Journal of Optimization Theory and Applications (2021) 189:437–457
https://doi.org/10.1007/s10957-021-01836-9

Bounds for the Tracking Error of First-Order Online Optimization Methods

Liam Madden[1] · **Stephen Becker**[1] · **Emiliano Dall'Anese**[2]

Received: 10 March 2020 / Accepted: 15 February 2021 / Published online: 16 March 2021

Abstract

This paper investigates online algorithms for smooth time-varying optimization problems, focusing first on methods with constant step-size, momentum, and extrapolation-length. Assuming strong convexity, precise results for the tracking iterate error (the limit supremum of the norm of the difference between the optimal solution and the iterates) for online gradient descent are derived. The paper then considers a general first-order framework, where a universal lower bound on the tracking iterate error is established. Furthermore, a method using "long-steps" is proposed and shown to achieve the lower bound up to a fixed constant. This method is then compared with online gradient descent for specific examples. Finally, the paper analyzes the effect of regularization when the cost is not strongly convex. With regularization, it is possible to achieve a non-regret bound. The paper ends by testing the accelerated and regularized methods on synthetic time-varying least-squares and logistic regression problems, respectively.

Keywords Smooth convex optimization · Online optimization · Convergence bound · Nesterov acceleration · Tikhonov regularization

Communicated by Jérôme Bolte.

✉ Liam Madden
liam.madden@colorado.edu

Stephen Becker
stephen.becker@colorado.edu

Emiliano Dall'Anese
emiliano.dallanese@colorado.edu

[1] Department of Applied Mathematics, University of Colorado Boulder, Boulder, USA

[2] Department of Electrical Computer and Energy Engineering, University of Colorado Boulder, Boulder, USA

1 Introduction

Modern optimization applications have increased in scale and complexity [16], and furthermore, some applications require solutions to a series of problems with low latency. For example, in contrast to legacy power distribution systems that were built for constant and unidirectional flow, modern power systems that include solar power at the residential nodes must incorporate variable and bidirectional flow. Thus, in order to preserve efficiency, control decisions, solved via optimization, need to be made frequently, at the time scale of changing renewable generation and non-controllable loads (e.g., seconds). Making the problem even harder is that the problem must be solved for a greater number of control points, and so it takes longer to find a suitable solution. In this example, and many others, batch algorithms take longer to find a suitable solution than is allowed, and so we have to abandon them in favor of online algorithms. Motivated by applications requiring a time-varying framework, such as power systems [17,45], transportation systems [11], and communication networks [14], this paper evaluates such online algorithms.

In particular, we consider online first-order algorithms. If it takes time $h > 0$ to make one gradient call (or one gradient call and one evaluation of the proximal operator), then we encode this into the time-varying problem by discretizing the cost function with respect to h, leading to a sequence of cost functions. The goal of an online algorithm is to track the minimum of this sequence, taking only one step at each time.

In the presence of strong convexity, upper bounds for online gradient descent have been proved in, e.g., [8,44]. For completeness, we include such an upper bound in Theorem 3.1. Furthermore, for the first time it is established (Theorem 3.2) that this is a tight bound. Beyond online gradient descent, there are dynamic regret bounds for online accelerated methods, but these are not shown to be optimal [50]. This paper goes further. First, Theorem 4.1 proves a lower bound for online first-order methods in general. Then, we define a method that we call online long-step Nesterov's method, and prove a proportional upper bound for it in Theorem 4.2. In the absence of strong convexity, we show, in Theorem 5.1, that regularizing online gradient descent leads to a useful bound that is not in terms of the regret. Finally, this paper demonstrates the performance of the algorithms by applying them to synthetic time-varying least squares and logistic regression problems.

The rest of the paper is organized as follows. Section 2 provides all the necessary preliminaries. Section 3 considers online gradient descent, online Polyak's method, and online Nesterov's method. Section 4 considers the full class of online first-order methods, including online long-step Nesterov's method. Section 5 details the results for online regularized gradient descent. Section 6 demonstrates the performance of the algorithms on some numerical examples. Section 7 concludes the paper.

2 Preliminaries

This section reviews useful results from convex analysis [6,7,12,13,32,33], and introduces key definitions that will be used throughout the paper. The paper considers

functions on a Hilbert space $\mathcal{H}$ with corresponding norm $\|\cdot\|$. We assume all functions, $f_t : \mathcal{H} \to \mathbb{R}$, are proper, lower semicontinuous, convex, and strongly smooth; $t \in \mathbb{N} \cup \{0\}$ denotes the time index [16,29,37,43]. The discretized time-varying optimization problem of interest is the sequence of convex problems (one for each t):

$$\min_{x \in \mathcal{H}} f_t(x), \qquad t \in \mathbb{N} \cup \{0\} \tag{1}$$

along with a time-varying minimizer set. In particular, assume that the minimizer set, denoted as $X_t^\star$, is nonempty for each t. Let $x_t^\star$ denote an element of $X_t^\star$ and $f_t^\star = f_t(x_t^\star)$. We denote the set of sequences of L-strongly smooth functions by $\mathcal{S}'(L)$. For function sequences that are additionally μ-strongly convex, $X_t^\star$ contains only one point; therefore, we can measure the temporal variability of the optimal solution trajectory of (1) with the sequence

$$\sigma_t := \|x_{t+1}^\star - x_t^\star\|, \qquad t \in \mathbb{N} \cup \{0\}.$$

We define $\mathcal{S}(\kappa^{-1}, L, \sigma)$, where κ is the condition number, as the set of L-strongly smooth, $\kappa^{-1}L$-strongly convex functions sequences with $\sigma_t \leq \sigma$ for all t. For both $\mathcal{S}'(L)$ and $\mathcal{S}(\kappa^{-1}, L, \sigma)$, the Hilbert space, along with its dimension, is left implicit since the results are independent of it.

Throughout the paper, we will consider various measures of optimality [8,10,16, 21,23,29,37,43,49]: the iterate error $\|x_t - x_t^\star\|$, the function error $f_t(x_t) - f_t^\star$, and the gradient error $\|\nabla f_t(x_t)\|$. For functions that are not strongly convex, the iterate error yields the strongest results, while the gradient error is the weakest. That is,

$$\|\nabla f_t(x)\| \leq L\|x - x_t^\star\|, \tag{2}$$

$$f_t(x) - f_t^\star \leq \frac{L}{2}\|x - x_t^\star\|^2, \tag{3}$$

$$\text{and } \|\nabla f_t(x)\|^2 \leq 2L\left(f_t(x) - f_t^\star\right) \tag{4}$$

where the first two inequalities follow from standard arguments in convex analysis, and the third inequality follows by comparing a point and a gradient step from it in the definition of strong smoothness [42, Sec. 12.1.3]. For functions that are strongly convex, bounds in the opposite directions can be found.

Due to the temporal variability, we cannot expect any of the error sequences to converge to zero in general. Thus, we will characterize the performance of online algorithms via bounds on the limit supremum of the errors, which we term "tracking" error, rather than bounds on the convergence rate (to the tracking error ball).

Note that for $\mathcal{S}'(L)$, the regret literature uses the dynamic regret, $Reg_T := \sum_{t=0}^{T} f_t(x_t) - f_t^\star$, as a measure of optimality [10,21,23,49]. The path variation, function variation, and gradient variation—respectively,

$$V_T^p = \max_{\{x_t^\star \in X_t^\star\}_{t=0}^T} \sum_{t=1}^{T} \|x_t^\star - x_{t-1}^\star\|,$$

$$V_T^f = \sum_{t=1}^{T} \sup_x |f_t(x) - f_{t-1}(x)|,$$
$$\text{and } V_T^g = \sum_{t=1}^{T} \sup_x \|\nabla f_t(x) - \nabla f_{t-1}(x)\|$$

—are used to bound the dynamic regret. On the other hand, our approach to $\mathcal{S}'(L)$ is inspired by regularization reduction [1]. Through regularization, we are able to bound the tracking gradient error via the (σ_t) corresponding to the regularized problem. But, there are a couple of reasons why our bound cannot be compared with the bounds in the regret literature. First, it is possible for $Reg_T/(T+1)$ to be bounded while the function error has a subsequence going to infinity. For example, let $(\Delta_t)_{t\in\mathbb{N}\cup\{0\}} = (1, 0, 2, 0, 0, 3, 0, 0, 0, \ldots)$. Then $\frac{1}{T+1}\sum_{t=0}^{T}\Delta_t \leq 1$ even though there is a subsequence of (Δ_t) that goes to infinity. Second, in order for the function variation or gradient variation to be finite, the constraint set must be compact. Our bound only requires that the optimal solution trajectory lie in a compact set.

