{\tau}_{\min}$ is small and we set a large γ, we can trade off a $1/\gamma$ increase in regret for a γ^2 decrease in space and time complexity when compared to exact KONS (e.g., setting $\gamma = 1/10$ would correspond to a tenfold increase in regret, but a hundred-fold reduction in computational complexity).

Asymptotic behavior. Notice however, that space and time complexity, grow roughly with a term $\Omega(t\min_{s=1}^{t}\widetilde{p}_s) \sim \Omega(t\max\{\gamma, \beta\overline{\tau}_{\min}\})$, so if this quantity does not decrease over time, the computational cost of SKETCHED-KONS will remain large and close to exact KONS. This is to be expected, since SKETCHED-KONS must always keep an accurate sketch in order to guarantee a logarithmic regret bound. Note that Luo et al. (2016) took an opposite approach for LOCO, where they keep a fixed-size sketch but possibly pay in regret, if this fixed size happens to be too small. Since a non-logarithmic regret is achievable simply running vanilla GD, we rather opted for an adaptive sketch at the cost of space and time complexity. In batch optimization, where ℓ_t does not change over time, another possibility is to stop updating the solution once $\overline{\tau}_{\min}$ becomes too small. When $\mathbf{H}_s$ is the Hessian of ℓ in $\mathbf{w}_s$, then the quantity $\mathbf{g}_t^\mathsf{T}\mathbf{H}_t^{-1}\mathbf{g}_t$, in the context of Newton's method, is called *Newton decrement* and it corresponds up to constant factors to $\overline{\tau}_{\min}$. Since a stopping condition based on Newton's decrement is directly related to the near-optimality of the current $\mathbf{w}_t$ (Nesterov & Nemirovskii, 1994), stopping when $\overline{\tau}_{\min}$ is small also provides guarantees about the quality of the solution.

Sampling distribution. Note that although $\gamma > 0$ means that all columns have a small uniform chance of being selected for inclusion in $\widetilde{\mathbf{A}}_t$, this is *not* equivalent to uniformly sampling columns. It is rather a combination of a RLS-based sampling to ensure that columns important to reconstruct $\mathbf{A}_t$ are selected and a threshold on the probabilities to avoid too much variance in the estimator.

Biased estimator and results in expectation. The random approximation $\widetilde{\mathbf{A}}_t$ is biased, since $\mathbb{E}[\overline{\mathbf{\Phi}}_t\mathbf{R}_t\mathbf{R}_t^\mathsf{T}\overline{\mathbf{\Phi}}_t^\mathsf{T}] = \overline{\mathbf{\Phi}}_t\operatorname{Diag}(\{\overline{\tau}_{t,t}\})\overline{\mathbf{\Phi}}_t^\mathsf{T} \neq \overline{\mathbf{\Phi}}_t\overline{\mathbf{\Phi}}_t^\mathsf{T}$. Another option would be to use a weighted and unbiased approximation $\widetilde{\mathbf{A}}_t' = \sum_{s=1}^{t}\eta_s z_s/\widetilde{p}_s\mathbf{g}_s\mathbf{g}_s^\mathsf{T}$ used in KORS and a common choice in matrix approximation methods, see e.g., Alaoui & Mahoney, 2015. Due to its unbiasedness, this variant would automatically achieve the same logarithmic regret as exact KONS *in expectation* (similar to the result obtained by Luo et al., 2016, using Gaussian random projection in LOCO). While any unbiased estimator, e.g., uniform sampling of $\mathbf{g}_t$, would achieve this result, RLS-based sampling already provides strong reconstruction guarantees sufficient to bound R_G. Nonetheless, the weights $1/\widetilde{p}_s$ may cause large variations in $\widetilde{\mathbf{A}}_t$ over consecutive steps, thus leading to a large regret R_D in high probability.

Limitations of dictionary learning approaches and open problems. From the discussion above, it appears that a weighted, unbiased dictionary may not achieve high-probability logarithmic guarantee because of the high variance coming from sampling. On the other hand, if we want to recover the regret guarantee, we may have to pay for it with a large dictionary. This may actually be due to the analysis, the algorithm, or the setting. An important property of the dictionary learning approach used in KORS is that it can only add *but not remove* columns and potentially re-weight them. Notice that in the batch setting (Alaoui & Mahoney, 2015; Calandriello et al., 2017), the sampling of columns does not cause any issue and we can have strong learning guarantees in high probability with a small dictionary. Alternative sketching methods such as Frequent Directions (FD, Ghashami et al., 2016a) do *create new atoms* as learning progresses. By restricting to composing dictionaries from existing columns, we only have the degree of freedom of the weights of the columns. If we set the weights to have an unbiased estimate, we achieve an accurate R_G but suffer a huge regret in R_D. On the other hand, we can store the columns unweighted to have small R_D but large R_G. This could be potentially fixed if we knew how to remove less important columns from dictionary to gain some slack in R_D.

