Partially Lazy Gradient Descent for Smoothed Online Learning

Naram Mhaisen[1] **George Iosifidis**[1]
[1]Faculty of Electrical Engineering, Mathematics and Computer Science.
TU Delft, Netherlands.

Abstract

We introduce k-LAZYGD, an online learning algorithm that bridges the gap between greedy Online Gradient Descent (OGD, for $k = 1$) and lazy GD/dual-averaging (for $k = T$), creating a spectrum between reactive and stable updates. We analyze this spectrum in Smoothed Online Convex Optimization (SOCO), where the learner incurs both hitting and movement costs. Our main contribution is establishing that laziness is possible without sacrificing hitting performance: we prove that k-LAZYGD achieves the optimal dynamic regret $\mathcal{O}(\sqrt{(P_T+1)T})$ for any laziness slack k up to $\Theta(\sqrt{T/P_T})$, where P_T is the comparator path length. This result formally connects the allowable laziness to the comparator's shifts, showing that k-LAZYGD can retain the inherently small movements of lazy methods without compromising tracking ability. We base our analysis on the Follow the Regularized Leader (FTRL) framework, and derive a matching lower bound. Since the slack depends on P_T, an ensemble of learners with various slacks is used, yielding a method that is provably stable when it can be, and agile when it must be.

1 INTRODUCTION

We study the problem of *Smoothed Online Convex Optimization* (SOCO) (Cesa-Bianchi et al., 2013; Chen et al., 2018b; Zhao et al., 2020), where a learner interacts with a possibly adversarial environment over T rounds. At each round $t = 1, \dots, T$, the learner selects an action from a convex compact set $\boldsymbol{x}_t \in \mathcal{X}$, then incurs a *hitting cost* $f_t(\boldsymbol{x}_t)$ and a *movement cost* $m(\boldsymbol{x}_t, \boldsymbol{x}_{t-1})$ penalizing changes in successive decisions. We follow the OCO convention in which the hitting cost function $f_t(\cdot)$ is revealed only *after* $\boldsymbol{x}_t$ is chosen. As is standard in SOCO, we set the movement cost to be the ℓ_2 distance, $m(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}) = \|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|$, though our analysis naturally extends to time-varying convex movement costs that are Lipschitz-continuous.

Our metric is *dynamic regret with switching cost*:

$$\mathcal{R}_T \doteq \sum_{t=1}^{T} \big(f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t)\big) + \sum_{t=1}^{T-1} \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|. \quad (1)$$

The first sum is the *dynamic regret* against an arbitrary comparator sequence $\{\boldsymbol{u}_t\}_{t=1}^T$, and the second one is the learner's *switching cost*. The switching cost of the comparator is termed "path length" $P_T \doteq \sum_{t=1}^{T-1} \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$.[1]

The $\mathcal{R}_T$ metric jointly measures the learner's hitting performance, that is, how well $\boldsymbol{x}_t$ competes with $\boldsymbol{u}_t$ in terms of $f_t(\cdot)$, and the stability of its actions over time. The goal is to design an algorithm ALG achieving $\mathcal{R}_T^{\text{ALG}} = \mathcal{O}(\sqrt{(P_T+1)T})$, where $\mathcal{R}_T^{\text{ALG}}$ refers to the regret defined in (1) when the actions $\boldsymbol{x}_t$ are generated via ALG. Such a rate is minimax-optimal: it matches the best possible bound known for dynamic regret (Zhang et al., 2018, Thm. 2), which applies to the more difficult metrics that include switching costs.

Two variants. Among the foundational methods in OCO, OGD (Zinkevich, 2003) plays a central role. It admits two classical update rules: a greedy variant and a lazy variant. Let $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$ be a subgradient of $f_t(\cdot)$ at $\boldsymbol{x}_t$, and let $\sigma > 0$ denote a *regularization* parameter. The greedy variant (often simply called gradient descent) takes a step from the previous iterate in the direction of $-\boldsymbol{g}_t$, scaled by $1/\sigma$,[2] and then

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

[1]Some formulations subtract P_T from $\mathcal{R}_T$, but this will not change the final order. See, e.g., Zhang et al. (2021).

[2]Most presentations of GD use a learning rate parameter η. Here, we instead write $\sigma = 1/\eta$, which is more natural for our later derivations in the FTRL framework.

projects back onto the feasible set:

$$\boldsymbol{x}_{t+1}^{\mathrm{G}} = \Pi\left(\boldsymbol{x}_t^{\mathrm{G}} - \frac{1}{\sigma}\boldsymbol{g}_t\right), \qquad \text{(GD)}$$

where $\Pi(\cdot)$ denotes the Euclidean projection onto the convex set $\mathcal{X}$: $\Pi(\boldsymbol{z}) \doteq \arg\min_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{z}\|$.

In contrast, the lazy variant, known as dual averaging (Xiao, 2009), first accumulates *all* past subgradients and then projects their scaled direction, making the update independent of the previous iterate:

$$\boldsymbol{x}_{t+1}^{\mathrm{L}} = \Pi\left(-\frac{\boldsymbol{g}_{1:t}}{\sigma}\right), \qquad \text{(LazyGD)}$$

where $\boldsymbol{g}_{a:b}$ denotes the sum $\sum_{\tau=a}^{b} \boldsymbol{g}_\tau$. The two variants may coincide in both their update rules and regret guarantees. In the unconstrained setting, $\mathcal{X} = \mathbb{R}^d$, the projection operator reduces to the identity, and the greedy update (GD) can be simply unrolled into the lazy one (LazyGD).[3] In constrained domains, the iterates generally differ, yet both variants achieve the same optimal *static* regret (i.e., when $\boldsymbol{u}_t = \boldsymbol{u}^\star, \forall t$) of $\mathcal{O}(\sqrt{T})$. They also share the same *worst-case* switching bound: $\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| \le G/\sigma$, where G is the Lipschitz constant, $\|\boldsymbol{g}_t\| \le G, \forall t$. This bound is a direct consequence of the non-expansiveness of the projection operator (see Appendix B). Yet, empirical evidence shows that, in many cases, this switching cost bound is considerably looser for LazyGD than it is for GD.

Switching behavior. The lower switching cost of LazyGD is arguably intuitive; accumulating gradients over time is inherently more stable than reacting to each new one individually. This distinction becomes evident in the illustrative examples of Fig. 1. In example (i), driven by an initial set of fixed gradients, both variants incur the same switching cost required to reach the boundary of the ℓ_1-ball, paying a switching cost of 1. However, once the gradients begin rotating thereafter, the greedy update of GD continues to chase the most recent direction, leading to a continually increasing, though sublinear, movement cost. In contrast, such rotations do not alter the optimum when appended to the aggregated gradients, causing the lazy iterates to stall. As a result, the lazy variant pays *zero* switching cost after reaching the boundary. Moreover, since gradients are orthogonal, chasing the most recent one yields no advantage in hitting costs.

Example (ii) highlights another advantage. Here, the feasible set is the ℓ_2-ball, and every new gradient forces both variants to move. Gradients have a fixed horizontal component and an alternating vertical one. The greedy update responds fully to each incoming gradient, as dictated by its update rule. The lazy update, however, attenuates movement in proportion to the magnitude of the *aggregated* gradients, effectively leveraging their correlation. Under sufficient gradient correlation, as in this example, the movement of the lazy variant is greatly reduced. As in Example (i), chasing the most recent gradient yields no improvement in hitting cost since only the fixed horizontal component contributes to loss reduction. Additional details on both examples are provided in Appendix C.

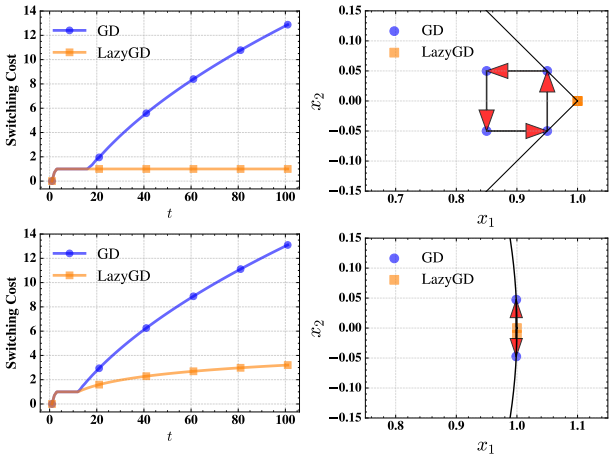

Figure 1: Switching in example (i, top), showing *staleness*, and example (ii, bottom), showing *stability*. Left: switching cost. Right: Snapshots over 4 (top) and 2 (bottom) rounds: greedy updates move continuously, whereas lazy updates remain still or move minimally.

These two examples illustrate each a key property underlying LazyGD's superior switching behavior: (i) *iterate staleness*: updates halt as long as the incoming gradient does not alter the minimizer of their accumulated sum; and (ii) *iterate stability*: even when the minimizer is altered, the movement's magnitude is attenuated proportionally to the size of the accumulated sum. And while in the worst case GD and LazyGD share the same movement bound of G/σ, the difference lies in sensitivity: in the former, each new gradient can trigger a movement of such size *regardless* of its effect on the *direction* of accumulated gradients, or on the *size* thereof, whereas LazyGD suppresses movement whenever either of the above properties is in effect.

Limits of lazy updates. Despite their stable behavior and minimal movement, lazy methods have not been used in SOCO, and for a good reason. An impossibility result by Jacobsen and Cutkosky (2022) shows that *any* method of the LazyGD form suffers *linear* dynamic regret, hence $\mathcal{R}_T = \Omega(T)$. This lower bound holds for *any* non-zero path length. That is, even when the environment changes minimally, lazy methods *fail* dramatically in the hitting cost. As recently observed

[3] If σ is not constant, this equivalence does not hold even in unconstrained settings. See Liu et al. (2025, Sec. 1).

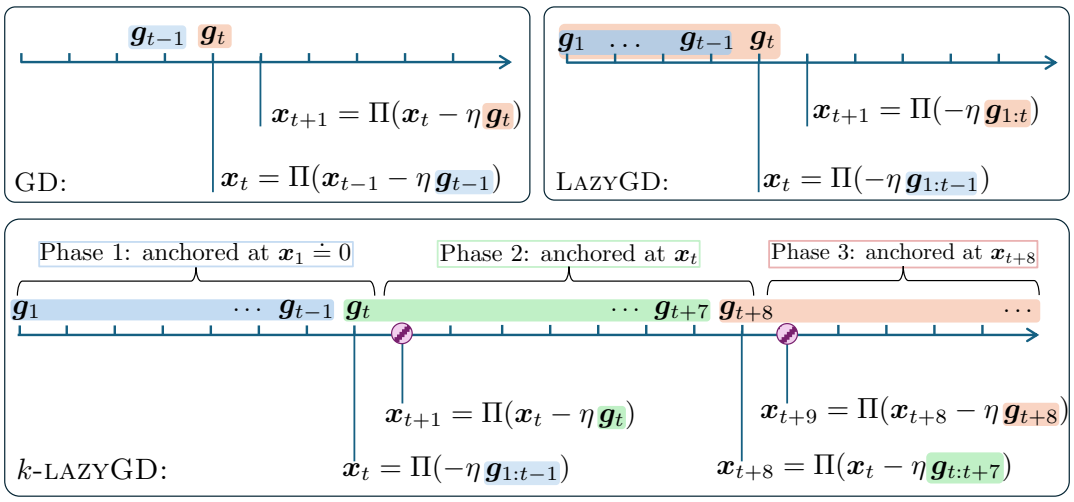

Figure 2: Top-left: GD updates from the previous iterate using only the most recent gradient. Top-right: LazyGD updates from the origin (assumed 0) using cumulative gradients. Bottom: k-LazyGD combines these two views: it partitions time into phases (of length $k = 8$ here), anchors each phase at its starting iterate, and updates from the anchor using the cumulative gradients within that phase. We use $\eta = 1/\sigma$ for clarity.

in (Mhaisen and Iosifidis, 2025), the root cause is that accumulating past gradients can mask subtle but important changes in the gradient direction, rendering the lazy iterate effectively unresponsive to shifts that are critical for minimizing dynamic regret.

This barrier highlights a tension between stability and responsiveness. Minimizing switching favors aggregating gradients over time, whereas minimizing dynamic regret requires continual adaptation to each new gradient. Neither extreme is ideal: greedy updates are overly reactive, while fully lazy ones can fail to track even simple non-stationary benchmarks. This paper therefore asks: *what is the maximum laziness compatible with optimal dynamic regret?* That is, how far can a learner accumulate past gradients while still tracking non-stationary comparators.

Unifying view. Motivated by this question, we explore the intermediate space of *partial* laziness, interpolating between greedy and lazy updates. One natural approach is to study an update of the form:

$$\begin{aligned} &x_{t+1} = \Pi\left(x_{t-n_t} - \frac{g_{t-n_t:t}}{\sigma}\right) \\ &n_t = (t-1) \mod (k), \end{aligned} \qquad (k\text{-LazyGD})$$

for a given *laziness slack* $k \in \{1, \dots, T\}$. Recall that $g_{t-n_t:t}$ is the sum of g_τ from $\tau = t - n_t$ to $\tau = t$. This scheme recovers the greedy update of GD when $k = 1$ (since in this case $n_t = 0, \forall t$), and the lazy update of LazyGD when $k = T$,[4] (since in this case $n_t = t - 1, \forall t$). For intermediate values of k, the rule k-LazyGD partitions the horizon into T/k phases of length k each[5]. Within each phase, gradients are accumulated as in LazyGD, but, crucially, the accumulation is reinitialized at the (dual mapping of) the last iterate from the preceding phase. Fig.2 visualizes an example of how the iterates are updated. To our knowledge, this phased form, and the advantages thereof, were not investigated in the literature.

Contributions. Our first result is showing that the non-standard update of k-LazyGD can be cast as an instance of the FTRL framework using the "pruning" trick from Mhaisen and Iosifidis (2025), who showed that responsiveness is achieved via a principled way of pruning past gradients at a given t. Here, we generalize this mechanism: we accumulate as many gradients as possible, hence preserving the staleness and stability properties, and *only* trigger pruning when further accumulation would otherwise hinder the algorithm's agility and affect the dynamic regret. This FTRL reduction also enables us to reuse existing analyses in a modular way, yielding a cleaner proof structure.

We then turn to the question of how "much" laziness (stability) can be tolerated without compromising the dynamic regret. To that end, we derive a universal lower bound for k-LazyGD methods that extends the impossibility result of Jacobsen and Cutkosky (2022, Thm. 1), which corresponds to the extreme case $k = T$. Our result reveals that the laziness slack may be as large as $k = \Theta(\sqrt{T/P_T})$ while still remaining responsive enough to maintain optimal dynamic regret. Be-

[4] Assuming the null initialization $x_1 = 0$.

[5] Only the last phase is of length $T \mod k$.

yond this threshold, the learner becomes insufficiently responsive to track non-stationary comparators, an insight that is new to the OCO literature.

Next, we quantify the benefit of allowing slacks larger than the current k=1 standard in SOCO. We formalize the two properties of lazy updates: *staleness*, which prevents unnecessary movement, and *stability*, which attenuates unavoidable movement, both of which benefit from a larger k. We then show that k-LAZYGD still achieves optimal dynamic regret, while having these two properties. Our analysis extends recent work on the FTRL framework with dynamic comparators, and reveals that optimal regret can be attained without *fully* discarding the structural benefits of laziness.

Lastly, the optimal choice of laziness slack depends on the comparator's non-stationarity, quantified in P_T. While one could specify an upper bound on P_T a priori, achieving simultaneous optimality across all comparator sequences necessitates adaptivity. This challenge is addressed by the ensemble (meta-learning) framework of Zhao et al. (2024), where multiple instances of GD with different *learning rates* are run in parallel and aggregated. The dependence of the *laziness slack* on P_T is analogous. Accordingly, we adopt the meta-learning principle, but across an ensemble of k-LAZYGD instances, each with a distinct (k, σ) pair.