In this paper, we consider three general algorithms: ALG(α, β, η) presented in Sect. 3, online long-step Nesterov's method (OLNM(T)) presented in Sect. 4, and online regularized gradient descent (ORGD(δ, x_c)), presented in Sect. 5. ALG(α, β, η) encompasses methods such as online gradient descent, online Polyak's method (also known as the online heavy ball method), and online Nesterov's method as special cases (see Table 1).

The proposed OLNM is motivated by the following observation: suppose that the optimizer has access to all the previous functions; depending on the temporal variability of the problem, the question we pose is whether it is better to use the same function for some number of iterations before switching to a new one, or utilize a new function at each iteration. This question is relevant especially in the case where "observing" a new function may incur a cost (for example, in sensor networks, where acquisition of new data may be costly from a battery and data transmission standpoint). We call the former idea "long-steps," drawing an analogy to interior-point methods [34, Sec. 14], [12, Sec. 11]. Specifically, we pause at a function for some number of iterations, and we apply a first-order algorithm with the optimal coefficients for the number of iterations, in the sense of [20,25,46]. Thus, short-steps are online gradient steps. Restarting Nesterov's method has been shown to be optimal for strongly convex functions in, e.g. [35], [2, App. D]. It turns out that the optimal restart count as a long-step length is optimal for the strongly convex time-varying problem.

On the other hand, the literature on batch algorithms with inexact gradient calls has shown that accelerated methods accumulate errors for some non-strongly convex functions [40, Prop. 2], [18, Thm. 7], [4,9,48]. This suggests that there may be some non-strongly convex function sequences such that all online accelerated methods perform worse than online gradient descent. On the other hand, only averaged function error bounds exist for online gradient descent. We get around this via regularization; in particular, we balance the error introduced by the regularization term with a smaller tracking error achieved by the regularized algorithm. This allows us to derive a non-averaged error bound for ORGD.

Table 1 Special cases of $\texttt{ALG}(\alpha, \beta, \eta)$ for $(f_t) \in \mathcal{S}(\kappa^{-1}, L, \sigma)$ where $\mu = L/\kappa$

Form of $\texttt{ALG}(\alpha, \beta, \eta)$	Typical parameter choice	Name
$\texttt{ALG}(\alpha, 0, 0)$	$0 < \alpha < \frac{2}{L}$	Online gradient descent
$\texttt{ALG}(\alpha, \beta, 0)$	$\alpha = \frac{4}{\left(\sqrt{L}+\sqrt{\mu}\right)^2}, \beta = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2$	Online Polyak's method [36]
$\texttt{ALG}(\alpha, \beta, \beta)$	$\alpha = 1/L, \beta = \eta = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$	Online Nesterov's method [32]

While the paper considers strongly smooth functions for simplicity of exposition, we note how to extend results to the composite case (where the cost function is the sum of f and a possibly nondifferentiable function with computable proximal operator, such as an indicator function encoding constraints or an ℓ_1 penalty). We leave the analysis for Banach spaces as a future direction. See [47, Sec. 3.1] for an excellent treatment of acceleration for Banach spaces.

3 Simple First-Order Methods

In this section, we focus on the class of algorithms that, given x_0, construct (x_t) via:

$$\begin{aligned} x_{t+1} &= x_t - \alpha \nabla f_t(y_t) + \beta(x_t - x_{t-1}) \qquad (\texttt{ALG}(\alpha, \beta, \eta)) \\ y_{t+1} &= x_{t+1} + \eta(x_{t+1} - x_t) \end{aligned} \tag{5}$$

which will be referred to as $\texttt{ALG}(\alpha, \beta, \eta)$, where $\alpha > 0$ is the step-size, β is the momentum, and η is the extrapolation-length. $\texttt{ALG}(\alpha, 0, 0)$ corresponds to online gradient descent, $\texttt{ALG}(\alpha, \beta, 0)$ to an online version of Polyak's method [36], and $\texttt{ALG}(\alpha, \beta, \beta)$ to an online version of Nesterov's method [32].

It should be pointed out that, while the analysis of online algorithms for time-varying optimization such as $\texttt{ALG}(\alpha, \beta, \eta)$ share commonalities with online learning [21–23], the two differ in their motivations and aspects of their implementations, as noted in [16]. For example, in online learning frameworks, the step size may depend on the time-horizon or, in the case of an infinite time-horizon, a doubling-trick may be utilized [41, Sec. 2.3.1]. On the other hand, every step of an online algorithm applied to a time-varying problem is essentially the *first* step at that time. Hence, the parameters of the algorithm should be cyclical or depend on measurements of the temporal variation. We only consider the former. The simplest subset of cyclical algorithms is the set of algorithms with constant parameters, as in ALG.

In the following subsections, we give a tight bound on the performance of online gradient descent and analyze ALG for two examples.

3.1 Online Gradient Descent

When the function f_t is strongly convex for all t, upper bounds on the tracking errors for online gradient descent are available in the literature (see, e.g., [8,19,28,44]); these results are tailored to the setting considered in this paper by the following theorem, which is followed by two examples used to give intuition and prove tightness results.

Theorem 3.1 *Suppose that* $(f_t) \in \mathcal{S}(\kappa^{-1}, L, \sigma)$ *and let* $\alpha \in]0, 2/(\mu + L)]$. *Then, given* x_0, $\texttt{ALG}(\alpha, 0, 0)$ *constructs a sequence* (x_t) *such that*

$$\limsup_{t\to\infty} \|x_t - x_t^\star\| \leq (\alpha\mu)^{-1}\sigma \tag{6}$$

where $\mu = \kappa^{-1}L$. *In particular, the bound is minimized for* $\alpha = \frac{2}{\mu+L}$, *in which case,*

$$\limsup_{t\to\infty} \|x_t - x_t^\star\| \leq \frac{\kappa+1}{2}\sigma. \tag{7}$$

The proof follows by using the fact that the map $I - \alpha\nabla f_t$ is Lipschitz continuous with parameter $c = \max\{|1 - \alpha\mu|, |1 - \alpha L|\}$ [39, Sec. 5.1] and $x_t^\star$ is a fixed point of that map. The result straightforwardly extends to the composite case by using the nonexpansiveness property of the proximal operator and a similar fixed point result.

3.2 Example: Translating Quadratic

For quadratic functions with constant positive definite matrix but minimizer moving at constant speed on a straight line, the analysis of [27] can be extended, using the Neumann series, to show that the set of (α, β, η) such that ALG has a finite worst-case tracking iterate error is exactly the stability set of (α, β, η) in the batch setting (i.e., in a setting where the cost does not change during the execution of the algorithm). As an example, the stability set for online Nesterov's method is given in [5, Prop. 3.6]. Thus, in the online setting, one should consider only (α, β, η) that are in the batch stability set.