We illustrate this problem with following simple scenario. The adversary always presents to the learner the same point $\mathbf{x}$ (with associated $\boldsymbol{\phi}$), but for the loss it alternates between $\ell_{2t}(\mathbf{w}_t) = (C - \boldsymbol{\phi}^\mathsf{T}\mathbf{w}_t)^2$ on even steps and $\ell_{2t+1}(\mathbf{w}_t) = (-C - \boldsymbol{\phi}^\mathsf{T}\mathbf{w}_t)^2$ on odd steps. Then, $\sigma_t = \sigma = 1/(8C^2)$, and we have a gradient that always points in the same $\boldsymbol{\phi}$ direction, but switches sign at each step. The optimal solution in hindsight is asymptotically $\mathbf{w} = \mathbf{0}$ and let this be also our starting point $\mathbf{w}_0$. We also set $\eta_t = \sigma$, since this is what ONS would do, and $\alpha = 1$ for simplicity.

For this scenario, we can compute several useful quantities in closed form, in particular, R_G and R_D,

$$R_G \leq \sum_{t=1}^{T} \frac{\dot{g}_t^2}{\sum_{s=1}^{t} \dot{g}_s^2\sigma + \alpha} \leq \sum_{t=1}^{T} \frac{C^2}{C^2\sigma t + \alpha} \leq \mathcal{O}(\log T),$$
$$R_D = \sum\nolimits_{s=1}^{t} (\eta_t - \sigma)(\mathbf{w}_t^\mathsf{T}\mathbf{g}_t)^2 = 0.$$

Note that although the matrix $\mathbf{A}_t$ is rank 1 at each time step, vanilla ONS does not take advantage of this easy data, and would store it all with a $\mathcal{O}(t^2)$ space in KOCO.

As for the sketched versions of ONS, sketching using FD (Luo et al., 2016) would adapt to this situation, and only store a single copy of $\mathbf{g}_t = \mathbf{g}$, achieving the desired regret with a much smaller space. Notice that in this example, the losses ℓ_t are effectively strongly convex, and even basic gradient descent with a stepsize $\eta_t = 1/t$ would achieve logarithmic regret (Zhu & Xu, 2015) with even smaller space. On the other hand, we show how the dictionary-based sketching has difficulties in minimizing the regret bound from Prop. 1 in our simple scenario. In particular, consider an arbitrary (possibly randomized) algorithm that is allowed only to reweight atoms in the dictionary and not to create new ones (as FD). In our example, this translates to choosing a schedule of weights w_s and set $\widetilde{\mathbf{A}}_t = \sum_{s=1}^{t} w_s \overline{\boldsymbol{\phi}}_s \overline{\boldsymbol{\phi}}_s = W_t \overline{\boldsymbol{\phi}\boldsymbol{\phi}}$ with total weight $W = W_T = \sum_{s=1}^{T} w_s$ and space complexity equal to the number of non-zero weights $B = |\{w_s \neq 0\}|$. We can show that there is no schedule for this specific class of algorithms with good performance due to the following three conflicting goals.

(1) To mantain R_G small, $\sum_{s=1}^{t} w_s$ should be as large as possible, as early as possible.

(2) To mantain R_D small, we should choose weights $w_t > 1$ as few times as possible, since we accumulate $\max\{w_t - 1, 0\}$ regret every time.

(3) To mantain the space complexity small, we should choose only a few $w_t \neq 0$.

To enforce goal (3), we must choose a schedule with no more than B non-zero entries. Given the budget B, to satisfy goal (2) we should use all the B budget in order to exploit as much as possible the $\max\{w_t - 1, 0\}$ in R_D, or in other words we should use exactly B non-zero weights, and none of these should be smaller than 1. Finally, to minimize R_G we should raise the sum $\sum_{s=1}^{t} w_s$ as quickly as possible, settling on a schedule where $w_1 = W - B$ and $w_s = 1$ for all the other B weights. It easy to see that if we want logarithmic R_G, W needs to grow as T, but doing so with a logarithmic B would make $R_D = T - B = \Omega(T)$. Similarly, keeping $W = B$ in order to reduce R_D would increase R_G. In particular notice, that the issue does not go away even if we know the RLS perfectly, because the same reasoning applies. This simple example suggests that dictionary-based sketching methods, which are very successful in batch scenarios, may actually fail in achieving logarithmic regret in online optimization.