2 k-LAZYGD AS AN FTRL INSTANCE

We introduce k-LAZYGD as an instance of the general FTRL framework. This is achieved via a flexible pruning rule, which enables us to control the amount of history (past gradients) retained. Formally, the FTRL update, executed on the linearized extended real value function $\bar{f}_t(\cdot) = f_t(\cdot) + I_{\mathcal{X}}(\cdot)$, with an ℓ_2 regularizer is:

$$\hat{\boldsymbol{x}}_{t+1} = \Pi(\boldsymbol{y}_{t+1}), \quad \boldsymbol{y}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{x}\|^2, \quad (2)$$

where $\boldsymbol{p}_{1:t}$ is referred to as the state vector and is calculated as the aggregation of the subgradients of $\bar{f}_t(\boldsymbol{x}_t)$:

$$\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I,$$
$$\boldsymbol{g}_t \in \partial f_t(\hat{\boldsymbol{x}}_t), \quad \boldsymbol{g}_t^I \in \partial I_{\mathcal{X}}(\hat{\boldsymbol{x}}_t) = \mathcal{N}_{\mathcal{X}}(\hat{\boldsymbol{x}}_t).$$

$\mathcal{N}_{\mathcal{X}}(\boldsymbol{x})$ denotes the normal cone of the set $\mathcal{X}$ at $\boldsymbol{x}$.[6] The solve-then-project routine in (2) can also be equivalently written as a direct minimization over $\mathcal{X}$ (McMahan, 2017, Thm. 11):

$$\hat{\boldsymbol{x}}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{x}\|^2. \quad (3)$$

[6]The *normal cone* at $\boldsymbol{x} \in \mathcal{X}$ is the set of vectors that form a non-acute angle with every feasible direction from $\boldsymbol{x}$: $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}) \doteq \{\boldsymbol{g} \,|\, \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}\rangle \le 0, \forall \boldsymbol{y} \in \mathcal{X}\}$.

Note that the standard choice is $\boldsymbol{g}_t^I = 0, \forall t$, which leads to the well-known FTRL iteration. However, we observe that this is not the only choice. Specifically, we propose the following: for $t = 1$, set $\boldsymbol{g}_1^I = 0$. $\forall t \ge 2$:

$$\boldsymbol{g}_t^I = \begin{cases} -\boldsymbol{p}_{1:t-1} - \sigma\hat{\boldsymbol{x}}_t & \text{if } (\boldsymbol{y}_t \notin \mathcal{X} \ \wedge \ n_t = 0), \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where n_t is a counter used to control the *rate* of pruning to be at most every k steps: $n_t = (t-1) \bmod (k)$.

Note that the choices made in (4) for $\boldsymbol{g}_t^I$ are always valid; when $\boldsymbol{y}_t \notin \mathcal{X}$, the projection lands on the boundary: $\boldsymbol{x}_t \in \mathbf{bd}(\mathcal{X})$, since $\mathcal{X}$ is compact. The normal cone $\mathcal{N}_{\mathcal{X}}(\hat{\boldsymbol{x}}_t)$ then contains nonzero vectors. In particular, it includes $-(\boldsymbol{p}_{1:t-1} + \sigma\hat{\boldsymbol{x}}_t)$. This follows directly from the optimality condition for the constrained update in (3) (see Beck (2017, Thm. 3.67)). In all cases, 0 is always a valid subgradient since all cones must contain the zero vector.

The choice of $\boldsymbol{g}_t^I$ makes explicit what we mean by *pruning*: discarding the accumulated state vector $\boldsymbol{p}_{1:t-1}$ and replacing it with a representative one $\sigma\hat{\boldsymbol{x}}_t$.[7]

Theorem 1. *The iterates of k-LAZYGD, $\{\boldsymbol{x}_t\}_{t=1}^T$, coincide with those of the FTRL routine defined above in* (2) *and* (4)*, $\{\hat{\boldsymbol{x}}_t\}_{t=1}^T$. Namely:*

$$\hat{\boldsymbol{x}}_{t+1} = \Pi\big(\boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t:t}}{\sigma}\big), \quad n_t = (t-1) \bmod (k).$$

Proof. We proceed by induction. Let the initialization be equal $\hat{\boldsymbol{x}}_1 = \boldsymbol{x}_1 = 0$, and let the Induction Hypothesis (IH) be $\hat{\boldsymbol{x}}_t = \boldsymbol{x}_t$, for some t. We now prove the inductive step: $\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{x}_{t+1}$.

Note that both, k-LAZYGD and FTRL in (2), perform a Euclidean projection of a given vector. Since the projection is unique, it suffices to show the equivalence of those vectors. That is, we want to show that:

$$\boldsymbol{y}_{t+1} = \boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t:t}}{\sigma}. \quad (5)$$

By the optimality conditions of the unconstrained minimization in (2) we know that, for all t,

$$\boldsymbol{y}_{t+1} = -\frac{\boldsymbol{p}_{1:t}}{\sigma}. \quad (6)$$

Hence, by substituting (6) in (5), it suffices to show:

$$-\frac{\boldsymbol{p}_{1:t}}{\sigma} \ = \ \boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t:t}}{\sigma}. \quad (7)$$

We distinguish two cases based on n_t.

Case 1: $n_t = 0$. *subcase 1.a:* $\boldsymbol{g}_t^I = 0$. From the condition in (4), this implies $\boldsymbol{y}_t \in \mathcal{X}$, and hence the projection $\hat{\boldsymbol{x}}_t$ is $\boldsymbol{y}_t$. It follows that

$$\hat{\boldsymbol{x}}_t \overset{(6)}{=} \frac{\boldsymbol{p}_{1:t-1}}{\sigma} \overset{\text{IH}}{=} \boldsymbol{x}_t.$$

[7]This is the dual mapping of $\boldsymbol{x}_t$, with map $\nabla\frac{1}{2}\|\boldsymbol{x}\|^2$.

(7) follows by adding $-\frac{\boldsymbol{g}_t}{\sigma}$ to both sides and recalling that in this subcase, we have $n_t = 0$ and $\boldsymbol{g}_t^I = 0$.

subcase 1.b: $\boldsymbol{g}_t^I = -\boldsymbol{p}_{1:t-1} - \sigma\hat{\boldsymbol{x}}_t$. In this case, we write

$$\frac{-\boldsymbol{p}_{1:t}}{\sigma} = \frac{-\boldsymbol{p}_{1:t-1}}{\sigma} - \frac{\boldsymbol{g}_t}{\sigma} - \frac{\boldsymbol{g}_t^I}{\sigma} = \hat{\boldsymbol{x}}_t - \frac{\boldsymbol{g}_t}{\sigma} \overset{\text{IH}}{=} \boldsymbol{x}_t - \frac{\boldsymbol{g}_t}{\sigma},$$

showing that (7) follows from the subcase conditions. From both subcases, we have that for any index τ,

$$n_\tau = 0 \;\Rightarrow\; \boldsymbol{x}_\tau - \frac{\boldsymbol{g}_\tau}{\sigma} = \frac{-\boldsymbol{p}_{1:\tau}}{\sigma}. \tag{8}$$

Case 2: $0 < n_t \leq k-1$ In this case, let $\tau \doteq t - n_t$ denote the last round in which pruning is attempted. By construction, we have $n_\tau = 0$, so Case 1 applies at round τ. Hence, from (8), we obtain:

$$-\frac{\boldsymbol{p}_{1:t-n_t}}{\sigma} \;=\; \boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t}}{\sigma}. \tag{9}$$

Now observe that by the pruning rule (4), no pruning occurs in the $k-1$ steps following τ. That is, pruning is inactive on the interval $\{\tau+1, \ldots, \tau+(k-1)\}$. Substituting $\tau = t - n_t$, we find that pruning is inactive during the interval $\{t-n_t+1, \ldots, t\}$. Thus

$$\boldsymbol{g}_{t-n_t+1}^I = \cdots = \boldsymbol{g}_t^I = 0.$$

Using this fact, and adding $-\boldsymbol{g}_{t-n_t+1:t}/\sigma$ to both sides of (9), we arrive at (7). □

In summary, k-LAZYGD is an FTRL instance with a specific choice of linearization $\boldsymbol{p}_t$. This perspective aids both the analysis and further generalization.

3 A LOWER BOUND FOR LAZY ALGORITHMS

Since the presented FTRL variant is not covered by existing lower bounds[8], we construct an adversarial instance showing that any k-LAZYGD algorithm must incur dynamic regret linear in k. Building on (Jacobsen and Cutkosky, 2022, Thm. 2), we generalize their argument for any laziness slack k and path length P_T.

Theorem 2. *For any k-LAZYGD learner and any $\tau \in [1, 2R(T-1)]$, where R is the radius of $\mathcal{X}$, there exists a comparator $\{\boldsymbol{u}_t\}_{t=1}^T$, with $P_T = \lfloor\tau\rfloor$, and a sequence $\{f_t(\cdot)\}_{t=1}^T$ such that $\mathcal{R}_T = \Omega(k\,P_T)$.*

Proof. Let the decision space be the line segment $\mathcal{X} = [-1, 1]$. Let the set of sub-intervals be defined as

$$\mathcal{T}_i = [(i-1)k+1, \ldots, ik] \quad \text{for } i = 1, \ldots, \lfloor T/k \rfloor.$$

[8]Except the universal $\sqrt{(P_T+1)T}$ (independent of k).

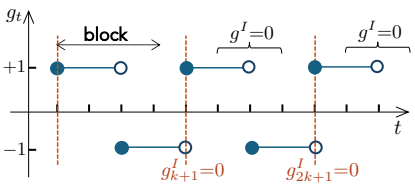

Figure 3: The constructed sequence ($k = 4$). g_t^I is zero within blocks by (4), and at block starts since $y = 0$ then.

For every interval $\mathcal{T}_i$, the adversary defines the cost as $f_t(x) = g_t\, x$, where g_t is a simple square wave:

$$g_t = \begin{cases} +1, & \text{for } t \in \left[(i-1)k+1,\ (i-1)k+\frac{k}{2}\right], \\ -1, & \text{for } t \in \left[(i-1)k+\frac{k}{2}+1,\ ik\right], \end{cases}$$

where k is assumed even w.l.o.g.[9]. Concatenating all intervals $\{\mathcal{T}_i\}_i$ yields the full horizon.

By construction, the cost sequence is k-periodic ($g_t = g_{t+k}$) with zero sum per base period. Hence, any window of length ik sums to zero:

$$\sum_{t=j+1}^{j+ik} g_t = 0,\ i{\in}\{1, \ldots, \lfloor T/k \rfloor\},\ j{\in}\{0, \ldots, T{-}ik\}. \tag{10}$$

Moreover, since the sequence begins with the positive half, sums starting at 1 are nonnegative: $g_{1:t} \geq 0,\ \forall t$.

We claim that, with the cost sequence above, and for any (k-LAZYGD) variant, the iterates satisfy $x_t \leq 0$ for all t. It is enough to show that $g_t^I = 0$ for every t, because in such case the update becomes

$$x_t = \Pi(y_t) \overset{(a)}{=} \min\left(1, \max(-1, y_t)\right) \overset{(2)}{=} \min\Big(1, \max\Big(-1, \frac{-g_{1:t-1}}{\sigma}\Big)\Big) \overset{(b)}{=} \max\Big(-1, \frac{-g_{1:t-1}}{\sigma}\Big) \leq 0.$$

Here, (a) uses that projecting onto $[-1, 1]$ is a sign-preserving clipping, and (b) follows from $g_{1:t} \geq 0$.

It remains to show that $g_t^I = 0$ for all t. Indeed, the construction in (4) implies that g_t^I can only be nonzero at indices of the form $t = jk{+}1 \leq T$ for some $j \geq 0$:

$$g_t^I = 0 \quad \forall\, t \notin \mathcal{I} \doteq \{\, jk{+}1 : j \geq 0,\ jk{+}1 \leq T \,\}. \tag{11}$$

The proof therefore reduces to showing $g_t^I = 0$ for all $t \in \mathcal{I}$. These indices mark the start of each block (Fig. 3). We use induction over blocks.

base case: $g_1^I = 0$ is true by definition. *Induction Hypothesis (IH):* $g_{lk+1}^I = 0,\ l \in \{1, \ldots, j\}$. Now we show the inductive step. Intuitively, we show that at each pruning opportunity (the dashed lines in Fig. 3), the unconstrained iterate in (2) has already returned to the feasible set, so no pruning occurs: $y_{(j+1)k+1}$

[9]For an odd k, we construct the square wave using the even $k-1$, and set the k^{th} cost to 0. We defer this detail to Appendix E since it is mainly index manipulation.

$$= \frac{-p_{1:(j+1)k}}{\sigma} \overset{(11)}{=} \frac{1}{\sigma}\Big(-\underbrace{p_{1:jk+1}}_{\substack{\text{inside \& start}\\ \text{of prev. blocks}}} - \overbrace{g_{jk+2:(j+1)k}}^{\text{inside last block}} \Big)$$

$$\overset{\text{(IH \& (11))}}{=} \frac{-g_{1:(j+1)k}}{\sigma} \overset{(10)}{=} 0.$$

This shows that $y_{(j+1)k+1} \in \mathcal{X}$, implying that $g^I_{(j+1)k+1} = 0$, and completing the inductive step.

We proceed to calculate the regret of these non-positive actions. Let $\tau' \doteq \lfloor \tau \rfloor$ denote our comparator's switch budget. We declare $\mathcal{T}_1$ *active*, and then mark a block active if we can spend *two* switch units inside it. Thus, the number of active blocks A is

$$A = 1 + \left\lfloor \frac{\tau' - 1}{2} \right\rfloor = \left\lceil \frac{\tau'}{2} \right\rceil$$

Fix an active block $\mathcal{T}_i$. By the argument above, $x_t \le 0$.

$$\sum_{t\in\mathcal{T}_i} g_t x_t \ge \sum_{t=(i-1)k+1}^{(i-1)k+k\ 2} g_t x_t \ \ge\ -\sum_{t=1}^{k\ 2} 1 \ =\ -\frac{k}{2}.$$

where the first equality is because $g_t x_t$ is positive in the second half of $\mathcal{T}_i$.

For the comparator cost, in $\mathcal{T}_1$, the comparator starts at $u_t = -1$ and switches to $+1$ at the midpoint, hence incurring loss -1 at every step: $\sum_{t\in\mathcal{T}_1} g_t u_t = -k$. In every subsequent active block $\mathcal{T}_i$ ($i \ge 2$), the comparator spends *two* switches: reset $u_t = -1$ at the block's first step and flip to $+1$ at its midpoint. This again yields $\sum_{t\in\mathcal{T}_i} g_t u_t = -k$. Summing over active blocks

$$\mathcal{R}_T \ge \frac{k}{2} A = \frac{k}{2} \left\lceil \frac{\tau'}{2} \right\rceil \ge \frac{k}{4} \tau',$$

from which the lower bound follows. □

Remarks. The result clarifies why any $P_T \ge 1$ forces linear regret for LazyGD (k=T). It also reveals a gap in admissible laziness: while GD (k=1) achieves $\sqrt{(P_T+1)T}$ rate, the lower bound *suggests* that this rate may still be attainable with a larger slack of up to $\Theta(\sqrt{T/P_T})$. We establish next a matching upper bound, showing that this threshold indeed marks the true tradeoff between laziness and regret. But first, we formalize how a larger k improves switching cost.

4 ANALYSIS OF k-LAZYGD

Assumptions. Throughout, we make the standard Lipschitzness and compact-domain assumptions: $\mathcal{X} \subset \mathbb{R}^d$ is convex and $\mathcal{X} \subseteq \{\boldsymbol{x} : \|\boldsymbol{x}\| \le R\}$; (sub)gradients satisfy $\|\boldsymbol{g}_t\| \le G$ for all t with $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$.

We formalize the staleness and stability properties mentioned in the introductory example. Fig. 4 depicts a simple case that contrasts a lazy and a greedy update in polyhedral and strongly convex domains. In the former, large normal cones at the vertices cause a stall in the lazy update, while the greedy update still moves. In the latter, cones reduce to rays, yet the same perturbation induces a smaller displacement when the unconstrained iterate is farther from the boundary.

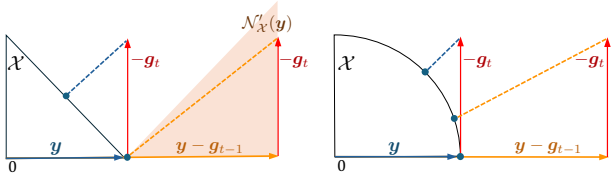

Figure 4: Geometric intuition for the effect of lazy vs. greedy updates in ℓ_1 (left) and ℓ_2 (right) ball domains. From y, greedy projects $y - g_t$, while lazy projects $y - g_t - g_{t-1}$ ($k = 2$, $\sigma = 1$). Blue dots are projections.