As a particular example, let $f(x) = \frac{1}{2}x^T Ax$, with $A = \mathrm{diag}(\mu, L, L, ...)$. Given (α, β, η) and initialization x_0, we want to construct $(f_t) \in \mathcal{S}(\mu/L, L, \sigma)$ in such a way that the iterates of $\texttt{ALG}(\alpha, \beta, \eta)$ trail behind $(x_t^\star)$ at a constant distance. Towards this end, define $\xi = \left(\frac{1-\beta}{\alpha\mu} + \eta\right)\sigma$, which will end up being the trailing distance. Let e denote the first canonical basis vector, and define $f_t = f(\cdot - x_t^\star)$ where $x_t^\star = x_0 + (t\sigma + \xi)e$. By induction, we will show that $\Delta_t := x_t - x_t^\star = -\xi e \; \forall t \in \mathbb{N} \cup \{0\}$. The base case follows by construction. Now, assume the result holds for all $t' \leq t$. Then,

$$\begin{aligned}
\Delta_{t+1} &= (1+\beta)x_t - \beta x_{t-1} - x_{t+1}^\star - \alpha\nabla f_t\big((1+\eta)x_t - \eta x_{t-1}\big) \\
&= (1+\beta)\Delta_t - \beta\Delta_{t-1} + (1+\beta)(x_t^\star - x_{t+1}^\star) - \beta(x_{t-1}^\star - x_{t+1}^\star) \\
&\quad - \alpha\nabla f\left((1+\eta)\Delta_t - \eta\Delta_{t-1} - \eta(x_{t-1}^\star - x_t^\star)\right)
\end{aligned}$$

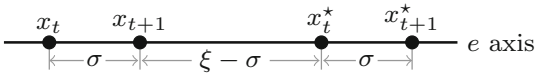

Fig. 1 Movement of iterates and minimizers

$$
\begin{aligned}
&= -(1+\beta)\xi e + \beta\xi e - (1+\beta)\sigma e + 2\beta\sigma e \\
&\quad - \alpha\nabla f\left(-(1+\eta)\xi e + \eta\xi e + \eta\sigma e\right) \\
&= -\xi e - (1-\beta)\sigma e - \alpha\nabla f(-\xi e + \eta\sigma e) \\
&= -\xi e + \alpha\mu\xi e - (1-\beta+\alpha\mu\eta)\sigma e \\
&= -\xi e
\end{aligned}
$$

and so the result holds for all t by induction. Figure 1 shows how the iterates trail behind the minimizers.
For online gradient descent, we have $\xi = (\alpha\mu)^{-1}\sigma$; this shows that the bound in Theorem 3.1 is tight. We formalize this tightness result (which is a contribution of the present paper) in Theorem 3.2.

Theorem 3.2 *For all* $\alpha \in]0, 2/(\mu + L)]$ *and initialization* x_0, $\exists (f_t) \in \mathcal{S}(\mu/L, L, \sigma)$ *such that* ALG$(\alpha, 0, 0)$ *constructs a sequence* (x_t) *with*

$$\limsup_{t\to\infty} \|x_t - x_t^\star\| = (\alpha\mu)^{-1}\sigma. \tag{8}$$

For Polyak's method, using the usual parameters, one gets $\xi = \sqrt{\kappa}\sigma$. On the other hand, for Nesterov's method one gets $\xi = \left(2\sqrt{\kappa} - 1\right)\sigma$. Thus, when applied to this example, the tracking iterate error scales with the square root of the condition number for both of these methods. However, as we will see in the next example, this is not always the case for these methods.

3.3 Example: Rotating Quadratic

Consider the function f utilized in the previous example, and consider rotating the first two canonical basis directions every iteration. We can reduce the full problem to one in $\mathbb{R}^2$. Define

$$
\begin{aligned}
f_t(x) &= \frac{1}{2}x^T A_t x \\
A_{2t} &= \begin{pmatrix} L & 0 \\ 0 & \mu \end{pmatrix} \\
A_{2t+1} &= \begin{pmatrix} \mu & 0 \\ 0 & L \end{pmatrix}
\end{aligned}
$$

and note that $(f_t) \in \mathcal{S}(\mu/L, L, \sigma)$ for all $\sigma \geq 0$. Now, let (x_t) be the sequence generated by ALG(α, β, η) given x_0. Denote $a_+ = (1+\beta) - (1+\eta)\alpha L$, $a_- = (1+\beta) - (1+\eta)\alpha\mu$, $b_+ = -\beta + \eta\alpha L$, and $b_- = -\beta + \eta\alpha\mu$. Then,

$$
\begin{aligned}
x_{2t} &= \begin{pmatrix} a_- & 0 \\ 0 & a_+ \end{pmatrix} x_{2t-1} + \begin{pmatrix} b_- & 0 \\ 0 & b_+ \end{pmatrix} x_{2(t-1)} \\
x_{2t+1} &= \begin{pmatrix} a_+ & 0 \\ 0 & a_- \end{pmatrix} x_{2t} + \begin{pmatrix} b_+ & 0 \\ 0 & b_- \end{pmatrix} x_{2t-1} \\
&= \begin{pmatrix} a_+a_- + b_+ & 0 \\ 0 & a_+a_- + b_- \end{pmatrix} x_{2t-1} + \begin{pmatrix} a_+b_- & 0 \\ 0 & a_-b_+ \end{pmatrix} x_{2(t-1)}.
\end{aligned}
$$

Define $z_t = [x_{2t};\ x_{2t+1}]$. Then,

$$
\begin{aligned}
z_t &= \begin{pmatrix} b_- & 0 & a_- & 0 \\ 0 & b_+ & 0 & a_+ \\ a_+b_- & 0 & a_+a_- + b_+ & 0 \\ 0 & a_-b_+ & 0 & a_+a_- + b_- \end{pmatrix} z_{t-1} \\
&:= C z_{t-1} \\
&= C^t z_0.
\end{aligned}
$$

Thus, $z_t \to z^\star = 0$ precisely when $\rho(C) < 1$ (since $\rho(C) = \lim_t \|C^t\|^{1/t}$). It is easy to see that C is similar to the block-diagonal matrix with D and E on the diagonal blocks where

$$
D = \begin{pmatrix} b_- & a_- \\ a_+b_- & a_+a_- + b_+ \end{pmatrix} \text{ and } E = \begin{pmatrix} b_+ & a_+ \\ a_-b_+ & a_+a_- + b_-. \end{pmatrix}
$$

Furthermore, D and E have the same trace and determinant:

$$
\begin{aligned}
\operatorname{tr}(D) &= \operatorname{tr}(E) = a_+a_- + b_+ + b_- \\
\det(D) &= \det(E) = a_+a_-b_- + b_+b_- - a_+a_-b_- = b_+b_-.
\end{aligned}
$$

Thus, $\rho(C) = \rho(D) = \rho(E)$. Note that $\rho(C) = 0$ when $\alpha = \frac{1}{L}$ and $\beta = \eta = 0$. In fact, $\texttt{ALG}(1/L, 0, 0)$ converges exactly in just two steps. Thus, online gradient descent performs better than online Polyak's method and online Nesterov's method for the rotating quadratic example. In fact, online Polyak's method actually *diverges*. For the usual parameters of Polyak's method, $\rho(C) = 6\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2$, which is bigger than 1 precisely when $\kappa > \left(\frac{\sqrt{6}+1}{\sqrt{6}-1}\right)^2 \approx 5.7$. We state this more formally in Theorem 3.3.

Theorem 3.3 *For all $\kappa \geq 6$, $L > 0$, $\sigma \geq 0$, $\exists (f_t) \in \mathcal{S}(\kappa^{-1}, L, \sigma)$, such that online Polyak's method diverges (for any non-optimal initialization x_0).*

In the batch setting, decreasing the step-size increases robustness to noise [15,27]. Thus, it makes sense that in the online setting, decreasing the step-size leads to greater stability. In other words, online Polyak's method diverges because of its large step-size. By reducing the stepsize to $\alpha = \frac{1}{L}$, then $\eta \leq \beta < 1$ is sufficient to guarantee $\rho(C) < 1$.

4 General First-Order Methods

In [30], Nemirovsky and Yudin proved lower bounds on the number of first-order oracle calls necessary for ϵ-convergence in both the smooth and convex setting and the smooth and strongly convex setting [30, Thm. 7.2.6]. Then, in [31], Nesterov constructed a method that achieved the lower bound in the smooth and convex setting. His method has a momentum parameter that goes to one as the iteration count increases. By modifying the momentum sequence, it is possible to achieve the lower bound in the smooth and strongly convex setting as well. This can be done by either setting the momentum parameter to a particular constant, restarting the original method every time the iteration count reaches a particular number, or adaptively restarting the original method [35].

Nesterov presents the lower bounds in Theorems 2.1.7 and 2.1.13 respectively of [32]. These theorems involve some subtleties, which we now discuss. First, Nesterov says that (x_t) comes from a first-order method if $x_{t+1} - x_0 \in \text{span}\{\nabla f(x_0), ..., \nabla f(x_t)\}$ for all t. This is the definition that we will generalize to the time-varying setting. Second, the lower bounds do not hold for all t. The lower bound in the smooth and convex setting only holds for $t < \frac{1}{2}(d-1)$ where d is the dimension of the space. In the smooth and strongly convex setting, Nesterov only proves the bound for infinite-dimensional spaces. In fact, for finite-dimensional spaces, the lower bound can only hold for $t \leq O(d)$ since the conjugate gradient method applied to quadratic functions converges exactly in d iterations. In order to prove lower bounds that hold for all t in finite-dimensional spaces, it is necessary to restrict to smaller classes of methods. In particular, [3] excludes the conjugate gradient method by restricting to methods with "stationary" update rules. Third, the lower bounds are based on explicit adversarial functions. In particular, we will use the adversarial function that Nesterov gives in [32, 2.1.13] to construct an adversarial sequence of functions in the time-varying setting. Thus, our lower bound only holds for infinite-dimensional spaces. It is an open problem whether the function sequence can be modified to give a lower bound that holds for all t in finite-dimensional spaces or whether it is necessary to restrict to smaller classes of methods.