This argument raises the question on how to design alternative sketching methods for the second-order KOCO. A first approach, discussed above, is to reduce the dictionary size dropping columns that become less important later in the process, without allowing the adversary to take advantage of this forgetting factor. Another possibility is to deviate from the ONS approach and $R_D + R_G$ regret decomposition. Finally, as our counterexample in the simple scenario hints, creating new atoms (either through projection or merging) allows for better adaptivity, as shown by FD (Ghashami et al., 2016a) based methods in LOCO. However, the kernelization of FD does not appear to be straighforward. The most recent step in this direction (in particular, for kernel PCA) is only able to deal with finite feature expansions (Ghashami et al., 2016b) and therefore its application to kernels is limited.

Acknowledgements The research presented was supported by French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council and French National Research Agency projects ExTra-Learn (n.ANR-14-CE24-0010-01) and BoB (n.ANR-16-CE23-0003)

References

Alaoui, Ahmed El and Mahoney, Michael W. Fast randomized kernel methods with statistical guarantees. In *Neural Information Processing Systems*, 2015.

Calandriello, Daniele, Lazaric, Alessandro, and Valko, Michal. Distributed sequential sampling for kernel matrix approximation. In *International Conference on Artificial Intelligence and Statistics*, 2017.

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

Cesa-Bianchi, Nicolo, Conconi, Alex, and Gentile, Claudio. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.

Cohen, Michael B, Musco, Cameron, and Pachocki, Jakub. Online row sampling. *International Workshop on Approximation, Randomization, and Combinatorial Optimization*, 2016.

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

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Gammerman, Alex, Kalnishkan, Yuri, and Vovk, Vladimir. On-line prediction with kernels and the complexity approximation principle. In *Uncertainty in Artificial Intelligence*, 2004.

Ghashami, Mina, Liberty, Edo, Phillips, Jeff M, and Woodruff, David P. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016a.

Ghashami, Mina, Perry, Daniel J, and Phillips, Jeff. Streaming kernel principal component analysis. In *International Conference on Artificial Intelligence and Statistics*, 2016b.

Hazan, Elad, Kalai, Adam, Kale, Satyen, and Agarwal, Amit. Logarithmic regret algorithms for online convex optimization. In *Conference on Learning Theory*, 2006.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kivinen, J., Smola, A.J., and Williamson, R.C. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52(8), 2004.

Le, Quoc, Sarlós, Tamás, and Smola, Alex J. Fastfood - Approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*, 2013.

Le, Trung, Nguyen, Tu, Nguyen, Vu, and Phung, Dinh. Dual Space Gradient Descent for Online Learning. In *Neural Information Processing Systems*, 2016.

Lu, Jing, Hoi, Steven C.H., Wang, Jialei, Zhao, Peilin, and Liu, Zhi-Yong. Large scale online kernel learning. *Journal of Machine Learning Research*, 17(47):1–43, 2016.

Luo, Haipeng, Agarwal, Alekh, Cesa-Bianchi, Nicolo, and Langford, John. Efficient second-order online learning via sketching. *Neural Information Processing Systems*, 2016.

Nesterov, Yurii and Nemirovskii, Arkadii. *Interior-point polynomial algorithms in convex programming*. Society for Industrial and Applied Mathematics, 1994.

Orabona, Francesco and Crammer, Koby. New adaptive algorithms for online classification. In *Neural Information Processing Systems*, 2010.

Orabona, Francesco, Keshet, Joseph, and Caputo, Barbara. The projectron: a bounded kernel-based perceptron. In *International Conference on Machine learning*, 2008.

Rudi, Alessandro, Camoriano, Raffaello, and Rosasco, Lorenzo. Less is more: Nyström computational regularization. In *Neural Information Processing Systems*, 2015.

Schölkopf, Bernhard and Smola, Alexander J. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, 2001.

Srinivas, Niranjan, Krause, Andreas, Seeger, Matthias, and Kakade, Sham M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

Tropp, Joel Aaron. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16: 262–270, 2011.

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

Zhdanov, Fedor and Kalnishkan, Yuri. An identity for kernel ridge regression. In *Algorithmic Learning Theory*, 2010.

Zhu, C. and Xu, H. Online gradient descent in function space. *ArXiv:1512.02394*, 2015.

Zinkevich, Martin. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, 2003.