4.1 Iterate Staleness

Denote by $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$ a closed convex cone associated with $\boldsymbol{x}_t$, and define its translation $\mathcal{N}'_{\mathcal{X}}(\boldsymbol{x}_t)$ as:

$$\mathcal{N}'_{\mathcal{X}}(\boldsymbol{x}_t) \doteq \boldsymbol{x}_t + \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t) \ =\ \{\boldsymbol{x}_t + \boldsymbol{v} : \boldsymbol{v} \in \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)\}.$$

Proposition 3 (Staleness of Lazy Iterates)**.** *For any t at which the gradient $\boldsymbol{g}_t$ satisfies the inclusion*

$$\boldsymbol{y}_{t+1} = \boldsymbol{x}_{t-n_t} - \frac{1}{\sigma}\boldsymbol{g}_{t-n_t:t} \ \in\ \mathcal{N}'_{\mathcal{X}}(\boldsymbol{x}_t), \qquad (12)$$

for some $0 < n_t \le k-1$, we have that

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| = 0.$$

Proof. From the first order optimality conditions of the Euclidean projection, and the uniqueness of the projection, we have that for any $\boldsymbol{z}$:

$$\boldsymbol{x}_t = \Pi(\boldsymbol{z}) \iff \boldsymbol{z} \in \mathcal{N}'_{\mathcal{X}}(\boldsymbol{x}_t). \qquad (13)$$

Comparing (12) and (13), we conclude that

$$\boldsymbol{x}_t = \Pi\big(\boldsymbol{x}_{t-n_t} - \frac{1}{\sigma}\boldsymbol{g}_{t-n_t:t}\big) = \boldsymbol{x}_{t+1}.$$

The last equality by k-LazyGD's update rule. □

Remarks. The condition in (12) becomes more likely as the laziness parameter k increases. In the lazy case ($k > 1$), the newest gradient $\boldsymbol{g}_t$ is added to an *existing aggregation* of up to $k-1$ previous gradients. Its individual influence is therefore dampened; the single most recent gradient is less likely to change the aggregate direction enough to *leave* the cone $\mathcal{N}'_{\mathcal{X}}(\boldsymbol{x}_t)$. As a result, the iterate is more likely to become "stale" ($\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$), as stated in the proposition.

In contrast, for the greedy case ($k = 1$), the update depends entirely on the single, most recent gradient. For the iterate to remain unchanged, we would need $\boldsymbol{x}_t - \frac{1}{\sigma}\boldsymbol{g}_t \in \mathcal{N}'_{\mathcal{X}}(\boldsymbol{x}_t)$, which can be a much stricter condition than (12) with a general k. This illustrates the "laziness advantage": a state that includes a primal point *plus* accumulated gradients is less sensitive to the variance of a single new gradient than a state consisting of *only* a bare primal point.

4.2 Iterate Stability

Proposition 4 (Stability of Lazy Iterates)**.** *For any t at which the gradient $\boldsymbol{g}_t$ satisfies the inequality:*

$$\|\boldsymbol{y}_{t+1}\| = \|\boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t:t}}{\sigma}\| \geq R + \frac{\alpha G\, n_t + \|\boldsymbol{g}_t\|}{\sigma}, \quad (14)$$

for some $0 < n_t \leq k-1, \alpha \in [0,1]$, and assuming an ℓ_2-ball domain, we have that

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \leq \frac{R\,G}{R\sigma + \alpha\, G\, n_t}. \quad (15)$$

That is, when the unconstrained iterate is outside $\mathcal{X}$,[10] and gradients have some correlation parametrized by α, the movement is further attenuated.

Proof. Note that for $\|\boldsymbol{y}\| > R$, projection onto the Euclidean ball is $\Pi(\boldsymbol{y}) = R\,\boldsymbol{y}/\|\boldsymbol{y}\|$. This map is differentiable on $\{\boldsymbol{y} : \|\boldsymbol{y}\| > R\}$, and its Jacobian has operator norm $\|D\Pi(\boldsymbol{y})\|_{\text{op}} = R/\|\boldsymbol{y}\|$ (shown in Appendix I.2). By the mean value inequality for vector mappings,

$$\begin{aligned}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| &= \|\Pi(\boldsymbol{y}_{t+1}) - \Pi(\boldsymbol{y}_t)\| \\ &\leq \sup_{\boldsymbol{z}\in[\boldsymbol{y}_t\ \boldsymbol{y}_{t+1}]} \|D\Pi(\boldsymbol{z})\|_{\text{op}}\, \|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|. \quad (16)\end{aligned}$$

Next, noting that $t + 1 - n_{t+1} = t - n_t$, we have

$$\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\| = \frac{\|\boldsymbol{g}_t\|}{\sigma} \leq \frac{G}{\sigma}. \quad (17)$$

Moreover, for any $s \in [0,1]$,

$$\begin{aligned}\|\boldsymbol{y}_t + s(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t)\| &\geq \|\boldsymbol{y}_{t+1}\| - \|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\| \\ &\geq \|\boldsymbol{y}_{t+1}\| - \frac{\|\boldsymbol{g}_t\|}{\sigma} \geq R + \frac{\alpha G n_t}{\sigma}, \quad (18)\end{aligned}$$

where the first inequality is the reverse triangle inequality and the last uses (14). Hence the entire segment $[\boldsymbol{y}_t, \boldsymbol{y}_{t+1}]$ lies outside the ball, and therefore

$$\sup_{\boldsymbol{z}\in[\boldsymbol{y}_t\ \boldsymbol{y}_{t+1}]} \|D\Pi(\boldsymbol{z})\|_{\text{op}} = \frac{R}{\min_{\boldsymbol{z}\in[\boldsymbol{y}_t\ \boldsymbol{y}_{t+1}]} \|\boldsymbol{z}\|}.$$

Substituting into (16) gives

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \leq \frac{R\,\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|}{\min_{\boldsymbol{z}\in[\boldsymbol{y}_t\ \boldsymbol{y}_{t+1}]} \|\boldsymbol{z}\|}. \quad (19)$$

Using (17) and the lower bound on $\min_{\boldsymbol{z}\in[\boldsymbol{y}_t\ \boldsymbol{y}_{t+1}]} \|\boldsymbol{z}\|$ implied by (18) yields the stated bound. □

Remark. For $k > 1$, the bound in (15) makes explicit the extra attenuation gained within each lazy phase through the $1/n_t$ factor, which reduces the switch size. For example, greedy OGD (k=1) maintains switches of order G/σ, whereas in the fully lazy case (k=T) they shrink to order $G/(\sigma + t)$, which is much stronger. We note that our use of the ℓ_2-ball domain is for notational simplicity. The strict contraction property in (19) holds when projecting for any strongly convex domain, albeit with different constants.

Props. 3 and 4 formalize the structure exploited by laziness: (12) shows that if the unconstrained iterate $\boldsymbol{y}_{t+1}$ remains inside the cone, the update halts; (14) shows that when $\|\boldsymbol{y}_{t+1}\|$ is large, movement is attenuated. Both effects strengthen as k increases. Appendix I.1 further ties switching cost to the deviation of $\boldsymbol{g}_t$ from its running mean, without domain or sequence assumptions. Next, we show that these stability gains are compatible with order-optimal dynamic regret when k is chosen according to the lower bound.

4.3 Regret Analysis

Theorem 5. *For any sequence of convex losses and comparators, the dynamic regret with switching cost of k-LAZYGD satisfies*

$$\begin{aligned}\mathcal{R}_T \leq \sum_{t=1}^{T} \min\big(G\delta_t,\, \frac{G^2}{2\sigma}\big) + (2R\sigma + kG)P_T \\ + \frac{\sigma R^2}{2} + \sum_{t=1}^{T} \min(\delta_t, \frac{G}{\sigma}),\end{aligned}$$

Moreover, setting[11]

$$\begin{aligned}\sigma = \sigma\ &\doteq \sqrt{(G^2+2G)T/(4RP_T+R^2)}, \\ k &= \lfloor c\sqrt{2RT/P_T}\rfloor,\end{aligned}$$

for some $c \geq 1$, gives $\mathcal{R}_T = \mathcal{O}\big(\sqrt{(P_T+1)T}\big)$.

Remarks. The general bound in terms of (k, σ) is a result of key lemmas that we present hereafter. Ideally, we would optimize (k, σ) to minimize this upper bound. However, this is impractical here: the dependence of δ_t on k is domain-specific and lacks a closed

[10] Otherwise, the projection is not active, and the lazy and greedy states $\boldsymbol{y}_{t+1}$ coincide. The prop. focuses on the interesting case where they differ and affect switching.

[11] This choice of k assumes P_T>0. Otherwise, we may take the largest admissible phase length, $k = T$ (i.e., recover LAZYGD).

form. Instead, we exploit two key facts established earlier: (*i*) the switching cost is decreasing in k (from the previous propositions), and (*ii*) the lower bound restricts k to at most $\Theta(\sqrt{T/P_T})$ to retain optimality. We therefore fix k at this threshold, i.e., use the largest admissible laziness slack. This allows us to focus only on the $1/\sigma$ branch of the min. Then, we obtain a bound that is convex in σ, optimizing over σ then yields the claimed rate.

The starting point of the proof is the *dynamic* FTRL lemma of Mhaisen and Iosifidis (2025, Lem. 4.1), adapted to include switching cost.

Lemma 6. *Let $\{f_t(\cdot), \boldsymbol{u}_t\}_{t=1}^T$ be an arbitrary set of functions and comparators in $\mathcal{X}$, respectively. Consider the extended value functions $\bar{f}_t(\cdot) = f_t(\cdot) + I_{\mathcal{X}}(\cdot)$ and let $r(\cdot)$ be a strongly convex regularizer such that*

$$\boldsymbol{x}_{t+1} \doteq \operatorname*{argmin}_{\boldsymbol{x}} h_{0:t}(\boldsymbol{x})$$

is well-defined, where $h_0(\boldsymbol{x}) \doteq I_{\mathcal{X}}(\boldsymbol{x}) + r(\boldsymbol{x})$, $\forall t \geq 1$:

$$h_t(\boldsymbol{x}) \doteq \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle \quad \boldsymbol{p}_t \in \partial \bar{f}_t(\boldsymbol{x}_t).$$

Then, the algorithm that selects the actions $\boldsymbol{x}_{t+1}, \forall t$ achieves the following dynamic regret bound:

$$\mathcal{R}_T \leq \sum_{t=1}^{T} \big(\overbrace{h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1})}^{(\mathbf{I})} \big) + \sum_{t=1}^{T-1} \overbrace{\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|}^{(\mathbf{II})} + \sum_{t=1}^{T-1} \big(\overbrace{h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t)}^{(\mathbf{III})} \big) + r(\boldsymbol{u}_1).$$

Note that the lemma's update matches the FTRL form of k-LAZYGD, in (3), with $r(\cdot) = 1/2 \|\cdot\|^2$.

This dynamic regret decomposition of FTRL yields three terms: (**I**) the penalty for not knowing $f_t(\cdot)$ at decision time, (**II**) the switching cost, and (**III**) the penalty from comparator shift, which is mainly driven by the state's size $\|\boldsymbol{p}_{1:t}\|$. $r(\boldsymbol{u}_1)$ is the minimum regularization penalty (i.e., when $P_T = 0$). These terms can be individually bounded in a way that leads to the result in Thm. 5, as the next lemma states.

Lemma 7. *Under the Lipschitzness of the loss functions and compactness of the domain, k-LAZYGD iterates guarantee the following bounds on each part of the dynamic FTRL lemma for any t:*

$$(\mathbf{I}) \leq \min\big(G\delta_t, \tfrac{G^2}{2\sigma}\big),$$

$$(\mathbf{II}) \leq \tfrac{G}{\sigma},$$

$$(\mathbf{III}) \leq (2R\sigma + kG) \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|.$$

The proof of each part is deferred to Appendix G (Lemma 7.a–c). The arguments rely on standard tools from FTRL analysis, which we reuse thanks to Thm. 1. However, an extra difficulty arises from the non-standard $\boldsymbol{g}_t^I$ term, whose behavior depends on the slack k, requiring a refinement of the classical analysis.

5 LEARNING THE OPTIMAL SLACK k

Thm. 5 assumes optimal tuning of the rate σ and slack k, requiring knowledge of P_T. While we can use an upper bound on it, achieving adaptivity to all comparators requires the unknown P_T. We adopt the ensemble framework, which discretizes the P_T space and uses a meta-learner to find the near-optimal choice. For simplicity, we impose the convention $k = \max(1, \lfloor \frac{2R}{G}\sigma \rfloor)$. Since the optimal values of both parameters is $\Theta(\sqrt{T/P_T})$, this restriction affects only constant factors. The advantage is that the search can be carried out solely over σ, while implicitly tuning k.

From the bound $P_T \leq 2R(T-1)$, the optimal regularization σ is guaranteed to lie in the interval

$$\sqrt{\tfrac{(G^2+2G)T}{R^2}} \;\geq\; \sigma \;\geq\; \sqrt{\tfrac{(G^2+2G)T}{8R^2T - 8R^2}}.$$

Let $B \doteq \sqrt{\frac{(G^2+2G)T}{R^2}}$, and construct the grid:

$$\mathcal{H} \doteq \Big\{ \tfrac{B}{2^{i-1}} : i = 1, \ldots, N \Big\},\; N \doteq \Big\lceil \tfrac{1}{2} \log(8T-7) \Big\rceil + 1.$$

Consider the exponent $s \doteq \log\big(\sqrt{1 + 4P_T/R}\big) + 1$ and its floor $s \doteq \lfloor \log\big(\sqrt{1 + 4P_T/R}\big) \rfloor + 1$. Then,

$$\frac{B}{2^{s-1}} \geq \frac{B}{2^{s^\star - 1}} \geq \frac{B}{2^s}, \Rightarrow \sigma^s \doteq \tfrac{B}{2^{s-1}} \geq \sigma \;\geq\; \tfrac{B}{2^s}.$$

By construction, $\sigma^s \in \mathcal{H}$, and hence the candidate set always contains a near-optimal choice.

Theorem 8. *For any comparator sequence, running the meta-learner of (Zhang et al., 2021, "SAder") over a set of k-LAZYGD experts from $\mathcal{H}$, guarantees*

$$\mathcal{R}_T = \mathcal{O}\big(\sqrt{(P_T + 1)T}\big).$$

Remarks. While this bound is known for ensembles of greedy OGD learners (Zhang et al., 2021), our construction differs in a key respect: the experts vary not only in their learning rate but also in the laziness slack, i.e., how many past gradients are aggregated *and* how they are weighted in the k-LAZYGD update. This design allows our ensemble to exploit the structural advantages captured by Propositions 3 and 4. Moreover, if one were to optimize k and σ independently, the

search would expand to their Cartesian product; by tying them together, we avoid this extra complexity while still achieving order-optimal guarantees. We detail the proof of this result in Appendix H.

6 RELATED WORK

SOCO admits two information models: a *look-ahead* setting where the cost is known before acting, usually studied through *competitive ratio* (CR) metric, and a *fully online* setting where the cost is revealed only after the decision, usually studied through regret $\mathcal{R}_T$.

Look-ahead An important result in the look-ahead category is Online Balanced Descent (OBD) (Chen et al., 2018a) and its follow-ups. OBD projects the previous iterate onto a carefully chosen level set of the *known* hitting cost to balance movement and hitting costs. This attains dimension-free CR, and its variants attain an optimal CR for specific families of costs (Goel et al., 2019). OBD also admits dynamic-regret guarantees with switching costs, but these hold only against a pre-fixed upper bound on P_T. The guarantees of OBD family were further generalized to models with look-ahead *window* (Lin et al., 2020) with possibly imperfect predictions (Rutten et al., 2023). Because these methods have access to the current hitting cost, they differ from our lazy update, which targets regret guarantees in the online setting.