We say that (x_t) comes from a first-order method if $x_{t+1} - x_0 \in \text{span}\{\nabla f_{\tau_0}(x_0), ..., \nabla f_{\tau_t}(x_t)\}$ where, for each t, $\tau_t \in \{0, ..., t\}$. More generally, we still consider (x_t) to be from a first-order method if (x_t) is a simple auxiliary sequence of some (y_t) that is more precisely first-order. Now, calling the most recent gradient at each step of the algorithm, as ALG does, corresponds to $\tau_t = t$ for all t. While it is possible for an algorithm to call an older gradient, it is not clear how this could be helpful. In Sect. 4.2 will show that having $\tau_t = T\lfloor t/T \rfloor$ for all t makes it possible to build up momentum in the online setting.

4.1 Universal Lower Bound

In Theorem 4.1 we give a generalization of Nesterov's lower bound for the online setting considered in this paper. In the proof, we omit certain details that can be found in [32, Sec. 2.1.4].

Theorem 4.1 *Let $\mathcal{H} = \ell_2(\mathbb{N})$. For any x_0, $\exists (f_t) \in \mathcal{S}(\kappa^{-1}, L, \sigma)$ such that, if (x_t) is generated by an online first-order method starting at x_0, then*

$$\|x_t - x_t^\star\| \geq \frac{\sqrt{\kappa} - 1}{2}\sigma.$$

Proof First, set $\gamma = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ and let c be the solution to $\gamma^c = \sigma\sqrt{\frac{1+\gamma}{1-\gamma}}$, namely $c = \log\left(\sigma\sqrt{\frac{1+\gamma}{1-\gamma}}\right)/\log(\gamma)$. Note that $\frac{\gamma}{\sqrt{1-\gamma^2}} = \frac{\sqrt{\kappa}-1}{2}\kappa^{-1/4} = \frac{\sqrt{\kappa}-1}{2}\sqrt{\frac{1-\gamma}{1+\gamma}}$. Set $\mu = \kappa^{-1}L$, define A as the symmetric tridiagonal operator on $\ell_2(\mathbb{N})$ with 2's on the diagonal and -1's on the sub-diagonal, and let $a \in [\mu, L]$. Abusing notation to write the operators as matrices, define

$$f_t(x) = \frac{1}{2}x^T \left(\begin{array}{c|c} aI_t & 0 \\ \hline 0 & \frac{L-\mu}{4}A + \mu I \end{array}\right) x - \gamma^c x^T \begin{pmatrix} a1_t \\ \frac{L-\mu}{4}e_1 \end{pmatrix}.$$

$$\text{Then, } \nabla f_t(x) = \left(\begin{array}{c|c} aI_t & 0 \\ \hline 0 & \frac{L-\mu}{4}A + \mu I \end{array}\right) x - \gamma^c \begin{pmatrix} a1_t \\ \frac{L-\mu}{4}e_1 \end{pmatrix}$$

$$\text{so } x_t^\star(i) = \begin{cases} \gamma^c & i \leq t \\ \gamma^{i-t+c} & i > t \end{cases}$$

$$\text{and so } \|x_{t+1}^\star - x_t^\star\|^2 = \gamma^{2c}\frac{1-\gamma}{1+\gamma}.$$

Thus, $(f_t) \in \mathcal{S}(\kappa^{-1}, L, \sigma)$. Without loss of generality, assume that $x_0 = 0$ since shifting (f_t) does not affect membership in $\mathcal{S}(\kappa^{-1}, L, \sigma)$. Then, $x_t(i) = 0\ \forall i > t$ for any first-order online algorithm. Thus,

$$\begin{aligned} \|x_t - x_t^\star\| &\geq \left(\sum_{i=t+1}^{\infty} \gamma^{2(i-t+c)}\right)^{1/2} \\ &= \gamma^c \frac{\gamma}{\sqrt{1-\gamma^2}} \\ &= \frac{\sqrt{\kappa}-1}{2}\gamma^c\sqrt{\frac{1-\gamma}{1+\gamma}} \\ &= \frac{\sqrt{\kappa}-1}{2}\sigma. \end{aligned}$$

□

Remark 4.1 If $\kappa \geq 5$, then $\frac{L-\mu}{4} \in [\mu, L]$. If we apply $\texttt{ALG}(4/(L-\mu), 0, 0)$ to the online Nesterov function with $a = \frac{L-\mu}{4}$, then it is easy to see that $\|x_t - x_t^\star\| = \frac{\sqrt{\kappa}-1}{2}\sigma$. Thus, online gradient descent with an appropriately large step-size performs optimally against the online Nesterov function.

In the following subsection, we will construct an algorithm that performs optimally up to a fixed constant against the full class $\mathcal{S}(\kappa^{-1}, L, \sigma)$; that is, it exhibits an upper bound that is equal to the lower bound of Theorem 4.1 times a fixed constant.

4.2 Online Long-Step Nesterov's Method

There is a conceptual difficulty when it comes to adapting accelerated methods to the online setting. Informally, in batch optimization, "acceleration" refers to the fact that accelerated methods converge faster than gradient descent. However, the goal in the online optimization framework considered here is reduced tracking error, and not necessarily faster convergence. As shown by the rotating quadratic example, tracking and convergence actually behave differently. Fortunately, we can leverage the fast convergence of Nesterov's method towards reduced tracking error.

In this section, we present a long-step Nesterov's method; the term "long-steps" refers to the fact that the algorithm takes a certain number of steps using the same stale function before catching up to the most recent function and repeating. For this particular long-step Nesterov's method, we are able to prove upper bounds on the tracking error (on the other hand, no bounds for the online Nesterov's method are yet available, and are the subject of current efforts).

The specific sequence constructed by $\texttt{OLNM}(T)$ is defined in Algorithm 1.

Algorithm 1 $\texttt{OLNM}(T)$

Require: x_0

$y_0 \leftarrow x_0, z_0 \leftarrow x_0, a_0 \leftarrow 1$
for $t = 1, 2, \ldots$ **do**
  $z_{t+1} = y_t - \frac{1}{L}\nabla f_{T\lfloor t/T \rfloor}(y_t)$
  if $T \nmid t+1$ **then**
    $a_{t+1} = \frac{1+\sqrt{1+4a_t^2}}{2}$
    $y_{t+1} = z_{t+1} + \frac{a_t - 1}{a_{t+1}}(z_{t+1} - z_t)$
    $x_{t+1} = x_t$ ▷ so $x_{t+1} = x_t = x_{T\lfloor t/T \rfloor}$
  else if $T \mid t+1$ **then**
    $a_{t+1} = 1$
    $y_{t+1} = z_{t+1}$
    $x_{t+1} = z_{t+1}$
  end if
end for

The method can be extended to the composite case by applying the proximal operator to the z iterates. Theorem 4.2 gives an upper bound on the tracking iterate error of OLNM, using results from [7, Thm. 10.34].

Theorem 4.2 *If $(f_t) \in \mathcal{S}(\kappa^{-1}, L, \sigma)$, then, given x_0, $\texttt{OLNM}(T)$ where $T = c\sqrt{\kappa}$ for $c > 2$ such that $c\sqrt{\kappa} \in \mathbb{N}$, constructs a sequence (x_t) such that*

$$\limsup_{t\to\infty} \|x_t - x_t^\star\| \leq \frac{2c(c-1)}{c-2}\sqrt{\kappa}\sigma. \tag{9}$$

Proof First, via standard batch optimization bounds, we have

$$\|x_{kT} - x^\star_{(k-1)T}\| \leq \frac{2\sqrt{\kappa}}{T}\|x_{(k-1)T} - x^\star_{(k-1)T}\|.$$

Thus,

$$\begin{aligned}\|x_{kT} - x^\star_{kT}\| &\leq \frac{2\sqrt{\kappa}}{T}\|x_{(k-1)T} - x^\star_{(k-1)T}\| + T\sigma \\ &\leq \left(\frac{2\sqrt{\kappa}}{T}\right)^k \|x_0 - x^\star_0\| + \frac{T\sigma}{1 - \frac{2\sqrt{\kappa}}{T}} \\ &= \left(\frac{2\sqrt{\kappa}}{T}\right)^k \|x_0 - x^\star_0\| + \frac{T^2\sigma}{T - 2\sqrt{\kappa}}\end{aligned}$$

and so

$$\begin{aligned}\|x_t - x^\star_t\| &= \|x_{T\lfloor t/T\rfloor} - x^\star_t\| \\ &\leq \|x_{T\lfloor t/T\rfloor} - x^\star_{T\lfloor t/T\rfloor}\| + \|x^\star_t - x^\star_{T\lfloor t/T\rfloor}\| \\ &\leq \left(\frac{2\sqrt{\kappa}}{T}\right)^{\lfloor t/T\rfloor} \|x_0 - x^\star_0\| + \frac{T^2\sigma}{T - 2\sqrt{\kappa}} + T\sigma.\end{aligned}$$

Then, taking the limit supremum, we get

$$\begin{aligned}\limsup_{t\to\infty} \|x_t - x^\star_t\| &\leq \frac{T^2\sigma}{T - 2\sqrt{\kappa}} + T\sigma \\ &= \frac{2T\sigma\left(T - \sqrt{\kappa}\right)}{T - 2\sqrt{\kappa}} \\ &= \frac{2c(c-1)}{c-2}\sqrt{\kappa}\sigma.\end{aligned}$$

□

If we minimize the bound in Eq. (9) over $c \in \mathbb{R}$, then we get $c = 2 + \sqrt{2}$ with a value of $2c(c-1)/(c-2) = 6 + 4\sqrt{2} \approx 11.66$; this is in contrast with the batch setting, where $c = \sqrt{8}$. However, we have the extra restriction that $c\sqrt{\kappa} \in \mathbb{N}$ so, in general, we will take $T = \lfloor(2 + \sqrt{2})\sqrt{\kappa}\rfloor$.

Note that the bound is asymptotically (as the condition number goes to infinity) optimal (hence tight) up to the constant $4c(c-1)/(c-2)$. In particular, for $\kappa \geq (4c(c-1)/(c-2))^2$, the bound is better than the bound for online gradient descent. However, these are bounds over a general class of functions. One question is how the two methods fare against specific examples, as exemplified next.

As we noted in Remark 4.1, online gradient descent with an appropriately large step-size performs optimally against the online Nesterov function. Even with a typical step-size, the tracking iterate error of online gradient descent still scales linearly

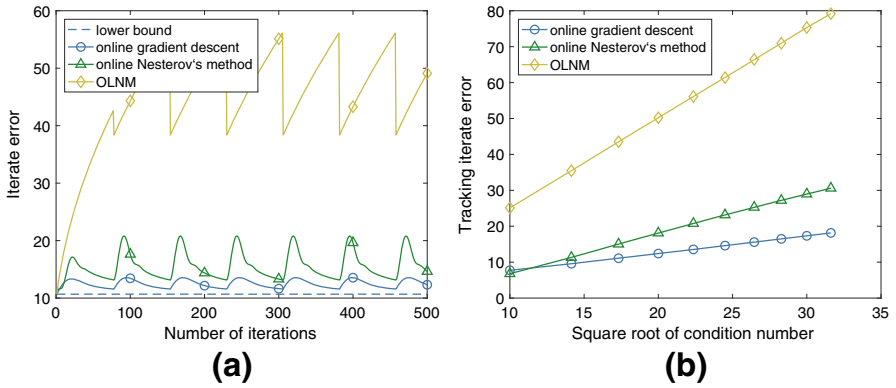

Fig. 2 Algorithms applied to the online Nesterov function with $L = 500$, $d = 1000$, $a = (L + \mu)/2$, $\sigma = 1$, and $T = \lfloor (2 + \sqrt{2})\sqrt{\kappa} \rfloor$. **a** Evolution of the iterate error for the particular example $\mu = 1$. **b** Tracking iterate error for varying μ

with the square root of the condition number. Furthermore, while OLNM also scales linearly with the square root of the condition number, online gradient descent has a smaller constant. Figure 2a depicts the iterate error for algorithms applied to the online Nesterov function with condition number equal to 500. Online gradient descent performs the best, followed by online Nesterov's method and OLNM. In fact, this can be seen in the dependence on the condition number as well. Figure 2b shows the linear dependence of the tracking iterate error on the square root of the condition number. The constant for online gradient descent is 0.481, the constant for online Nesterov's method is 1.101, and the constant for OLNM is 2.491. Note that, despite OLNM having a worse constant than online gradient descent, the former's constant is still less than its upper bound of $2c(c-1)/(c-2) \approx 11.66$.

For the rotating quadratic example, the minimizer is fixed, so OLNM has the same convergence rate as Nesterov's method does in the batch setting. However, with the right step-size, online gradient descent can converge in just two steps! On the other hand, for the translating quadratic example, since the tracking iterate error of online gradient descent scales with κ, while the tracking iterate error of OLNM scales with $\sqrt{\kappa}$, OLNM outperforms online gradient descent for sufficiently high condition number. In particular, Fig. 3a depicts the iterate error for algorithms applied to the translating quadratic function with condition number equal to 500. Online Nesterov's method performs the best, followed by OLNM and online gradient descent. Figure 3b shows the tracking iterate error for varying condition number for OLNM (online gradient descent and online Nesterov's method are left out since we analytically solved for their tracking iterate error). The constant is 7.21. Note that this is larger than the constant for OLNM applied to the online Nesterov function. While the online Nesterov function is a universal adversary, the translating quadratic is more particularly adversarial for OLNM because the minimizer is maximizing its distance away from the OLNM iterates by moving in a straight line away from the old minimizer the OLNM iterates are approaching. Also note that this is more than half of the upper bound, showing that the upper bound is tight at least up to a constant less than two.

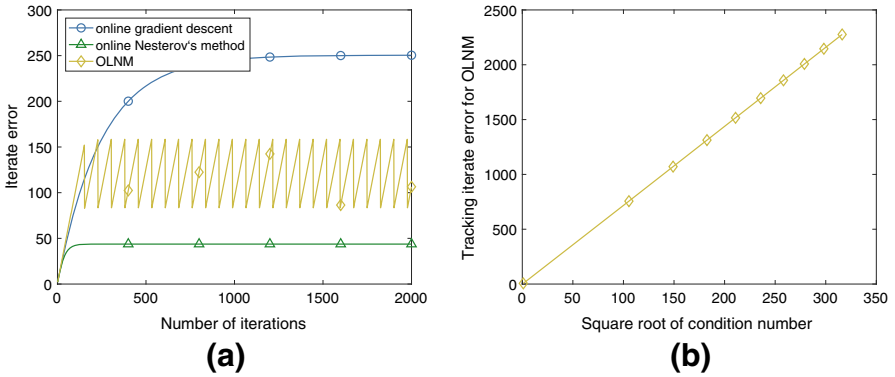

Fig. 3 Algorithms applied to the translating quadratic function with $d = 2, \sigma = 1$, and $T = \lfloor (2+\sqrt{2})\sqrt{\kappa} \rfloor$. **a** shows the evolution of the iterate error for $L = 500$ and $\mu = 1$. **b** shows the tracking iterate error for $L = 1$ and varying μ

5 Regularization

When the functions are convex, but not strongly convex, performance metrics for online first-order methods typically rely on dynamic regret bounds. The dynamic regret is the *averaged* function error. We take a different approach, however, deriving a gradient error bound via regularization. While gradient error bounds are weaker than function error bounds, the benefit is that we bound the error, rather than the averaged error.

The main idea is that there is a trade-off between tracking error and regularization error. Regularizing by any amount means that we can apply Theorem 3.1. In this case, increasing the amount of regularization increases the regularization error while also decreasing the tracking error. In the following, we provide a framework for balancing these two errors.