Fully online. In this setting, Li and Li (2020) studied SOCO with prediction windows under smoothness and strong convexity assumptions. In contrast, Zhang et al. (2021) achieved optimal dynamic regret with switching-cost guarantees for general convex functions,[12] which, along with its generalization to dynamic regret over any sub-interval (Zhang et al., 2022), represents the current state of the art in both CR and dynamic regret. Their analysis relies on an ensemble of GD base learners with different σ but fixed k (k=1 for all learners), reinforcing the common view that gradient accumulation (dual-averaging) cannot yield dynamic-regret guarantees. Our work shows that a *rationed* form of accumulation does achieve such guarantees, and can indeed be instantiated within the same ensemble framework. Overall, dual-averaging and related lazy methods remain unexplored in SOCO, whether under look-ahead or fully online models, and under either CR or dynamic-regret criteria.

FTRL/OMD interplay. GD is typically identified with Online Mirror Descent (OMD) using the squared ℓ_2 norm as the distance-generating function (see, e.g., Orabona (2022, Sec . 6.2)). In contrast, LazyGD corresponds to FTRL with the same function as a regularizer (see, e.g., McMahan (2017, Sec. 3.2)). More recently, Jacobsen and Cutkosky (2022) introduced "centered OMD", a method that can reproduce both the lazy and greedy forms of OGD. However, their work does not investigate nor optimize the interpolation between these two extremes. Mhaisen and Iosifidis (2025) introduced the pruning perspective as a unifying lens for greedy and lazy updates, showing its role in recovering dynamic-regret guarantees. However, they do not incorporate laziness slack, which is the quantity we identified, and optimized, as the key driver of low switching cost in SOCO.

Other "lazy" forms. The term "lazy" has also appeared in the literature with a different meaning. In the related work of Sherman and Koren (2021) and Agarwal et al. (2023), a method is called lazy if the expected *number* (not magnitude) of switches is bounded. Their construction relies on FTRL, augmented with a *lazy sampling* scheme that couples consecutive decisions maximally, thereby ensuring such switching guarantees. However, these works restrict attention to the weaker static regret metric. Moreover, since our method naturally operates in lazy phases, it is in fact possible in principle to combine k-LazyGD with their lazy-sampling scheme, yielding the same bounded switching number within each phase, while also having dynamic regret guarantees. Finally, we discuss in Appendix F the related framework and algorithms of "OCO with memory"(Anava et al., 2015).

7 CONCLUSION

We introduced k-LazyGD, a *partially*-lazy learner via an FTRL–pruning view. It achieves minimax dynamic regret while partially retaining the switching behavior of lazy methods, with a matching lower bound. Promising extensions include non-Euclidean and adaptive-rate schemes, incorporating predictions, and competitive ratio or adaptive regret analysis.

Acknowledgment

This work was supported by the Dutch National Growth Fund project "Future Network Services", and by the European Commission through Grants No. 101139270 "ORIGAMI" and No. 101192462 "FLECON-6G".

References

Naman Agarwal, Satyen Kale, Karan Singh, and Abhradeep Thakurta. Differentially private and lazy online convex optimization. In *Proc. of COLT*, 2023.

[12] They also develop models for the look-ahead setting.

Oren Anava, Elad Hazan, and Shie Mannor. Online learning for adversaries with memory: price of past mistakes. In *Proc. of NeurIPS*, 2015.

Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proc. of ICML*, 2012.

Amir Beck. *First-Order Methods in Optimization*. SIAM-Society for Industrial and Applied Mathematics, 2017.

Nicolo Cesa-Bianchi, Ofer Dekel, and Ohad Shamir. Online learning with switching costs and other adaptive adversaries. In *Proc. of NeurIPS*, 2013.

Lin Chen, Qian Yu, Hannah Lawrence, and Amin Karbasi. Minimax regret of switching-constrained online convex optimization: No phase transition. In *Proc. of NeurIPS*, 2020.

Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Proc. of COLT*, 2018a.

Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Proc. of COLT*, 2018b.

Genevieve E Flaspohler, Francesco Orabona, Judah Cohen, Soukayna Mouatadid, Miruna Oprescu, Paulo Orenstein, and Lester Mackey. Online learning with optimism and delay. In *Proc. of ICML*, 2021.

Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Proc. of NeurIPS*, 2019.

Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. *Machine learning*, 80(2), 2010.

Andrew Jacobsen and Ashok Cutkosky. Parameter-free mirror descent. In *Proc. of COLT*, 2022.

Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms. In *Proc. of AAAI*, 2016.

Yingying Li and Na Li. Leveraging predictions in smoothed online convex optimization via gradient-based algorithms. *Proc. of NeurIPS*, 2020.

Yiheng Lin, Gautam Goel, and Adam Wierman. Online optimization with predictions and non-convex losses. *Proc. ACM Meas. Anal. Comput. Syst.*, 4 (1):1–32, 2020.

Tuo Liu, El Mehdi Saad, Wojciech Kotłowski, and Francesco Orabona. Dual averaging converges for nonconvex smooth stochastic optimization. *arXiv:2505.21394*, 2025.

H. Brendan McMahan. A Survey of Algorithms and Analysis for Adaptive Online Learning. *J. Mach. Learn. Res.*, 18(1):3117–3166, 2017.

Neri Merhav, Erik Ordentlich, Gadiel Seroussi, and Marcelo J Weinberger. On sequential strategies for loss functions with memory. *IEEE Transactions on Information Theory*, 48(7):1947–1958, 2002.

Naram Mhaisen and George Iosifidis. On the dynamic regret of following the regularized leader: Optimism with history pruning. In *Proc. of ICML*, 2025.

Francesco Orabona. A Modern Introduction to Online Learning. *arXiv.1912.13213*, 2022.

Daan Rutten, Nicolas Christianson, Debankur Mukherjee, and Adam Wierman. Smoothed online optimization with unreliable predictions. *Proc. ACM Meas. Anal. Comput. Syst.*, 7(1):1–36, 2023.

Sarah Sachs, Hédi Hadiji, Tim van Erven, and Cristóbal Guzmán. Between stochastic and adversarial online convex optimization: Improved regret bounds via smoothness. In *Proc. of NeurIPS*, 2022.

Shai Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University of Jerusalem, 2007.

Uri Sherman and Tomer Koren. Lazy oco: Online convex optimization on a switching budget. In *Proc. of COLT*, 2021.

Lin Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *Proc. of NeurIPS*, 2009.

Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive online learning in dynamic environments. In *Proc. of NeurIPS*, 2018.

Lijun Zhang, Wei Jiang, Shiyin Lu, and Tianbao Yang. Revisiting smoothed online learning. In *Proc. of NeurIPS*, 2021.

Lijun Zhang, Wei Jiang, Jinfeng Yi, and Tianbao Yang. Smoothed online convex optimization based on discounted-normal-predictor. In *Proc. of NeurIPS*, 2022.

Peng Zhao, Yu-Jie Zhang, Lijun Zhang, and Zhi-Hua Zhou. Adaptivity and non-stationarity: Problem-dependent dynamic regret for online convex optimization. *JMLR*, 25(98):1–52, 2024.

Yawei Zhao, Qian Zhao, Xingxing Zhang, En Zhu, Xinwang Liu, and Jianping Yin. Understand dynamic regret with switching cost for online decision making. *ACM Trans. on Intelligent Systems and Technology*, 11(3), 2020.

Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proc. of ICML*, 2003.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes.
   (b) Complete proofs of all theoretical results. Yes.
   (c) Clear explanations of any assumptions. Yes.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes.
   (d) A description of the computing infrastructure used. No.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Not Applicable
   (b) The license information of the assets, if applicable. Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

Partially Lazy Gradient Descent for Smoothed Online Learning Supplementary Materials

A SUMMARY OF k-LAZYGD FORMS

We summarize the two formulations of the k-LAZYGD algorithm, detailed in Algorithms 1 and 2. Theorem. 1 establishes that the projection and FTRL forms produce identical iterates. Hence, for notational simplicity, we omit the distinction between $\boldsymbol{x}_t$ and $\boldsymbol{x}_t$ below.

Projection form. The k-lazy update can be expressed directly in terms of projected gradient steps:

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \Pi\left(\boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t:t}}{\sigma}\right)\\ n_t &= (t-1) \mod (k)\end{aligned}$$

where Π denotes Euclidean projection onto $\mathcal{X}$.

Alg. 1 k-LAZYGD (projection form)

Input: compact set $\mathcal{X}$, regularization $\sigma > 0$, laziness slack $k \in \mathbb{N}$, horizon T.
Output: $\{\boldsymbol{x}_t\}_{t=1}^T$.

Initialize $\boldsymbol{x}_1 = 0$
for $t = 1, 2, \ldots, T$ **do**
  Use action $\boldsymbol{x}_t$.
  $f_t(\cdot)$ *is revealed*
  Compute a subgradient $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$.
  Set the counter $n_t \doteq (t-1) \bmod (k)$. (within-phase index)
  Accumulate the within-phase gradients $\boldsymbol{g}_{t-n_t:t} \doteq \sum_{\tau=t-n_t}^{t} \boldsymbol{g}_\tau$.
  Form the unconstrained step $\boldsymbol{y}_{t+1} \doteq \boldsymbol{x}_{t-n_t} - \frac{1}{\sigma}\boldsymbol{g}_{t-n_t:t}$.
  Project back to the domain $\boldsymbol{x}_{t+1} = \Pi(\boldsymbol{y}_{t+1})$.
end for

FTRL form. Equivalently, k-LAZYGD admits the following Follow-the-Regularized-Leader formulation:

$$\boldsymbol{x}_{t+1} = \Pi(\boldsymbol{y}_{t+1}), \quad \boldsymbol{y}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{x}\|^2,$$

or, equivalently, as a constrained minimization:

$$\boldsymbol{x}_{t+1} = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{x}\|^2. \tag{20}$$

State. The state vector $\boldsymbol{p}_t$ accumulates both the loss subgradients and a possible correction term arising from the normal cone:

$$\begin{aligned}&\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I,\\ &\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t), \quad \boldsymbol{g}_t^I \in \partial I_{\mathcal{X}}(\boldsymbol{x}_t) = \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t).\end{aligned}$$

The correction term implements pruning at the prescribed rate:

$$\boldsymbol{g}_t^I = \begin{cases} -\boldsymbol{p}_{1:t-1} - \sigma\boldsymbol{x}_t, & \text{if } (\boldsymbol{y}_t \notin \mathcal{X} \ \wedge \ n_t = 0),\\ 0, & \text{otherwise.}\end{cases} \tag{21}$$

The above provides a complete and equivalent characterization of the k-LAZYGD update rule in both projection and FTRL form.

Alg. 2 k-LAZYGD as FTRL with pruning (equivalent to Alg. 1)

Input: compact set $\mathcal{X}$, regularization $\sigma > 0$, laziness slack $k \in \mathbb{N}$, horizon T.
Output: $\{\boldsymbol{x}_t\}_{t=1}^{T}$.

Initialize $\boldsymbol{x}_1 \in \mathcal{X}$, set $\boldsymbol{p}_{1:0} \doteq \mathbf{0}$.
for $t = 1, 2, \dots, T$ **do**
  Use action $\boldsymbol{x}_t$.
  $f_t(\cdot)$ *is revealed*
  Compute a subgradient $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$.
  Set the counter $n_t \doteq (t-1) \bmod (k)$
  Compute the unconstrained FTRL center $\boldsymbol{y}_t = -\frac{1}{\sigma}\boldsymbol{p}_{1:t-1}$.
  Set the normal-cone correction $\boldsymbol{g}_t^I$ (pruning rule, cf. (21)).
  Set $\boldsymbol{p}_t \doteq \boldsymbol{g}_t + \boldsymbol{g}_t^I$ and update the state $\boldsymbol{p}_{1:t} \doteq \sum_{\tau=1}^{t} \boldsymbol{p}_\tau$.
  Solve the (unconstrained) FTRL subproblem

$$\boldsymbol{y}_{t+1} \doteq \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{x}\|^2 \;=\; -\frac{1}{\sigma}\boldsymbol{p}_{1:t}.$$

  Project to the feasible set (equivalently, solve the constrained form (20))

$$\boldsymbol{x}_{t+1} = \Pi(\boldsymbol{y}_{t+1}) \quad \big(\text{ i.e., } \;\boldsymbol{x}_{t+1} \doteq \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{x}\|^2\big).$$

end for

B SWITCHING COST

For both greedy and lazy variants of OGD, the switching cost follows directly from the non-expansiveness of the Euclidean projection (see, e.g., Beck (2017, Thm. 5.4)). Formally, for any $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ and convex set $\mathcal{X}$, the projection $\Pi(\cdot)$ satisfies

$$\|\Pi(\boldsymbol{u}) - \Pi(\boldsymbol{v})\| \;\le\; \|\boldsymbol{u} - \boldsymbol{v}\|. \tag{22}$$

B.1 GD

Recall the update rule:

$$\boldsymbol{x}_{t+1} = \Pi\left(\boldsymbol{x}_t - \frac{1}{\sigma}\boldsymbol{g}_t\right).$$

Applying (22) gives

$$\begin{aligned}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| &= \left\|\Pi\left(\boldsymbol{x}_t - \frac{1}{\sigma}\boldsymbol{g}_t\right) - \boldsymbol{x}_t\right\| \\ &\overset{(a)}{\le} \left\|\boldsymbol{x}_t - \frac{1}{\sigma}\boldsymbol{g}_t - \boldsymbol{x}_t\right\| = \frac{1}{\sigma}\|\boldsymbol{g}_t\| \;\le\; \frac{G}{\sigma}.\end{aligned}$$

The first inequality is by non-expansiveness of Π, and the second by G-Lipschitzness of the losses.

B.2 LAZYGD

Recall the update rule:

$$\boldsymbol{x}_{t+1} = \Pi\left(-\frac{\boldsymbol{g}_{1:t}}{\sigma}\right).$$

Similarly, for the lazy variant, we obtain

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| = \left\|\Pi\left(-\frac{1}{\sigma}\boldsymbol{g}_{1:t}\right) - \Pi\left(-\frac{1}{\sigma}\boldsymbol{g}_{1:t-1}\right)\right\|$$

$$\overset{(a)}{\leq} \left\| -\frac{1}{\sigma}\boldsymbol{g}_{1:t} + \frac{1}{\sigma}\boldsymbol{g}_{1:t-1} \right\| = \frac{1}{\sigma}\|\boldsymbol{g}_t\| \;\leq\; \frac{G}{\sigma}.$$

Again, the first inequality is due to non-expansiveness of Π, and the last from Lipschitzness.

B.3 Discussion

When the projection is inactive, the inequalities (a) in both GD and LazyGD become equalities. In this case, and assuming $\|\boldsymbol{g}_t\| = G$ for all t, both greedy and lazy variants incur identical per-round movement of G/σ.

The distinction emerges only when the projection is *active*, the two updates behave differently. For the lazy variant, the term

$$\left\|\Pi(-\tfrac{1}{\sigma}\boldsymbol{g}_{1:t}) - \Pi(-\tfrac{1}{\sigma}\boldsymbol{g}_{1:t-1})\right\|$$

is often small, or even zero, because the state is represented indirectly through the aggregate $\boldsymbol{g}_{1:t}$. As a result, many consecutive gradient-sum pairs $(\boldsymbol{g}_{1:t-1}, \boldsymbol{g}_{1:t})$ can map to nearly identical primal points, a phenomenon previously noted by McMahan (2017, Sec. 6). In this work, we formalize this observation through our *laziness propositions*: the projection operator of the two expressions (which differ only in a single gradient) can be shown to be multiple-to-one mapping (Prop. 3), or strict contraction (Prop. 4). Moreover, in k-LazyGD, we partially recover this stabilizing effect by introducing controlled pruning: the switching term now takes the form:

$$\left\|\Pi\big(\boldsymbol{x}_{t-n_t} - \tfrac{1}{\sigma}\boldsymbol{g}_{t-n_t:t-1}\big) - \Pi\big(\boldsymbol{x}_{t-n_t} - \tfrac{1}{\sigma}\boldsymbol{g}_{t-n_t:t}\big)\right\|.$$

This design ensures that while movements remain bounded by the same G/σ worst-case limit, the effective switching cost is often smaller in practice, reflecting the same suppression that characterizes fully lazy methods.

Finally, it is worth noting that when projection operator is inactive $\forall t$, k-LazyGD produces that same iterates for all k, implying the equivalence between (GD) and (LazyGD) (recall the constant learning/regularization rate assumption).

C MORE DETAILS ON THE ILLUSTRATIVE EXAMPLES

We provided two stylized examples in the introduction, and we detail the setup below.

Example (i)**.** We consider a 2-dimensional test with horizon $T = 101$. The cost sequence $g_1, \ldots, g_T \in \mathbb{R}^2$ is axis-aligned and unit-norm: $g_t = (-1, 0)$ for $t = 1, \ldots, 12$, and for $t = 13, \ldots, 101$ the costs cycle with period four through $(1, 0), (0, 1), (-1, 0), (0, -1)$, producing a periodic drift with direction flips. Decisions $\boldsymbol{x}_t$ lie in the ℓ_1-ball $\{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\|_1 \leq 1\}$. The regularization rate is fixed to $\sigma = \sqrt{t}$ for both algorithms. For the actions snapshot, we use a fixed $\sigma = \sqrt{T}$ instead to better capture the four actions.

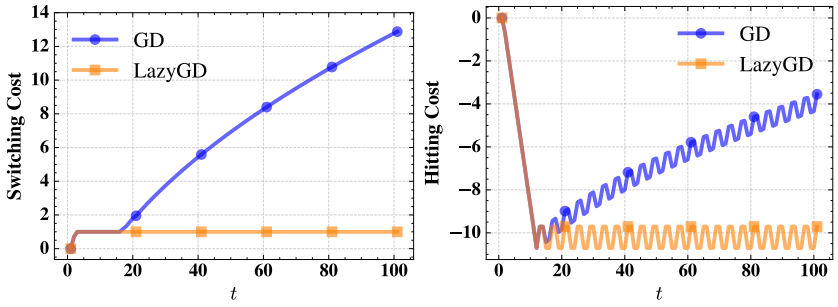

Figure 5: Switching and hitting cost in Example (i).

Example (ii)**.** We consider a 2-dimensional test with horizon $T = 101$. The cost sequence $g_1, \ldots, g_T \in \mathbb{R}^2$ is axis-aligned: $g_t = (-1, 0)$ for $t = 1, \ldots, 11$, and for $t = 12, \ldots, 101$ we set $g_t = (-1, (-1)^t)$, i.e. the second coordinate alternates in sign producing $(-1, 1), (-1, -1), \ldots$. This yields a piecewise-constant initial segment followed by rapid vertical oscillations with abrupt sign flips. Decisions $\boldsymbol{x}_t$ lie in the ℓ_2-ball $\{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\|_2 \leq 1\}$.

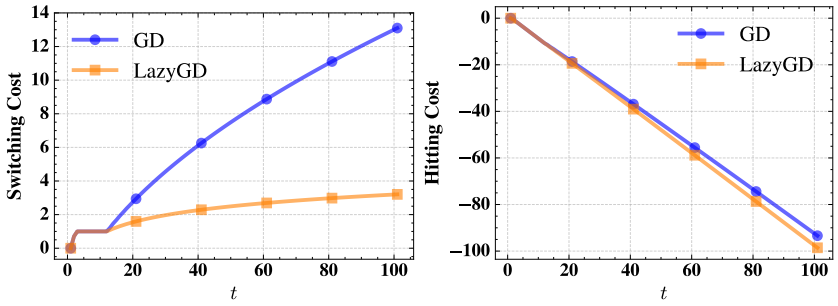

Figure 6: Switching and hitting cost in Example (*ii*).

Again, the regularization rate is fixed to $\sigma = \sqrt{t}$ for both algorithms. For the actions snapshot, we use a fixed $\sigma = \sqrt{T}$ instead to better capture the four actions.

The message of the two examples is: Greedy GD ($k = 1$) is not necessarily good for SOCO results. In this scenario, stepping in the direction of the most recent gradient causes worse movement *and* hitting cost than aggregating all past gradients.

D k-LAZYGD SIMULATIONS

We complement our theory with simulations illustrating the behavior of (k-LAZYGD)[13]. Our goal is to verify that partial laziness can substantially reduce switching cost without degrading hitting performance. In particular, we expect the switching cost to be empirically comparable to (LAZYGD) and the hitting cost close to (GD), leading to overall performance that improves upon the standard SOCO baseline (GD).

For each algorithm, we report three quantities:

$$\textit{switching cost:} \quad \sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|,$$

$$\textit{dynamic regret:} \quad \sum_{t=1}^{T} \big(f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t)\big),$$

$$\textit{total regret:} \sum_{t=1}^{T} \Big(f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t)\Big) + \sum_{t=1}^{T} \Big(\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\| - \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|\Big) = \mathcal{R}_T - \sum_{t=1}^{T-1} \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|. \tag{23}$$

Note that (23) differs from the regret $\mathcal{R}_T$ defined in the main text in that it subtracts the comparator's path length. This simplification was adopted in the theoretical analysis for cleaner presentation, since $\mathcal{R}_T$ still upper bounds (23). For the numerical results, however, we plot the full expression (23).

D.1 Shifting Stochastic Sequences

In this experiment, gradients $\boldsymbol{g}_t$ are normalized to satisfy $\|\boldsymbol{g}_t\|_2 = 1$, so $G \doteq 1$. We consider linear costs

$$f_t(\boldsymbol{x}) \doteq \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle, \qquad \boldsymbol{g}_t \in [-1, 1]^5, \ \|\boldsymbol{g}_t\|_2 = 1,$$

with decisions constrained to the ℓ_2 unit ball

$$\mathcal{X} \doteq \{\boldsymbol{x} \in \mathbb{R}^5 : \|\boldsymbol{x}\|_2 \leq 1\}.$$

Sequence generation. Let $T_{\text{phase}} \doteq 4000$, $P \doteq 15$. We generate P phases with alternating mean directions $\mu_p \in \{+1, -1\}$ (starting positive), and sample within each phase i.i.d. Gaussian gradients with variance 10 per coordinate. Each sampled row is then ℓ_2-normalized (i.e., projected Gaussian). Concatenating the P phases yields

[13]The code for the experiments can be found on the following repository `https://github.com/Naram-m/k-lazy`.

a horizon of length $T_{\text{tot}} \doteq P\,T_{\text{phase}}$. This construction is interesting because it is reflective of the Stochastically Extended Adversary (SEA) model: environment *is* stochastic, perhaps with high variance, but with *adversarial* distribution shifts (see more details on the SEA model in Sachs et al. (2022)).

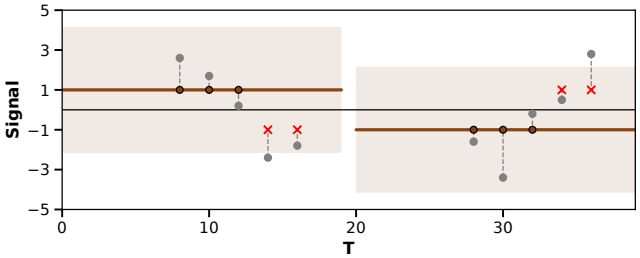

Figure 7: Example cost sequence. 1 coordinate, 2 phases of length 20. Few samples of each phase and their projections on $\{1, -1\}$ are shown. Red 'x' is a point of opposite sign to the phase.

Comparator. The comparator $\{\boldsymbol{u}_t\}$ is piecewise-constant across phases. In phase p, $\boldsymbol{u}_t$ is fixed to a unit ℓ_2 vector pointing opposite to the phase mean (equal-mass over coordinates), and switches at phase boundaries; hence the path length $P_T = 28$ (2 at each switch, with 14 total switches).

Algorithms. We instantiate GD, LazyGD, and k-LazyGD with laziness parameters $k \in \{65, 150, 300, 1500\}$. All algorithms operate in $\mathcal{X}$ on the same gradient sequence $\{\boldsymbol{g}_t\}$ and use an identical regularization schedule, $\sigma = \sqrt{t/60} \approx \sqrt{t/2\tau}$, where $\tau = 30$ is an upper bound on the comparator path length. Since the horizon T is unknown, each algorithm simply substitutes the current time t for T. Other approaches, e.g., the doubling trick, are possible; the important point is that all methods are treated consistently. The sole distinction is that k-LazyGD varies in k; all other settings are kept the same. Note that $\sqrt{T/\tau} \approx 50$ (recall $k = \Theta(\sqrt{T/P_T})$), so the first two k values are representative of this range. In the experiments, we observe that $k \in \{65, 150, 300\}$ all outperform the GD benchmark, achieving improvements of up to 45%.[14]

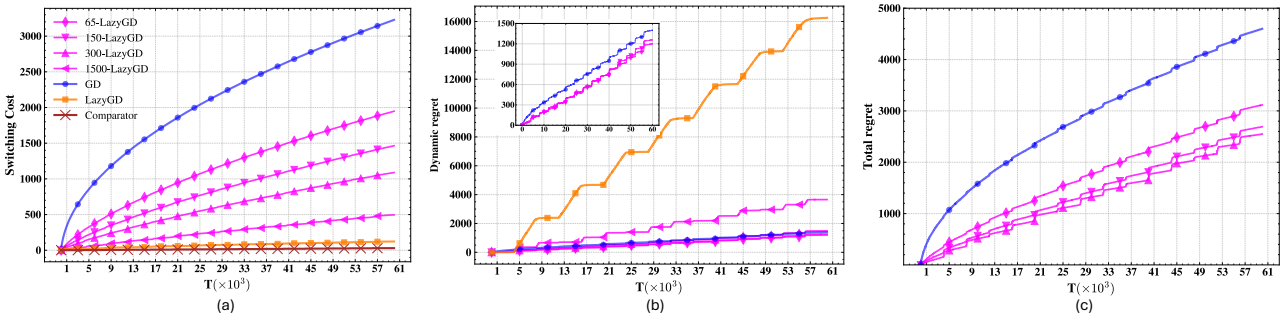

Figure 8: Shifting stochastic sequences. (a) Switching cost, (b) hitting cost, and (c) total regret for GD, LazyGD, and k-LazyGD with different k.

D.2 Corrupted Sequences

We next consider a deterministic counterpart to the stochastic phases above. Gradients $\boldsymbol{g}_t$ are normalized to satisfy $\|\boldsymbol{g}_t\|_2 = 1$, so $G \doteq 1$. We consider linear costs

$$f_t(\boldsymbol{x}) \doteq \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle, \qquad \boldsymbol{g}_t \in [-1, 1]^5,\ \|\boldsymbol{g}_t\|_2 = 1,$$

with decisions constrained to the ℓ_1 unit ball

$$\mathcal{X} \doteq \{\boldsymbol{x} \in \mathbb{R}^5 : \|\boldsymbol{x}\|_1 \le 1\}.$$

[14] If the ensemble framework were employed, it would automatically select the best-performing k for the realized sequence, regardless of the minimax-optimal $k^\star$. While $k^\star$ is a minimax choice, in easier sequences other values of k may perform better, which is precisely the motivation for Sec. 5.

Sequence generation. In each phase, gradients are predominantly aligned with one sign for long stretches (here of length 100), but are interspersed with short bursts of the opposite sign (length 10). The phase length is 2000, and a phase is labeled positive or negative according to its dominant sign, while the bursts act as controlled perturbations. All gradients are ℓ_2-normalized, and concatenating $P = 10$ alternating phases yields a horizon of length $T_{\text{tot}} \doteq P\,T_{\text{phase}}$.

This construction is interesting because it isolates the effect of structured, adversarially placed bursts, without the variability introduced by sampling. It therefore serves as a complementary stress test for how the algorithms respond to disruptive fluctuations.

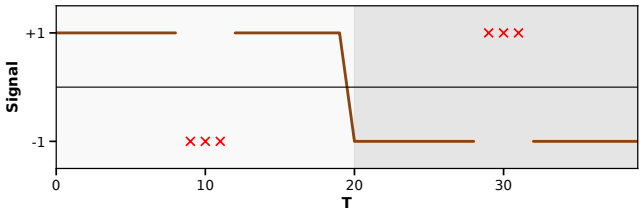

Figure 9: Example cost sequence. 1 coordinate, 2 phases of length 20. Red 'x' is the corruption burst.

Comparator. As before, the comparator $\{\boldsymbol{u}_t\}$ is piecewise-constant across phases, always pointing opposite to the phase mean. Switches occur at phase boundaries, giving path length $P_T = 9\,\frac{2}{\sqrt{5}}$ ($\frac{2}{\sqrt{5}}$ at each of the $P = 9$ switches, recall $\boldsymbol{u}$ is unit ℓ_1 norm here).

Algorithms. We evaluate the same set of algorithms as in the stochastic case: GD, LazyGD, and k-LazyGD with $k \in \{50, 150, 500, 1500\}$. All methods share the same regularization schedule: $\sigma = \sqrt{t/16} \approx \sqrt{t/2\tau}$ (recall $P_T \leq \tau$), and *differ only in the laziness parameter* k. Note that $\sqrt{T/\tau} \approx 50$ (Recall $k = \Theta(\sqrt{T/P_T})$), and hence the first two k values are representative of this range.

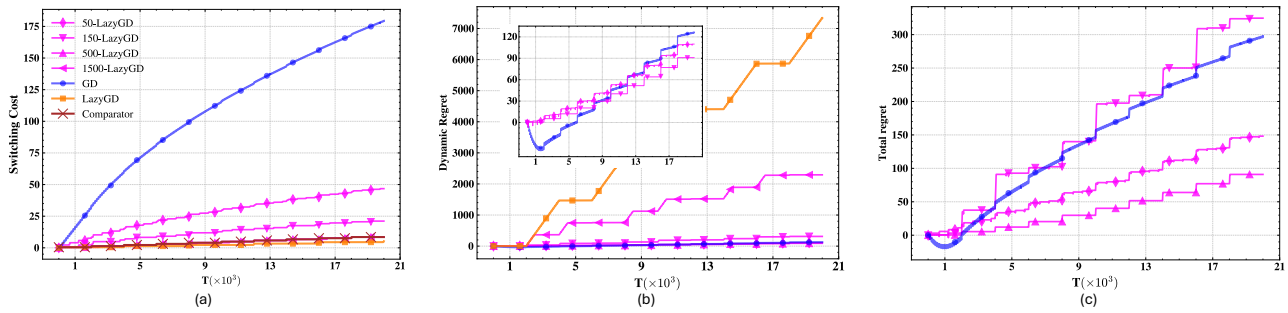

Figure 10: Corrupted phases. (a) Switching cost, (b) dynamic (hitting) regret, and (c) total regret for GD, LazyGD, and k-LazyGD with different k.

D.3 Takeaway from Experiments

The simulations highlight three key observations: (i) k-LazyGD can achieve substantial reductions in switching cost across different choices of k (see parts (a)). (ii) In many cases, these savings come with little deterioration in hitting cost, and in some cases even yield improvements (parts (b)). (iii) The reduction in switching mostly dominates the minor loss in tracking, resulting in lower overall regret (parts (c)). Overall, the results show that controlled laziness offers a favorable trade-off, yielding both stability and responsiveness.

The advantages of k-lazy algorithms are most pronounced in settings with many *undue movements* (movements that add switching cost without improving hitting performance). Both experimental scenarios share this property. In the stochastic case, chasing individual random gradients induces movement but rarely improves tracking, since subsequent samples tend to cancel out and only the mean direction matters. In the corrupted-phase case,

reacting to short bursts similarly increases movement, yet the transient gains are offset by the losses incurred when returning to the baseline.

The worst case for k-LAZYGD. Of course, one can construct sequences that are adversarial for k-lazy methods, where *every* switch is informative and delaying movement is always worse than immediate movements (this is the lower-bound construction). Though we argue this is less practical than the SEA or corrupted sequences above, we provide this case for transparency. The setup is similar to that of corrupted sequences (with 5 phases of length 300 each), except there are *no* outliers; each phase is a *pure* sign (and the comparator is of opposite sign). Here $\sigma = \sqrt{t/2\tau}, \tau = 4, k = 19 = \lfloor\sqrt{T/\tau}\rfloor$. Note that the final regret amount is still in the order of $\sqrt{2R\,(P_T{+}1)\,T}$.