Given x_0, let *online regularized gradient descent,* $\texttt{ORGD}(\delta, x_c)$, with $\delta > 0$, construct the sequence (x_t) via

$$x_{t+1} = x_t - \frac{2}{L + 2\delta}(\nabla f_t(x_t) + \delta(x_t - x_c)). \qquad (\texttt{ORGD}(\delta, x_c))$$

It is easy to see that $\texttt{ORGD}(\delta, x_c)$ is vanilla online gradient descent for the regularized problem $f_t(\cdot; \delta, x_c) = f_t + \frac{\delta}{2}\| \cdot - x_c\|^2$. Since we don't vary x_c in the analysis, we write $f_t(\cdot; \delta)$ for simplicity.

Now, in order to bound the algorithm error of ORGD, we need to bound the regularization error. As with the algorithm error, we can measure the regularization error in terms of the variable, the cost, or the gradient. Unfortunately, it is impossible, without further assumptions, to bound the variable regularization error, $\|x^\star - x_r^\star\|$ where $x_r^\star$ is the unique minimizer to the regularized problem [26,38]. For example, if we regularize a constant function by any amount, then there are minimizers arbitrarily far away from the regularized minimizer. However, if we assume the function is coercive ($\|x\| \to \infty \implies f(x) \to \infty$), then we know the variable regularization error *is* bounded. But, it is still impossible to bound it in terms of the

strong smoothness constant. For example, assume that we are going to regularize by adding $\frac{\delta}{2}\|\cdot\|^2$. Then, for an arbitrary distance ξ, there exists an L-smooth function such that the variable regularization error is greater than ξ. In particular, consider $f(x) = \frac{\lambda}{2}\|x - 2\xi e\|^2$ where $\lambda \leq \min\{L, \delta\}$ and e is a unit vector. Then $x_r^\star = \frac{\lambda}{\lambda+\delta}e$ and so $\|x^\star - x_r^\star\| = \frac{\delta}{\lambda+\delta}\|2\xi e\| \geq \xi$.

Fortunately, even without coercivity, it *is* possible to bound the gradient regularization error [32, Thm. 2.2.7], which allows us to bound the tracking gradient error of ORGD. However, we do have to make an additional assumption. Loosely, we have to assume that the sequence of minimizer sets doesn't "drift." To make the assumption more precise, we need some definitions.

First, let $x_t^\star(\delta) = \operatorname{argmin}_x f_t(x; \delta)$ for $\delta > 0$. We know the right-hand side is a singleton by strong convexity. Furthermore, since f_t is proper, closed, and convex, ∂f_t is maximally monotone, and so we can apply [6, Thm. 23.44], which tells us that $x_t^\star(\cdot)$ is continuous and $\lim_{\delta\to 0} x_t^\star(\delta)$ is the unique projection of x_c onto the zero set of ∂f_t. Thus, we can define $x_t^\star(0) = \lim_{\delta\to 0} x_t^\star(\delta)$. We also have that $\delta \mapsto \|x_t^\star(\delta) - x_c\|$ is monotonically decreasing (this is not hard to show and can be found in the proof of [32, Thm. 2.2.7]). Let $R(\delta; x_c) = \sup_{t\in\mathbb{N}\cup\{0\}} \|x_t^\star(\delta) - x_c\|$ for all $\delta \geq 0$. Again, since we do not vary x_c in the analysis, we write $R(\delta)$ for simplicity. Note that $R(\cdot)$ is also monotonically decreasing. Thus, if $R(0) < \infty$, then $R(\delta) < \infty$ for all $\delta \geq 0$. We will assume this is true:

$$R(0) < \infty. \qquad \text{(bounded drift)}$$

While this assumption precludes problems like the translating quadratic, it is realistic when the problem is data-dependent. For machine learning problems it is common to have normalized data and to be learning normalized weights. Then, the minimizer will, in fact, lie in a bounded set.

Let $\sigma(\delta) = \sup_{t\in\mathbb{N}} \|x_t^\star(\delta) - x_{t-1}^\star(\delta)\|$ for $\delta \geq 0$. Note that $\sigma(\delta)$ is bounded above by $2R(\delta)$, via the triangle inequality, which in turn is bounded above by $R(0) < \infty$, via the monotonicity of $R(\cdot)$.

Finally, consider the function $h(\delta) = \frac{\sigma(\delta)}{R(\delta)} - 2\left(\frac{\delta}{L}\right)^2$ for $\delta \geq 0$. $h(\cdot)$ is continuous since $x_t^\star(\cdot)$ is continuous and $R(\delta) > 0$ for all $\delta \geq 0$ (unless $x_c = x_t^\star(0)$, which would be the trivial case). Also, $h(0) = \frac{\sigma(0)}{R(0)} > 0$ and $h(L) \leq 0$. Thus, via the Intermediate Value Theorem, we have just proved the following lemma.

Lemma 5.1 *If $(f_t) \in \mathcal{S}'(L)$ has bounded drift, then $\exists\, 0 < \delta \leq L$ s.t. $\delta = L\sqrt{\frac{\sigma(\delta)}{2R(\delta)}}$.*

In Theorem 5.1, we derive a bound on the tracking gradient error in terms of $\sigma(\delta)$ and $R(\delta)$. However, since $\sigma(\delta)$ and $R(\delta)$ both depend on δ, it is impossible, without further information about the function sequence, to minimize the bound explicitly with respect to δ. The δ in Lemma 5.1 corresponds to what the minimizing δ would be if $\sigma(\delta)$ and $R(\delta)$ were constant.

Theorem 5.1 *If $(f_t) \in \mathcal{S}'(L)$ has bounded drift, then $\exists\ 0 < \delta \leq L$ such that, given x_0, ORGD(δ, x_c) constructs a sequence (x_t) with*

$$\limsup_{t\to\infty} \|\nabla f_t(x_t)\| \leq 2\sqrt{2}L\sqrt{\sigma(\delta)R(\delta)}. \tag{10}$$

Proof Let δ be as in Lemma 5.1. Observe,

$$\begin{aligned} 0 &= \nabla f_t(x_t^\star(\delta); \delta) \\ &= \nabla f_t(x_t^\star(\delta)) + \delta(x_t^\star(\delta) - x_c) \end{aligned}$$

so

$$\|\nabla f_t(x_t^\star(\delta))\| = \delta\|x_t^\star(\delta) - x_c\|.$$

Thus,

$$\begin{aligned} \|\nabla f_t(x_t)\| &\leq \|\nabla f_t(x_t^\star(\delta))\| + \|\nabla f_t(x_t) - \nabla f_t(x_t^\star(\delta))\| \\ &\leq \delta\|x_t^\star(\delta) - x_c\| + L\|x_t - x_t^\star(\delta)\| \\ &\leq \delta R(\delta) + L\|x_t - x_t^\star(\delta)\| \end{aligned}$$

so

$$\begin{aligned} \limsup_{t\to\infty} \|\nabla f_t(x_t)\| &\leq \delta R(\delta) + \frac{L(L+2\delta)\sigma(\delta)}{2\delta} \\ &= L\sqrt{2\sigma(\delta)R(\delta)} + L\sigma \\ &\leq L\sqrt{2\sigma(\delta)R(\delta)} + L\sqrt{2\sigma(\delta)R(\delta)} \\ &= 2\sqrt{2}L\sqrt{\sigma(\delta)R(\delta)}. \end{aligned}$$

□

Note that if $\sigma(0) \approx \sigma(\delta) \approx R(\delta) \approx R(0)$ then the bound in Eq. 10 is $\approx 2\sqrt{2}LR(0)$, which is of the same form but has worse constants than the bound for the algorithm which abstains from tracking (i.e., $\forall t\ x_t = x_c$):

$$\begin{aligned} \|\nabla f_t(x_t)\| &= \|\nabla f_t(x_c)\| \\ &= \|\nabla f_t(x_c) - \nabla f_t(x_t^\star(0))\| \\ &\leq L\|x_c - x_t^\star(0)\| \\ &\leq LR(0). \end{aligned}$$

Conversely, if $\sigma(\delta) \ll R(\delta)$, then δ is small so $\sigma(0) \approx \sigma(\delta)$ and $R(0) \approx R(\delta)$. In this case, Eq. 10 becomes $\approx 2\sqrt{2}\sqrt{\frac{\sigma(0)}{R(0)}}LR(0) \ll LR(0)$. Thus, we can only guarantee the usefulness of ORGD when $\sigma(\delta) \ll R(\delta)$.