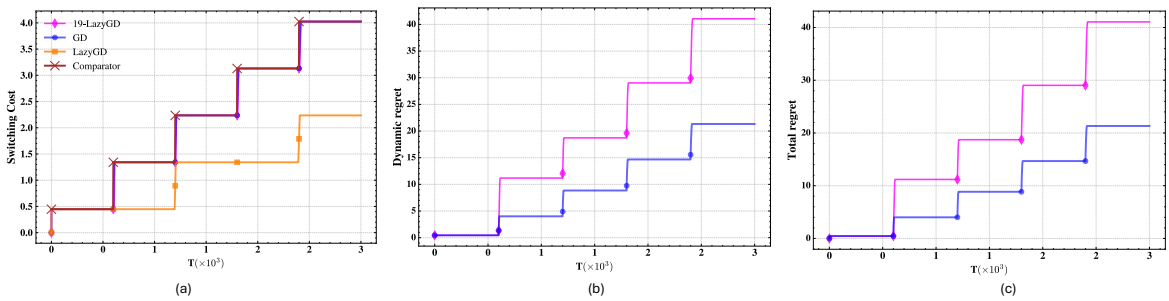

Figure 11: Worst-case for k-LAZYGD methods: every switch is the beginning of a new phase: no outliers due to randomness or deterministically. (a) Switching cost, (b) dynamic (hitting) regret, and (c) total regret for GD, LAZYGD, and k-LAZYGD with different k.

Note that the existence of such sequences does not undermine the findings: the ensemble framework can already include the greedy update as the special case $k = 1$, ensuring robustness across all scenarios. In practice, this means that when undue movements dominate, larger k values yield stability benefits, while in worst-case settings the ensemble naturally falls back to the fully greedy strategy.

D.4 The Ensemble Framework: SADER with OGD vs. k-LAZYGD Base Learners

As discussed earler, the optimal slack parameter k is not known in advance. We therefore learn it online via an ensemble approach. Specifically, we adopt the Smoothed Ader (SAder) meta-learner of Zhang et al. (2021) and use it to aggregate a set of base learners with different slack values.

We compare two ensembles: (i) the original SADER, which aggregates a set of OGD base learners with different learning rates, and (ii) SADER-k, which uses the *same* SADER meta-learner (with the same meta learning rate and the same learning-rate set), but replaces each OGD base learner with a k-lazyOGD base learner.

SAder (OGD experts). The base learners are OGD algorithms with learning rates

$$\eta \in \left\{ \frac{1}{\sqrt{T}}, \frac{2}{\sqrt{T}}, \ldots, \frac{16}{\sqrt{T}} \right\}.$$

SAder-k (k-LAZYGD experts). We use the same learning-rate set, and associate to each η a laziness slack $k = \frac{5\sigma}{2} = \frac{5}{2\eta}$. Thus, each OGD expert is replaced by its corresponding k-LAZYGD variant.

D.4.1 Shifting Stochastic Sequences

We follow the shifting stochastic phases setup from above. Figure 12 reports switching cost, hitting (dynamic) regret, and their sum (total regret). SADER-k improves both components ($\sim 8.5\%$ in hitting costs, and 11.4% in improvement cost).

These gains are consistent with the SEA-like structure of the sequence: within each phase, the variance in gradients induce frequent, often uninformative OGD updates, whereas k-LAZYGD filters such fluctuations (by simply aggregating) and reduces unnecessary switching.

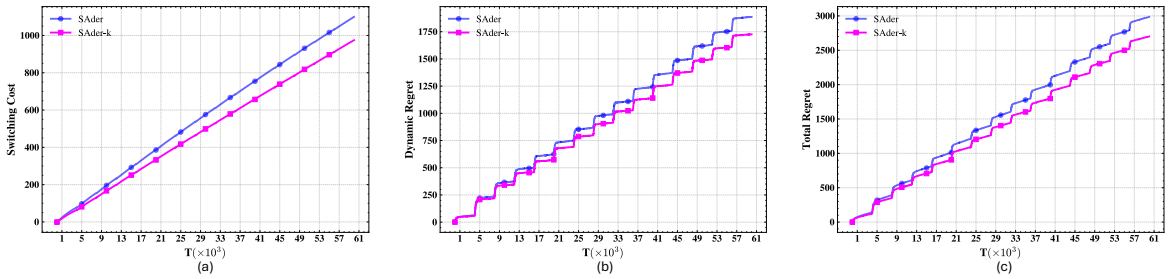

Figure 12: Shifting stochastic sequences with ensemble over greedy (SADER) vs. k-LAZY (SADER-K) learners. (a) Switching cost, (b) dynamic (hitting) regret, and (c) total regret.

D.4.2 Corrupted Sequences

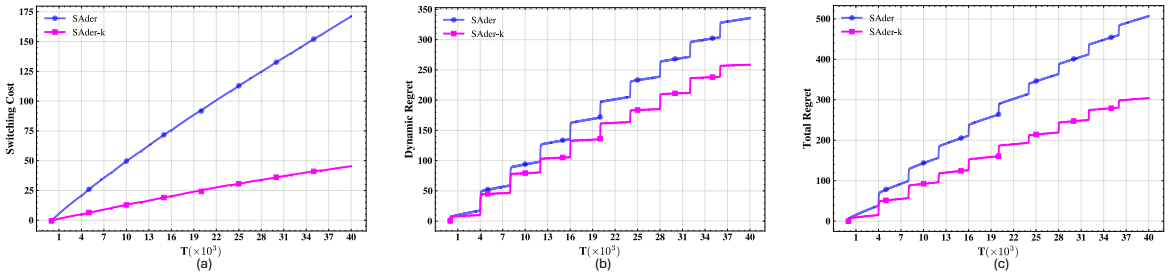

Figure 13: Corrupted sequences with ensemble over greedy (SADER) vs. k-LAZY (SADER-K) learners. (a) Switching cost, (b) dynamic (hitting) regret, and (c) total regret.

We next follow the corrupted phases setup. Figure 13 summarizes the results. Here, SADER-k yields a notable reduction in switching cost alongside improved hitting performance: ($\sim 23\%$ in hitting costs, and 74% in improvement cost)

The improvement is driven by the ability of k-LAZYGD experts to ignore short corruption bursts that would otherwise trigger expensive, transient movements in greedy OGD experts; the meta learner can then concentrate weight on these more stable experts, resulting in lower total regret.

E MORE DETAILS ON THE LOWER BOUND

The idea behind the lower bound is to construct a sequence that makes any k-lazy algorithm behave as if it were fully T-lazy. This is done by forcing the iterate to return to the feasible set every k steps, thereby preventing pruning during the following $k-1$ steps, and so on.

Within each such k-step interval, and since pruning occurs between intervals, the learner accumulates a loss of at least $-\frac{k}{2}$, while the comparator achieves $-k$ (as long as it still has two switches available). Hence, the regret *per interval* is roughly $\frac{k}{2}$, and the total regret is on the order of $\frac{k}{2} \cdot \frac{P_T}{2}$. The formal proof in the main text spells out the index arithmetic needed to make this precise.

E.1 Odd k

When k is odd, we use the modified square shape

$$g_t = \begin{cases} +1, & \text{for } t \in \left[(i-1)k+1, (i-1)k+\frac{k-1}{2}\right] \quad \text{(First half)} \\ -1, & \text{for } t \in \left[(i-1)k+\frac{k-1}{2}+1, ik-1\right] \quad \text{(Second half)} \\ 0, & \text{for } t = ik \quad \text{(Residual)} \end{cases}$$

for $i = 1, \ldots, \lfloor T/k \rfloor$. This construction maintains the two properties that the lower bound used. Mainly, it is still k periodic: That is, $\forall i \in \{1, \ldots, \lfloor \frac{T}{k} \rfloor\}, j \in \{0, \ldots, T-ik\}$:

$$\sum_{t=j+1}^{j+ik} g_t = 0.$$

Moreover, since the sequence starts with the positive half, the sum over any window is non-negative:

$$\sum_{t=1}^{j} g_t \geq 0, \quad \forall j \in [T].$$

Moreover, the losses within an active interval differ slightly for the learner and the comparator. Specifically, the learner incurs $-\frac{k-1}{2}$, while the comparator incurs $-(k-1)$, since the final step contributes zero cost. Hence, the regret over an active interval is

$$-\frac{k-1}{2} - \big(-(k-1)\big) \;=\; \frac{k-1}{2},$$

which yields the same lower bound for all $k \geq 2$ (noting that $k = 1$ corresponds to GD, for which this argument does not apply).

Note that combining our lower bound with the universal one $\Omega(\sqrt{T(P_T+1)})$ from (Zhang et al., 2018, Thm. 2) results in an overall lower bound of:

$$\mathcal{R}_T = \Omega\left(\max\left(\sqrt{T(P_T+1)}, k\,P_T\right)\right) \tag{24}$$

F ADDITIONAL RELATED WORK

F.1 SOCO, Adaptive Adversaries, and Policy Regret.

This subsection connects the SOCO framework to related concepts in the online learning literature, in a way that, to our knowledge, does not appear compiled in one place, especially for the SOCO model. Our aim is to clarify why the cost is naturally decomposed into hitting and movement terms, why the updates optimize only the hitting part f_t, and how this nevertheless yields control over the joint hitting+switching objective. We also show that other convex and Lipschitz movement costs can be incorporated. The discussion below focuses on the fully online (no look-ahead) variant of SOCO.

Absorbing switching into the per-round loss. A natural idea is to absorb the movement penalty into the per-round loss (as it stays convex in $\boldsymbol{x}$) and apply a standard OCO algorithm. Concretely, define

$$j_t(\boldsymbol{x}) \;\doteq\; f_t(\boldsymbol{x}) \;+\; \|\boldsymbol{x} - \boldsymbol{x}_{t-1}\|.$$

If an algorithm guarantees sublinear static regret on $\{j_t\}$, this seems to yield small hitting and switching costs simultaneously. The problem is that the induced benchmark is *misaligned* with the SOCO objective. Indeed,

$$\sum_{t=1}^{T}\big(j_t(\boldsymbol{x}_t) - j_t(\boldsymbol{u}_t)\big) = \sum_{t=1}^{T}\Big(f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t)\Big) \;+\; \sum_{t=1}^{T}\Big(\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\| - \|\boldsymbol{u}_t - \boldsymbol{x}_{t-1}\|\Big), \tag{25}$$

which compares our switching to the *comparator's switching measured against our own history* $\boldsymbol{x}_{t-1}$, rather than its history $\boldsymbol{u}_{t-1}$. As first noted in Arora et al. (2012), this corresponds to a *reactive/non-oblivious* setup: the loss mapping $j_t : \mathcal{X} - \mathbb{R}$ is *parameterized by the learner's realized decision* $\boldsymbol{x}_{t-1}$.

In SOCO, the resulting mismatch term is *negative* and may be dropped. That is, this misaligned quantity in (25) can still be bounded by $\mathcal{R}_T$ similar to what we did in (23). For a *single fixed* k, such looseness can still suffice to analyze SOCO. However, this approach breaks down in *ensemble* (expert/meta) designs: to tune σ, k online, we require the algebraic cancellation between meta-regret and base-expert regret (see the common terms in Theorem. 8, especially the cancellation between (39) and (38)). That cancellation fails if per-round losses reference the learner's own trajectory (via $\boldsymbol{x}_{t-1}$) rather than an external benchmark.

Hence, embedding switching inside $j_t(\boldsymbol{x})$ is brittle; even when it does not alter a fixed (σ, k) analysis, it breaks meta-learning over (σ, k). This motivates the standard SOCO practice of analyzing hitting regret against $\{f_t\}$ and *separately* controlling $\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|$. A parallel route to the same two-part analysis is via policy regret.

Policy regret fixes the benchmark. The remedy is to explicitly model dependence on the previous action and study *policy regret* for an $m=1$ memory loss. Define the bivariate loss

$$j_t(\boldsymbol{y}, \boldsymbol{x}) \doteq f_t(\boldsymbol{x}) + \|\boldsymbol{x} - \boldsymbol{y}\|, \qquad (\boldsymbol{y}, \boldsymbol{x}) \in \mathcal{X} \times \mathcal{X},$$

and evaluate

$$\sum_{t=1}^{T} \underbrace{j_t(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)}_{\text{our policy at } t} - \underbrace{j_t(\boldsymbol{u}_{t-1}, \boldsymbol{u}_t)}_{\text{comparator policy at } t} = \sum_{t=1}^{T} \Big(f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t) \Big) + \sum_{t=1}^{T} \Big(\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\| - \|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\| \Big),$$

which exactly recovers the SOCO dynamic-regret-with-switching objective. This places us in the *oblivious-with-memory* model: the loss mapping $j_t : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is *independent of the learner's realized decision* (hence non-reactive), and dependence on $(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$ arises only through *evaluation*, not through the *definition* of j_t after observing the learner's past. Such memoryful but oblivious adversaries were first handled in Anava et al. (2015).

Unary relaxation and why we optimize only f_t. As observed by Anava et al. (2015), memory losses admit a Lipschitz relaxation. For any loss l_t that is L-Lipschitz in its first argument,

$$l_t(\boldsymbol{y}, \boldsymbol{x}) \leq \underbrace{l_t(\boldsymbol{x}, \boldsymbol{x})}_{\doteq\, l_t(\boldsymbol{x}) \text{ unary form}} + L\,\|\boldsymbol{y} - \boldsymbol{x}\|. \tag{26}$$

This yields the following recipe: (i) run any OCO method on the unary losses $j_t(\boldsymbol{x})$, and (ii) ensure successive decisions are stable (small $\|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\|$). Together, these two properties imply a policy-regret guarantee, hence a SOCO guarantee.

For SOCO, $j_t(\boldsymbol{y}, \boldsymbol{x})$ is 1-Lipschitz in $\boldsymbol{y}$,[15] and its unary form is

$$j_t(\boldsymbol{x}, \boldsymbol{x}) = f_t(\boldsymbol{x}) + \|\boldsymbol{x} - \boldsymbol{x}\| = f_t(\boldsymbol{x}).$$

Therefore, it suffices to run an OCO algorithm on f_t with appropriate (static/dynamic) regret guarantees, provided the algorithm also ensures small movement.

This perspective clarifies why SOCO algorithms, classical GD, dual-averaging / lazy-GD, and our partially-lazy variants, use only gradients of f_t: the hitting part is handled by OCO guarantees, while the switching part is handled by stability (via non-expansiveness and regularization). Our k-lazy design strengthens this effect, as quantified in Propositions 3 and 4 in the main text, and 11 presented later in the appendix.

On other switching-cost forms. From (26), we can see that the fixed ℓ_2 norm can be replaced by any other norm (not necessarily Mahalanobis norms $\|\cdot\|_{A_t}$), provided the movement cost is Lipschitz with respect to that norm and the constant L is adjusted accordingly[16]. Moreover, by properties of Euclidean projection, the switching cost remains bounded in the ℓ_2 norm, which in turn controls all other norms by equivalence of norms.

F.2 Relation to other Phased Updates

The phased structure of the update rule in k-LAZYGD is reminiscent of blocking methods, which also operate in phases (Merhav et al., 2002; Chen et al., 2020). These methods adopt the "blocking argument": the time horizon T is partitioned into S (or k) equally sized blocks, and the cumulative loss of each block is treated as a single loss function for a new OCO problem with only S rounds. While conceptually analogous, the implementation is critically different. The blocking strategy implies a piecewise-constant approach where *a single decision* is made and held constant for the duration of each block, with an update occurring only at the block boundaries. In contrast, the update in k-LAZYGD generates a new decision vector at every time step. This decision follows the regular FTRL update, but adjusted to be initialized differently at the beginning of each phase.

A second point of contrast is with the literature on OCO with delayed feedback (Joulani et al., 2016; Flaspohler et al., 2021), which also use "stale" information from $t - d_t$ (or $t - n_t$) where the d_t is an external, potentially

[15] By the triangle inequality, $\big|\, \|\boldsymbol{x} - \boldsymbol{y}\| - \|\boldsymbol{x} - \boldsymbol{y}'\| \,\big| \leq \|\boldsymbol{y} - \boldsymbol{y}'\|$ for fixed $\boldsymbol{x}$.

[16] Convexity is also required for the meta learning framework.

adversarial, imposed delay. In the delayed-feedback setting, the gradient $\boldsymbol{g}_t$ is not available at the end of round t but arrives at some future time $t + d_t$. The update in k-LAZYGD operates under a different paradigm. It exists within the standard, non-delayed OCO framework where the gradient $\boldsymbol{g}_t$ is revealed immediately. The "laziness" or "delay" inherent in the k-LAZYGD update is not an external constraint to be overcome but rather a deliberate, deterministic *algorithmic mechanism*. The algorithm has access to all gradients $\boldsymbol{g}_1, \dots, \boldsymbol{g}_t$ when computing $\boldsymbol{x}_{t+1}$ but chooses to anchor its update to a past iterate $\boldsymbol{x}_{t-n_t}$ and use an accumulation of recent gradients thereafter in order to avoid over-reactivity to noisy or anomalous gradients.