6 Illustrative Numerical Examples

Our code is online at github.com/liammadden/time-varying-experiments. In this section, we consider time-varying least-squares regression and time-varying logistic regression. At each $t \in \mathbb{N} \cup \{0\}$, we are given an input matrix $A^{(t)} \in \mathbb{R}^{n \times d}$, with i-th row vector denoted by $a_i^{(t)}$, and an n-dimensional output vector $b^{(t)}$, where each $(a_i^{(t)}, b_i^{(t)})$ corresponds to a single data point. For least-squares regression, $b^{(t)} \in \mathbb{R}^n$. For logistic regression, $b^{(t)} \in \{-1, 1\}^n$. For simplicity, we assume the inputs are constant across time, i.e. $A_t = A$ for all t, while only the outputs change. This type of time-variation fits applications with time-series data where the time-dependency is not captured by any of the input features. For example, a problem may have a discrete number of states that it switches between based on a hidden variable.

The cost function for least-squares regression is

$$\begin{aligned} f_t(x) &= \frac{1}{2} \sum_{i=1}^{n} \left(\langle a_i, x \rangle - b_i^{(t)} \right)^2 \\ &= \frac{1}{2} \| Ax - b^{(t)} \|^2 \end{aligned}$$

and for logistic regression is

$$f_t(x) = \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + \exp(-b_i^{(t)} \langle a_i, x \rangle) \right).$$

The least-squares cost function is strongly convex while the logistic cost function is only strictly convex. However, it can be shown that gradient descent still achieves linear convergence when applied to the logistic cost function [24].

A "weight vector," $x \in \mathbb{R}^d$, predicts an output, b, from an input, a. For least-squares regression, $b = \langle a, x \rangle$. For logistic regression, $b = \text{sign}(\langle a, x \rangle)$; more precisely, x predicts $b = 1$ with probability $(1 + \exp(-\langle a, x \rangle))^{-1}$ and $b = -1$ with the remaining probability. The minimizer, $x_t^\star$, of the respective objective function is the "optimal" weight vector.

6.1 Least Squares Regression

For the least-squares regression we considered the case $n = 20$ and $d = 5$. We generated A by defining its singular value decomposition. For its left and right-singular vectors, we sampled two Haar distributed orthogonal matrices, $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{d \times d}$. We let its singular values be equally spaced from $1/\sqrt{\kappa}$ to 1. We generated b_t by varying $x_t^\star$ via a random walk with $\sigma = 1$ and adding a low-level of Gaussian noise (mean, 0; standard deviation, 10^{-3}) to the corresponding outputs. We initialized $x_0^\star$ as the vector of ones. For each κ, we ran the experiment 200 times and took the average of the tracking errors.

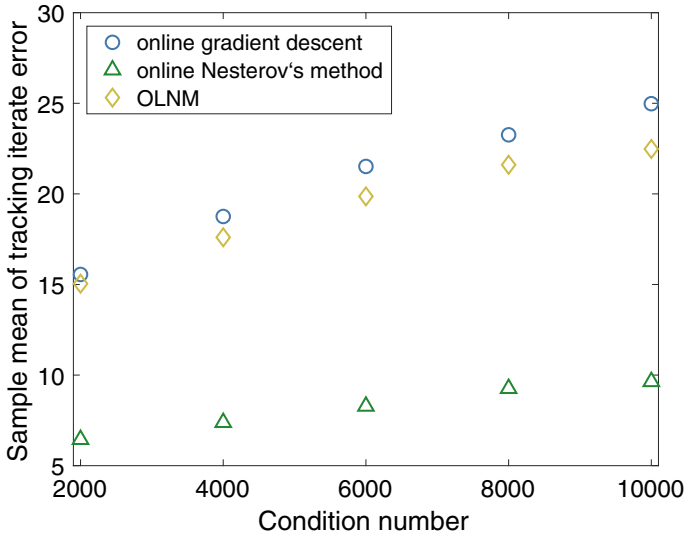

Fig. 4 Least-squares regression with random data matrix and random walk variation of the minimizer

For OLNM, we set $T = \lfloor (2+\sqrt{2})\sqrt{\kappa} \rfloor$. However, instead of updating the x iterate by the z iterate only every T iterations, we updated it every iteration. Doing so works better in practice, though we lose the theoretical guarantees of Theorem 4.2. However, the theory only requires that the x iterates do not move away from the minimizer that the z iterates are converging to. Not updating the x iterates ensures they do not move away, but it is too conservative in practice.

For all the algorithms, we initialized $x_0 = x_0^\star$ so that we wouldn't need a long "burn-in" period while waiting for convergence to the tracking error. We computed the tracking error as the maximum error over the 5th cycle of T. Figure 4 shows the results. OLNM performs better than online gradient descent, however, online Nesterov's method performs much better than both of the other methods, despite its lack of guarantees.

6.2 Logistic Regression with Streaming Data

For logistic regression we again considered the case $n = 20$ and $d = 5$. We generated A in the same way as for least-squares regression, but with singular values drawn from the uniform distribution on (0,1) and then scaled to have maximum singular value equal to $2\sqrt{L}$ (since the strong smoothness constant of the cost function is $\|A\|_2^2/4$). We initialized b_t from the Rademacher distribution and uniformly at random chose one label to flip each time step.

A classic problem that logistic regression is used for is spam classification. The problem is to predict whether an email is spam or not based on a list of features. Since emails are received in a streaming fashion, this is a time-varying problem. Thus, when an email first arrives, we may report that it is spam, but then later realize it is not,

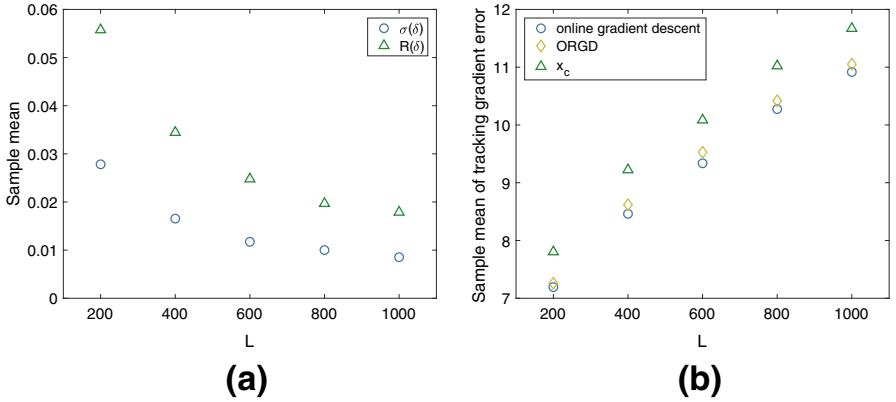

Fig. 5 Logistic regression with random data matrix and randomly flipping labels

or vice versa. Such an extension of spam classification can be incorporated into the time-varying logistic regression framework.

We let x_c be the zero vector. As a weight vector, x_c predicts 1 or -1 with equal probability. For each L, we approximated the solution to the fixed point equation $\delta = L\sqrt{\frac{\mathbb{E}(\sigma(\delta))}{\mathbb{E}(R(\delta))}}$ where the expectation is taken over A and the b_t's. Figure 5a shows how $\sigma(\delta)$ and $R(\delta)$ vary with L. The ratio between $\sigma(\delta)$ and $R(\delta)$ stays constant at ≈ 0.5.

We ran the experiment 200 times and took the average of the tracking errors. We calculated the tracking error using windows of 100 iterations and stopping when the maximum error over the last two windows was the same as over the last window. Figure 5b shows the results. Online gradient descent (which lacks theoretical tracking gradient error bounds) performs slightly better than `ORGD` and both outperform x_c. Thus, with only one label flipping every time step, it is still possible to track the solution. However, if we sufficiently increased the number of labels that flip every time step, then it would be better to just guess the labels, as x_c does.

7 Conclusions

By categorizing classes of functions based on the minimizer variation, this paper was able to generalize results from batch optimization to the online setting. These results are important for time-varying optimization, especially for applications in power systems, transportation systems, and communication networks. We showed that fast convergence does not necessarily lead to small tracking error. For example, online Polyak's method can diverge even when the minimizer variation is zero. On the other hand, online gradient descent is guaranteed to have a tracking error that does not grow more than linearly with the minimizer variation. We also gave a universal lower bound for online first-order methods and showed that `OLNM` achieves it up to a constant factor. It is a future research direction to consider more deeply the connection to the long-steps of interior-point methods and see if a satisfactory criteria can be found for adaptively

deciding when to long-step. Perhaps, this enquiry may eventually lead to error bounds for online Nesterov's method itself.