G k-LAZYGD ANALYSIS

G.1 Proof of Lemma 6

Lemma 6. Let $\{f_t(\cdot), \boldsymbol{u}_t\}_{t=1}^T$ be an arbitrary set of functions and comparators within $\mathcal{X}$, respectively. Let $r(\cdot)$ be strongly convex regularizer such that

$$\boldsymbol{x}_{t+1} \doteq \operatorname*{argmin}_{\boldsymbol{x}} h_{0:t}(\boldsymbol{x})$$

is well-defined, where $h_0(\boldsymbol{x}) \doteq I_{\mathcal{X}}(\boldsymbol{x}) + r(\boldsymbol{x})$, $\forall t \geq 1$:

$$h_t(\boldsymbol{x}) \doteq \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle \quad \boldsymbol{p}_t \in \partial \bar{f}_t(\boldsymbol{x}_t).$$

Then, the algorithm that selects the actions $\boldsymbol{x}_{t+1}, \forall t$ achieves the following dynamic regret bound:

$$\mathcal{R}_T \leq \sum_{t=1}^{T} \big(\overbrace{h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1})}^{\textbf{(I)}} \big) + \sum_{t=1}^{T-1} \overbrace{\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|}^{\textbf{(II)}} + \sum_{t=1}^{T-1} \big(\overbrace{h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t)}^{\textbf{(III)}} \big) \; + \; r(\boldsymbol{u}_1).$$

Proof. We start from a quantity that is the dynamic regret w.r.t. the $h_t(\cdot)$ functions. Then we decompose it further.

$$\begin{aligned}
&\sum_{t=1}^{T} h_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} h_t(\boldsymbol{u}_t) \\
&\quad = \sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{x}_t) - h_{0:t-1}(\boldsymbol{x}_t) \right) - \left(\sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{u}_t) - h_{0:t-1}(\boldsymbol{u}_t) \right) \right) \\
&\quad = \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=1}^{T} h_{0:t-1}(\boldsymbol{x}_t) - \left(\sum_{t=1}^{T} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=1}^{T} h_{0:t-1}(\boldsymbol{u}_t) \right) \\
&\quad = \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=0}^{T-1} h_{0:t}(\boldsymbol{x}_{t+1}) - \left(h_{0:T}(\boldsymbol{u}_T) + \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=0}^{T-1} h_{0:t}(\boldsymbol{u}_{t+1}) \right) \\
&\quad \leq \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{x}_{t+1}) - \left(h_{0:T}(\boldsymbol{x}_{T+1}) + \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_{t+1}) - r(\boldsymbol{u}_1) \right) \\
&\quad = \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_{t+1}) - \left(\sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_{t+1}) \right) + r(\boldsymbol{u}_1).
\end{aligned}$$

The inequality holds because of the update rule for each $\boldsymbol{x}_{t+1}$ for any t. I.e.,

$$h_{0:T}(\boldsymbol{x}_{T+1}) \leq h_{0:T}(\boldsymbol{u}_T).$$

Also, $h_0(\boldsymbol{u}_1) = r(\boldsymbol{u}_1)$.

Writing h_t of the LHS explicitly, and adding the total switching cost to both sides, we get

$$\sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{x}_t - \boldsymbol{u}_t \rangle + \sum_{t=1}^{T-1} \|\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\| \leq \sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) \right) + \sum_{t=1}^{T-1} \left(h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t) \right)$$

$$+\sum_{t=1}^{T-1}\|\boldsymbol{x}_t-\boldsymbol{x}_{t-1}\|+r(\boldsymbol{u}_1).$$

Lastly, by convexity we have that

$$f_t(\boldsymbol{x}_t)-f_t(\boldsymbol{u}_t)=\bar{f}_t(\boldsymbol{x}_t)-\bar{f}_t(\boldsymbol{u}_t)\leq\langle\boldsymbol{p}_t,\boldsymbol{x}_t-\boldsymbol{u}_t\rangle$$

which completes the proof □

We now state the helper Lemma 9, which provides an additional characterization of the k-LazyGD iterate using their FTRL from. Specifically, we show that $\hat{\boldsymbol{x}}_t$ is the minimizer not only of the original update rule in (3) in the main paper, but also of a related linearized expression.

Lemma 9. *For any $\boldsymbol{x}_t\in\mathcal{X}$ satisfying*

$$\boldsymbol{x}_t\doteq\arg\min_{\boldsymbol{x}}h_{0:t-1}(\boldsymbol{x}),$$

it also holds that

$$\boldsymbol{x}_t\doteq\arg\min_{\boldsymbol{x}}\Big(h_{0:t-1}(\boldsymbol{x})+\langle\boldsymbol{g}_t^I,\boldsymbol{x}\rangle\Big),$$

where $\boldsymbol{g}_t^I$ is chosen according to (21).

Proof. We are given that $\boldsymbol{x}_t=\arg\min_{\boldsymbol{x}\in\mathcal{X}}h_{0:t-1}(\boldsymbol{x})$. By the first-order optimality condition for constrained minimization (e.g., (Beck, 2017, Thm. 3.67)), it follows that

$$-\nabla h_{1:t-1}(\boldsymbol{x}_t)\in\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t).\tag{27}$$

We now consider the objective $h_{0:t-1}(\boldsymbol{x})+\langle\boldsymbol{g}_t^I,\boldsymbol{x}\rangle$ and verify that $\boldsymbol{x}_t$ also minimizes this function. The corresponding optimality condition is:

$$-\nabla h_{1:t-1}(\boldsymbol{y})-\boldsymbol{g}_t^I\in\mathcal{N}_{\mathcal{X}}(\boldsymbol{y}),$$

which, when evaluated at $\boldsymbol{y}=\boldsymbol{x}_t$, becomes:

$$-\nabla h_{1:t-1}(\boldsymbol{x}_t)-\boldsymbol{g}_t^I\in\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t).\tag{28}$$

We now inspect the two possible values of $\boldsymbol{g}_t^I$ as defined in (21):

(i) If $\boldsymbol{g}_t^I=0$, then (28) reduces to (27), which holds by assumption.

(ii) If $\boldsymbol{g}_t^I=-(\boldsymbol{p}_{1:t-1}+\sigma\boldsymbol{x}_t)$, then

$$-\nabla h_{1:t-1}(\boldsymbol{x}_t)-\boldsymbol{g}_t^I=-\boldsymbol{p}_{1:t}-\sigma\boldsymbol{x}_t-\boldsymbol{g}_t^I=0.$$

and hence (28) becomes $0\in\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$, which always holds.

In both cases, $\boldsymbol{x}_t$ satisfies the optimality condition for minimizing $h_{0:t-1}(\boldsymbol{x})+\langle\boldsymbol{g}_t^I,\boldsymbol{x}\rangle$ over $\mathcal{X}$, and is therefore a valid minimizer. □

G.2 Proof of Lemma 7

Lemma 7 Under the Lipschitzness of the loss functions and compactness of the domain, k-LazyGD iterates guarantee the following bounds on each part of the dynamic FTRL lemma for any t:

$(\mathbf{I})\leq\min\big(G\delta_t,\ \frac{G^2}{2\sigma}\big)$,

$(\mathbf{II})\leq\frac{G}{\sigma}$,

$(\mathbf{III})\leq(2R\sigma+kG)\,\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|$,

Lemma 7.a

$$(\mathbf{I}) \;\le\; \min\left(G\delta_t,\, \frac{G^2}{2\sigma}\right).$$

Proof. By strong convexity of $h_{0:t}$, for any $\boldsymbol{s}_t \in \partial h_{0:t}(\boldsymbol{x}_t)$,

$$h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) \;\le\; \langle \boldsymbol{s}_t, \boldsymbol{x}_{t+1} - \boldsymbol{x}_t\rangle - \frac{\sigma}{2}\delta_t^2. \tag{29}$$

From Lemma 9, the optimality condition of the update step at $\boldsymbol{x}_t$ gives $0 \in \partial h_{0:t-1}(\boldsymbol{x}_t) + \boldsymbol{g}_t^I$ That is, $-\boldsymbol{g}_t^I \in \partial h_{0:t-1}(\boldsymbol{x}_t)$. Since $\boldsymbol{p}_t = \boldsymbol{g}_t^I + \boldsymbol{g}_t \in \partial h_t(\boldsymbol{x}_t)$, it follows that

$$\boldsymbol{g}_t = -\boldsymbol{g}_t^I + (\boldsymbol{g}_t^I + \boldsymbol{g}_t) \in \partial h_{0:t-1}(\boldsymbol{x}_t) + \partial h_t(\boldsymbol{x}_t) \subseteq \partial h_{0:t}(\boldsymbol{x}_t),$$

where the last step uses the subdifferential sum rule. Thus we may take $\boldsymbol{s}_t = \boldsymbol{g}_t$ in (29), giving

$$(\mathbf{I}) \;\le\; \langle \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{x}_t\rangle \le G\delta_t. \tag{30}$$

For the alternative bound, we use $ax - \frac{b}{2}x^2 \;\le\; \frac{a^2}{2b}, \quad a, b > 0$ to obtain

$$(\mathbf{I}) \;\le\; \|\boldsymbol{g}_t\|\delta_t - \frac{\sigma}{2}\delta_t^2 \;\le\; \frac{G^2}{2\sigma}. \tag{31}$$

Combining (30) and (31), we get the result □

Lemma 7.b

$$(\mathbf{II}) \;\le\; \frac{G}{\sigma}.$$

To uniformly bound the switching cost and prove this lemma, we recall convex conjugates and some of their key properties. With pruning in play, non-regular behavior arises at block boundaries when $n_t = k - 1$, (when t is a multiple of k). By leveraging the smoothness of the conjugate function, we can handle all time steps uniformly, including those boundary cases.

Definition. The convex conjugate of a function r is defined as

$$r^*(\boldsymbol{g}) = \sup_{\boldsymbol{x}\in\mathbb{R}^d} \{\langle \boldsymbol{g}, \boldsymbol{x}\rangle - r(\boldsymbol{x})\}.$$

Properties. Let f be a 1- strongly convex function w.r.t some norm $\|\cdot\|$ From the classical results of Shalev-Shwartz (2007, Lemma 15), the following hold:

- $f^*(\cdot)$ is differentiable and 1-smooth with respect to the dual norm $\|\cdot\|_*$.
- $\operatorname{argmin}_{\boldsymbol{x}}\{\langle \boldsymbol{v}, \boldsymbol{x}\rangle + f(\boldsymbol{x})\} = \nabla f^*(-\boldsymbol{v})$.

Note that, by the above definition and properties, we can write k-LAZYGD in yet another way as

$$\boldsymbol{x}_t = \nabla h_0^*(-\boldsymbol{p}_{1:t-1}).$$

Proof. (of **Lemma 7.b**) The regularizer h_0 is 1-strongly convex w.r.t. the scaled norm $\|\cdot\|_s \doteq \sqrt{\sigma}\|\cdot\|$, hence its conjugate h_0^* is 1-smooth w.r.t. $\|\cdot\|_s$. By Lemma 9, we may also write

$$\boldsymbol{x}_t = \nabla h_0^*(-\boldsymbol{p}_{1:t-1} - \boldsymbol{g}_t^I),$$

so that[17]

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_s = \left\|\nabla h_0^*(-\boldsymbol{p}_{1:t}) - \nabla h_0^*(-\boldsymbol{p}_{1:t-1} - \boldsymbol{g}_t^I)\right\|_s \;\le\; \|\boldsymbol{g}_t\|_s \;=\frac{\|\boldsymbol{g}_t\|}{\sqrt{\sigma}} \;\le\; \frac{G}{\sqrt{\sigma}}.$$

The first inequality follows from the smoothness of h_0, and the second from the Lipschitzness assumption. Hence,

$$\sqrt{\sigma}\;\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \le \frac{G}{\sqrt{\sigma}},$$

which implies the bound (by multiplying both sides by the positive $1/\sqrt{\sigma}$). □

[17] The additional characterization of Lemma 9 is used only for $\boldsymbol{x}_t$, not for $\boldsymbol{x}_{t+1}$.

Lemma 7.c

$$(\mathbf{III}) \;\le\; (2R\sigma + kG)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$$

Proof. From strong convexity

$$(\mathbf{III}) \le \|\boldsymbol{q}_t\|\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| - \frac{\sigma_{1:t}}{2}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|^2, \tag{32}$$

where $\boldsymbol{q}_t \in \partial h_{0:t}(\boldsymbol{u}_{t+1})$.

To investigate $\boldsymbol{q}_t$, we expand the subdifferential:

$$\partial h_{0:t}(\boldsymbol{x}) = \boldsymbol{p}_{1:t} + \sigma\boldsymbol{x} + \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}).$$

Choosing the zero vector from $\mathcal{N}_{\mathcal{X}}(\boldsymbol{u}_{t+1})$, we obtain: $\boldsymbol{q}_t = \boldsymbol{p}_{1:t} + \sigma\boldsymbol{u}_{t+1}$. Since $\|\boldsymbol{u}_{t+1}\| \le R$, it follows that

$$\|\boldsymbol{q}_t\| \le \|\boldsymbol{p}_{1:t}\| + R\sigma.$$

Substituting into (32), this yields[18] the bound:

$$(\mathbf{III}) \le (\|\boldsymbol{p}_{1:t}\| + R\sigma)\,\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|. \tag{33}$$

We now derive a bound on the state norm $\|\boldsymbol{p}_{1:t}\|$. Recall that in the proof of Theorem 1, we have shown that:

$$-\frac{\boldsymbol{p}_{1:t}}{\sigma} \;=\; \boldsymbol{x}_{t-n_t} - \frac{\boldsymbol{g}_{t-n_t:t}}{\sigma},$$

of which a direct consequence (from triangular inequality), is

$$\begin{aligned}\|\boldsymbol{p}_{1:t}\| &\le \sigma\|\boldsymbol{x}_{t-n_t}\| + \|\boldsymbol{g}_{t-n_t:t}\| \le R\sigma + \sum_{s=t-n_t}^{t}\|\boldsymbol{g}_s\| \\ &\le R\sigma + (n_t+1)G \le R\sigma + kG \end{aligned} \tag{34}$$

The results follows form (33) and (34) □

G.3 Proof of Theorem 5

Theorem 5. For any sequence of convex losses and comparators, the dynamic regret with switching cost of k-LAZYGD satisfies

$$\mathcal{R}_T \le \sum_{t=1}^{T}\min\big(G\delta_t,\, \frac{G^2}{2\sigma}\big) + (2R\sigma + kG)P_T + \frac{\sigma R^2}{2} + \sum_{t=1}^{T}\min(\delta_t, \frac{G}{\sigma}),$$

where $\delta_t \doteq \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|$. Moreover, setting $k = \lfloor c\sqrt{2RT/P_T}\rfloor, c \ge 1$ and $\sigma = \sigma \;\doteq \sqrt{(G^2{+}2G)T\,/\,(4RP_T{+}R^2)}$ gives $\mathcal{R}_T = \mathcal{O}\big(\sqrt{(P_T+1)T}\big)$.

Proof. Starting from the result of Lemma 6, and substituting the bounds for each part from Lemma 7, we obtain

$$\begin{aligned}\mathcal{R}_T &\le \sum_{t=1}^{T}\min\big(G\delta_t,\, \frac{G^2}{2\sigma}\big) \;+\; (2R\sigma + kG)P_T \;+\; \frac{\sigma R^2}{2} \;+\; \sum_{t=1}^{T}\min(\delta_t, \frac{G}{\sigma}), \\ &\le \frac{G^2T}{2\sigma} + \frac{\sigma}{2}R^2 + 2R\sigma P_T + k\,GP_T + \frac{G}{\sigma}T \\ &= \sqrt{T}\,\sqrt{(G^2+2G)(R^2+4RP_T)} + G\sqrt{TP_T}. \end{aligned} \tag{35}$$

where we used that $r(\boldsymbol{u}_1) \le \sigma\frac{R^2}{2}$. The second inequality follows by selecting the second branch of the min terms, and setting $k \le c\sqrt{T/P_T}$. The last equality follows by substituting the optimal σ: $\sigma \;= \sqrt{\frac{T\,(G^2+2G)}{R^2+4RP_T}}$. □

Remark. *Note on the choice of k.* As noted in Footnote 11, the choice $k = \lfloor c\sqrt{T/P_T}\rfloor$ implicitly assumes $P_T > 0$. In the static case $P_T = 0$, one may simply take $k = T$. More generally, this is also harmless whenever $P_T \le T^{-1}$ [2], since then $kP_T \le TP_T \le \sqrt{T}$, so the overall regret order in (24) is unchanged.