Acknowledgements All three authors gratefully acknowledge support from the NSF program "AMPS-Algorithms for Modern Power Systems" under Award # 1923298.

References

1. Allen-Zhu, Z., Hazan, E.: Optimal black-box reductions between optimization objectives (2016). [Online] Available at: arXiv:1603.05642
2. Allen-Zhu, Z., Orecchia, L.: Linear coupling: An ultimate unification of gradient and mirror descent. In: Proceedings of the 8th Innovations in Theoretical Computer Science, ITCS '17 (2017). Full version available at arXiv:1407.1537
3. Arjevani, Y., Shalev-Shwartz, S., Shamir, O.: On lower and upper bounds in smooth and strongly convex optimization. J. Mach. Learn. Res. **17**(1), 4303–4353 (2016)
4. Aujol, J., Dossal, C.: Stability of over-relaxations for the forward-backward algorithm, application to fista. SIAM J. Optim. **25**(4), 2408–2433 (2015)
5. Aybat, N., Fallah, A., Gürbüzbalaban, M., Ozdaglar, A.: Robust accelerated gradient methods for smooth strongly convex functions. SIAM J. Optim. **30**, 717–751 (2020)
6. Bauschke, H., Combettes, P.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 2nd edn. Springer-Verlag, New York (2017)
7. Beck, A.: First-Order Methods in Optimization. MOS-SIAM Series on Optimization (2017)
8. Bernstein, A., Dall'Anese, E., Simonetto, A.: Online primal-dual methods with measurement feedback for time-varying convex optimization. IEEE Trans. Signal Process. **67**(8), 1978–1991 (2019)
9. Bertsekas, D., Tsitsiklis, J.: Gradient convergence in gradient methods with errors. SIAM J. Optim. **10**(3), 627–642 (2000)
10. Besbes, O., Gur, Y., Zeevi, A.: Non-stationary stochastic optimization. Oper. Res. **63**(5), 1227–1244 (2015)
11. Bianchin, G., Pasqualetti, F.: A network optimization framework for the analysis and control of traffic dynamics and intersection signaling. In: IEEE Conference on Decision and Control, pp. 1017–1022. IEEE (2018)
12. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
13. Bubeck, S.: Convex Optimization: Algorithms and Complexity. In: Foundations and Trends in Machine Learning, (2015)
14. Chen, J., Lau, V.K.N.: Convergence analysis of saddle point problems in time varying wireless systems: Control theoretical approach. IEEE Trans. Signal Process. **60**(1), 443–452 (2012)
15. Cyrus, S., Hu, B., Scoy, B.V., Lessard, L.: A robust accelerated optimization algorithm for strongly convex functions. In: IEEE Annual American Control Conference, pp. 1376–1381 (2018)
16. Dall'Anese, E., Simonetto, A., Becker, S., Madden, L.: Optimization and learning with information streams: Time-varying algorithms and applications. IEEE Signal Processing Magazine (2020). To appear; see arXiv preprint arXiv:1910.08123
17. Dall'Anese, E., Simonetto, A.: Optimal power flow pursuit. IEEE Trans. Smart Grid **9**, 942–952 (2018)
18. Devolder, O., Glineur, F., Nesterov, Y.: First-order methods of smooth convex optimization with inexact oracle. Math. Program. **146**, 37 (2014)
19. Dixit, R., Bedi, A.S., Tripathi, R., Rajawat, K.: Online learning with inexact proximal online gradient descent algorithms. IEEE Trans. Signal Process. **67**(5), 1338–1352 (2019)
20. Drori, Y., Teboulle, M.: Performance of first-order methods for smooth convex minimization: a novel approach. Math. Program. **145**, 541 (2012)
21. Hall, E.C., Willett, R.M.: Online convex optimization in dynamic environments. IEEE J. Select. Top. Signal Process. **9**(4), 647–662 (2015)
22. Hazan, E., Agarwal, A., Kale, S.: Logarithmic regret algorithms for online convex optimization. Mach. Learn. **69**(2), 169–192 (2007)
23. Jadbabaie, A., Rakhlin, A., Shahrampour, S., Sridharan, K.: Online optimization: competing with dynamic comparators. PMLR **38**, 398–406 (2015)

24. Karimi, H., Nutini, J., Schmidt, M.: Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In: Machine Learning and Knowledge Discovery in Database - European Conference, pp. 795–811 (2016)
25. Kim, D., Fessler, J.: Optimized first-order methods for smooth convex minimization. Math. Program. **159**, 81 (2016)
26. Koshal, J., Nedic, A., Shanbhag, U.: Multiuser optimization: distributed algorithms and error analysis. SIAM J. Optim. (2011). https://doi.org/10.1137/090770102
27. Lessard, L., Recht, B., Packard, A.: Analysis and design of optimization algorithms via integral quadratic constraints. SIAM J. Optim. **26**, 57–95 (2016)
28. Madden, L., Becker, S., Dall'Anese, E.: Online sparse subspace clustering. In: IEEE Data Science Workshop (2019)
29. Mokhtari, A., Shahrampour, S., Jadbabaie, A., Ribeiro, A.: Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 7195–7201 (2016)
30. Nemirovsky, A., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. Wiley, Hoboken (1983)
31. Nesterov, Y.: A method for solving the convex programming problem with convergence rate o(1/k2) (1983)
32. Nesterov, Y.: Lectures on Convex Optimization, *Springer Optimization and Its Applications*, vol. 137, 2 edn. Springer, Switzerland (2018)
33. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course, Applied Optimization, vol. 87. Kluwer, Boston (2004)
34. Nocedal, J., Wright, S.: Numerical Optimization. Operations Research and Financial Engineering, 2nd edn. Springer, Berlin (2006)
35. O'Donoghue, B., Candes, E.: Adaptive restart for accelerated gradient schemes. Found. Comput. Math. **53**, 715 (2015)
36. Polyak, B.: Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Math. Phys. **4**, 1–17 (1964)
37. Popkov, A.Y.: Gradient methods for nonstationary unconstrained optimization problems. Autom. Remote Control **66**(6), 883–891 (2005)
38. Rockafeller, R.: Augmented lagrangians and applications of the proximal point algorithm in convex programming. Math. Oper. Res. **1**(2), 97 (1976)
39. Ryu, E., Boyd, S.: A primer on monotone operator methods. Appl. Comput. Math. **15**(1), 3–43 (2016)
40. Schmidt, M., Roux, N., Bach, F.: Convergence rates of inexact proximal-gradient methods for convex optimization. In: Advances in neural information processing systems, pp. 1458–1466 (2011)
41. Shalev-Shwartz, S.: Online Learning and Online Convex Optimization. Found. Trends Mach. Learn. **4**, 107 (2012)
42. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: From theory to algorithms. Cambridge University Press, Cambridge (2014)
43. Simonetto, A., Leus, G.: Double smoothing for time-varying distributed multiuser optimization. In: IEEE Glob. Conf. Signal Inf. Process. (2014)
44. Simonetto, A.: Time-varying convex optimization via time-varying averaged operators (2017). [Online] Available at: arXiv:1704.07338
45. Tang, Y., Dvijotham, K., Low, S.: Real-time optimal power flow. IEEE Trans. Smart Grid **8**, 2963–2973 (2017)
46. Taylor, A., Hendrickx, J., Glineur, F.: Smooth strongly convex interpolation and exact worst-case performance of first-order methods. Math. Program. **161**, 307 (2016)
47. Tseng, P.: Approximation accuracy, gradient methods, and error bounds for structured convex optimization. Math. Program. **125**, 263 (2010)
48. Villa, S., Salzo, S., Baldassarre, L., Verri, A.: Accelerated and inexact forward-backward algorithms. SIAM J. Optim. **23**(3), 1607–1633 (2013)
49. Yang, T., Zhang, L., Jin, R., Yi, J.: Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient. In: International Conference on Machine Learning (2016)
50. Yang, T.: Accelerated gradient descent and variational regret bounds (2016)