[18] Dropping the negative quadratic may seem wasteful, motivating the use of $ax - \frac{b}{2}x^2 \le \frac{a^2}{2b}$. However, this hides x as a complexity term. Fixing x, even to its minimizer a/b, makes the bound linear regardless.

H THE ENSEMBLE FRAMEWORK ANALYSIS

As mentioned in the paper, we use the meta learner of SAder from Zhang et al. (2021), but over an ensemble that vary *both* the regularization/learning rate and the laziness slack (see Fig.14 below). For completeness, we provide their algorithm and its guarantee below.

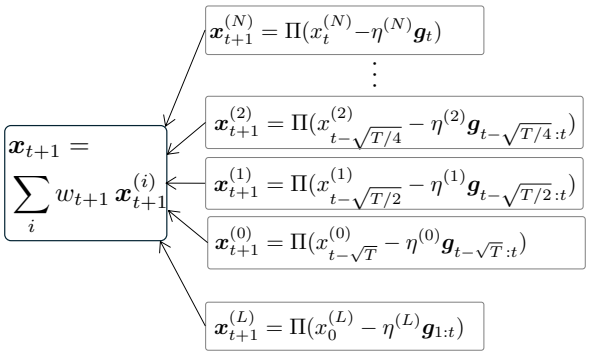

Figure 14: Illustration of the meta-learning ensemble at a round $t > \sqrt{T}$. Each base learner corresponds to a different pair of hyperparameters: a learning rate/regularization scale and a laziness slack. Negative indices are interpreted as 0, and non-integer indices are rounded down so that all update ranges are well defined. The last lazy learner (L) corresponds to the fully lazy endpoint, i.e., the static case $P_T = 0$; see the remark in Appendix G.3.

Alg. 3 SAder

Input: A step size β, the set $\mathcal{H}$ containing the regularization/laziness slack for each expert.
Output: $\{\boldsymbol{x}_t\}_{t=1}^T$.

1: Activate the set of experts $\{E^\sigma | \sigma \in \mathcal{H}\}$
2: Maintain $\sigma^{(1)} \geq \sigma^{(2)} \geq \dots \sigma^{(N)}$, and initialize $w_1^{\sigma^{(i)}} = \frac{C}{i(i+1)}$.
3: **for** $t = 1, 2, \dots, T$ **do**
4: Receive $\boldsymbol{x}_t^\sigma$ from $E^\sigma, \forall \sigma \in \mathcal{H}$
5: Output the weighted average $\boldsymbol{x}_t = \sum_{\sigma \in \mathcal{H}} w_t^\sigma \boldsymbol{x}_t^\sigma$
6: Observe loss $f_t(\cdot)$ and record $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$
7: Updates the weight of each expert by (36)
8: Send $\boldsymbol{g}_t$ to all experts.
9: **end for**

The SAder algorithm follows the following weight update rule:

$$w_{t+1}^\sigma = \frac{1}{M} w_t^\sigma e^{-\beta\ r_t(\boldsymbol{x}_t^\sigma\ \boldsymbol{x}_{t-1}^\sigma)}, \quad M \doteq \sum_{\sigma \in \mathcal{H}} w_t^\sigma e^{-\beta\ r_t(\boldsymbol{x}_t^\sigma\ \boldsymbol{x}_{t-1}^\sigma)}, \quad \text{where } \ell_t(\boldsymbol{x}, \boldsymbol{y}) \doteq \langle \boldsymbol{g}_t, \boldsymbol{x} - \boldsymbol{x}_t \rangle + \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{36}$$

Lemma 10 (Ensemble regret)**.** *(Zhang et al., 2021, Lemma 3) Run Algorithm 3 with*

$$\beta \doteq \frac{2}{(2G+1)D} \sqrt{\frac{2}{5T}},$$

over the expert set of k-LAZYGD learners:

$$\mathcal{H} \doteq \left\{ \tfrac{B}{2^{i-1}} : i = 1, \dots, N \right\},\ N \doteq \left\lceil \tfrac{1}{2} \log_2(8T - 7) \right\rceil + 1.$$

indexing each k-LAZYGD expert directly by $\sigma \in \mathcal{H}$ and setting $k = \max(1, \lfloor \frac{2R}{G} \sigma \rfloor)$. Let $\boldsymbol{x}_t$ be the meta-learner's decisions and $\boldsymbol{x}_t^\sigma$ those of expert σ. Then, for any $\sigma \in \mathcal{H}$,

$$\sum_{t=1}^T \Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t \rangle + \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \Big) \ - \ \sum_{t=1}^T \Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t^\sigma \rangle + \|\boldsymbol{x}_{t+1}^\sigma - \boldsymbol{x}_t^\sigma\| \Big) \ \leq \ (2G+1)\, 2R \left(\ln\left(1/w_1^\sigma\right) + 1 \right) \sqrt{\frac{5}{8} T}.$$

H.1 Proof of Theorem 8

Theorem 8. For any comparator sequence, running the meta-learner of (Zhang et al., 2021, "SAder") over a set of k-LAZYGD experts from $\mathcal{H}$, guarantees

$$\mathcal{R}_T = \mathcal{O}\Big(\sqrt{(P_T+1)T}\Big).$$

Proof. (of **Theorem 8**) Recall the regret bound for any k-LAZYGD expert (i.e., for any σ) from (35):

$$\sum_{t=1}^{T}\Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t^{\sigma}\rangle + \|\boldsymbol{x}_{t+1}^{\sigma} - \boldsymbol{x}_t^{\sigma}\| \;-\; f_t(\boldsymbol{u}_t)\Big) \le \frac{G^2T}{2\sigma} + \frac{\sigma}{2}R^2 + 2R\sigma P_T + kGP_T + \frac{G}{\sigma}T. \tag{37}$$

In the main paper, we defined the expert $\sigma^{(s)}$ where

$$\sigma^{(s)} = \frac{B}{2^{s-1}}, \quad s \doteq \frac{1}{2}\lfloor \log\big(1+4P_T/R\big)\rfloor + 1$$

writing the above bound in (37) for expert $\sigma^{(s)}$, and recalling $k = \max(1, \lfloor \frac{2R}{G}\sigma \rfloor)$

$$\begin{aligned}
\sum_{t=1}^{T}\Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t^{\sigma^{(s)}}\rangle + \|\boldsymbol{x}_{t+1}^{\sigma^{(s)}} - \boldsymbol{x}_t^{\sigma^{(s)}}\| \;-\; f_t(\boldsymbol{u}_t)\Big) &\le \frac{(G^2/2+G)T}{\sigma^{(s)}} + \sigma^{(s)}\left(R^2/2 + 2RP_T + \max(\tfrac{GP_T}{\sigma^{(s)}}, 2RP_T)\right) \\
&\le \frac{(G^2/2+G)T}{\sigma} + 2\sigma\;\left(R^2/2 + 2RP_T + \max(\tfrac{GP_T}{\sigma^{\star}}, 2RP_T)\right) \;\le\; \frac{(G^2/2+G)T}{\sigma} + 2\sigma\;\left(R^2/2 + 2RP_T + 2\sqrt{2}RP_T\right) \\
&= \sqrt{(G^2/2+G)T}\left(\sqrt{2RP_T + R^2/2} + \frac{5RP_T + R^2/2}{\sqrt{RP_T + R^2/4}}\right) \;=\; \mathcal{O}\big(\sqrt{(P_T+1)T}\big)
\end{aligned} \tag{38}$$

where the second inequality follows since

$$\frac{B}{2^{s-1}} \;\ge\; \frac{B}{2^{s^*-1}} \;\ge\; \frac{B}{2^{s}}, \;\iff\; \sigma^{(s)} \;\ge\; \sigma \;\;\ge\; \frac{\sigma^{(s)}}{2}, \qquad s \;\doteq \tfrac{1}{2}\log\big(\sqrt{1+4P_T/R}\big) + 1.$$

The third inequality follows from $\sigma \;\ge \frac{G}{2\sqrt{2}R}$ Lastly, since expert $\sigma^{(s)}$ is in the set $\mathcal{H}$ by construction, we can write the result of Lemma. 10, for this specific expert:

$$\sum_{t=1}^{T}\Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t\rangle + \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|\Big) - \sum_{t=1}^{T}\Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t^{(\sigma^s)}\rangle + \|\boldsymbol{x}_{t+1}^{(\sigma^s)} - \boldsymbol{x}_t^{(\sigma^s)}\|\Big) \le (2G+1)\,2R\left(\ln\big(1/w_1^{\sigma}\big)+1\right)\sqrt{\frac{5}{8}T}. \tag{39}$$

Summing (39) and (38) cancels the $\sum_{t=1}^{T}\Big(\langle \boldsymbol{g}_t, \boldsymbol{x}_t^{\sigma^{(s)}}\rangle + \|\boldsymbol{x}_{t+1}^{\sigma^{(s)}} - \boldsymbol{x}_t^{\sigma^{(s)}}\|\Big)$ term. The proof is then completed using

$$w_1^{\sigma^s} \ge \frac{1}{(s+1)^2}$$

and hence

$$\ln(1/w_1^{\sigma^s}) \le 2\ln(s+1) \le 2\ln(\log(\sqrt{1+6P_T/R}+1))$$

□

I ADDITIONAL DETAILS ON LAZINESS ADVANTAGES

I.1 Variance-based Switching Cost

Beyond the stability properties formalized in Propositions 3 and 4, laziness also yields a variance-based switching bound. unlike those results, this bound requires no additional assumptions on the domain or on the magnitude/direction of the accumulated gradients. Instead of tying movement to the magnitude of the most recent gradient, it relates switching to its deviation from the running mean. This is a classical result for FTRL based on Hazan and Kale (2010, Lemma 6).

Proposition 11. *For any convex domain and sequence of gradients, fix $0 \le n_t < k-1$ (i.e., within a lazy phase). Let $\boldsymbol{G}_t \doteq \boldsymbol{g}_{t-n_t:t-1}$. and define the average state*

$$\boldsymbol{\mu}_t = \frac{\boldsymbol{G}_t}{n_t}$$

where $\boldsymbol{G}_t$ is the sum of gradients accumulated in the current lazy block. Then

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \;\le\; \frac{5}{4\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{4R}{n_t}.$$

Proof. We distinguish two cases based on the growth of $\boldsymbol{G}_t$

Case 1: $\|\boldsymbol{G}_t\| \le 4R\sigma$:

$$\begin{aligned}\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| &\overset{\text{Lem. 7.b}}{\le} \frac{1}{\sigma}\|\boldsymbol{g}_t\| = \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t + \boldsymbol{\mu}_t\| \\ &\le \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{1}{\sigma}\|\boldsymbol{\mu}_t\| \\ &\le \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{4R\|\boldsymbol{\mu}_t\|}{\|\boldsymbol{G}_t\|} = \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{4R}{n_t}\end{aligned}$$

Case 2: $\|\boldsymbol{G}_t\| \ge 4R\sigma$: Here, we show that

$$\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| \le \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{R\|\boldsymbol{g}_t\|}{\|\boldsymbol{G}_t\|}$$

From which the result follows since by the case assumption:

$$\frac{R\|\boldsymbol{g}_t\|}{\|\boldsymbol{G}_t\|} \le \frac{R\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + R\|\boldsymbol{\mu}_t\|}{\|\boldsymbol{G}_t\|} \le \frac{1}{4\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{R}{n_t}.$$

Consider the triangle $0, \boldsymbol{x}_t, \boldsymbol{y}_t$, and note that by the sines law

$$\sin(\theta) = \frac{\|\boldsymbol{x}_t\|\sin(\varphi)}{\|\boldsymbol{y}_t\|} \le \frac{\sigma R}{\|\boldsymbol{G}_t\|}, \tag{40}$$

where θ is the angle between $\boldsymbol{y}_t$ and $\boldsymbol{y}_t - \boldsymbol{x}_t$, or equivalently, between their unit vectors $\boldsymbol{v}$ and $\boldsymbol{u}$, respectively (see Fig. 15). Let $\boldsymbol{g}_t = \boldsymbol{g}_t^{\boldsymbol{v}} + \boldsymbol{g}_t^{\boldsymbol{u}}$ be the component of the gradient along the said directions.

Consider the point $\boldsymbol{y}_t - \frac{1}{\sigma}\boldsymbol{g}_t^{\boldsymbol{u}}$, we know that:

(i) it is outside $\boldsymbol{X}$:

$$\|\boldsymbol{y}_t - \frac{1}{\sigma}\boldsymbol{g}_t^{\boldsymbol{u}}\| \ge \|\boldsymbol{y}_t\| - \frac{1}{\sigma}\|\boldsymbol{g}_t^{\boldsymbol{u}}\| \ge \frac{1}{\sigma}\|\boldsymbol{G}_t\| - \frac{G}{\sigma} \ge 4R - \frac{G}{\sigma} \overset{(\sigma \ge G \;\ge\; 2\sqrt{2}R)}{\ge} R.$$

Where we used a crude lower bound on $\sigma = \sqrt{(G^2 + 2G)T/(4RP_T + R^2)}$.

(ii) its projection to $\mathcal{X}$ is $\boldsymbol{x}_t$ since it is on the same line as $\boldsymbol{y}_t - \boldsymbol{x}_t$. Thus,

$$\|\boldsymbol{x}_t - \boldsymbol{x}_{t+1}\| \le \|\boldsymbol{y}_{t+1} - (\boldsymbol{y}_t - \frac{1}{\sigma}\boldsymbol{g}_t^{\boldsymbol{u}})\| = \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{g}_t^{\boldsymbol{u}}\|.$$

Lastly,

$$\begin{aligned}\frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{g}_t^{\boldsymbol{u}}\| &\overset{(a)}{\le} \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{z}\| \overset{(b)}{\le} \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{g}_t^{\boldsymbol{v}}\| + \frac{1}{\sigma}\|\boldsymbol{g}_t^{\boldsymbol{v}} - \boldsymbol{z}\| \\ &\overset{(c)}{\le} \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{1}{\sigma}\|\boldsymbol{g}_t^{\boldsymbol{v}} - \boldsymbol{z}\| = \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{1}{\sigma}\|\boldsymbol{g}_t^{\boldsymbol{v}}\|\sin(\theta) \\ &\overset{(d)}{\le} \frac{1}{\sigma}\|\boldsymbol{g}_t - \boldsymbol{\mu}_t\| + \frac{R\|\boldsymbol{g}_t\|}{\|\boldsymbol{G}_t\|},\end{aligned}$$

where (a) holds $\forall \boldsymbol{z}$ in the span $\boldsymbol{u}$, (b) by the triangular inquality, (c) since $\boldsymbol{\mu}_t$ is in the span of $\boldsymbol{v}$, and $\boldsymbol{g}_t^{\boldsymbol{v}}$ is the projection of $\boldsymbol{v}_t$ onto the span of $\boldsymbol{v}$, and (d) from $\|\boldsymbol{g}_t^{\boldsymbol{v}}\| \le \|\boldsymbol{g}_t\|$ and (40). □

Remarks. For the fully lazy case ($k = T$), the result shows that switching can be bounded as

$$\Theta\big(\|\boldsymbol{g}_t - \tfrac{1}{t}\boldsymbol{g}_{1:t}\| + \tfrac{1}{t} \big),$$

namely in terms of the deviation of the most recent gradient from the running mean, rather than its raw magnitude. In the k-lazy case, the same principle applies phase by phase, with the bound expressed relative to the average within each lazy block. This perspective, originally from Hazan and Kale (2010, Lemma 6), shows an additional laziness advantage that naturally attenuates unnecessary movement by exploiting gradient concentration, without requiring the additional domain or gradient assumptions of Propositions 3 and 4.

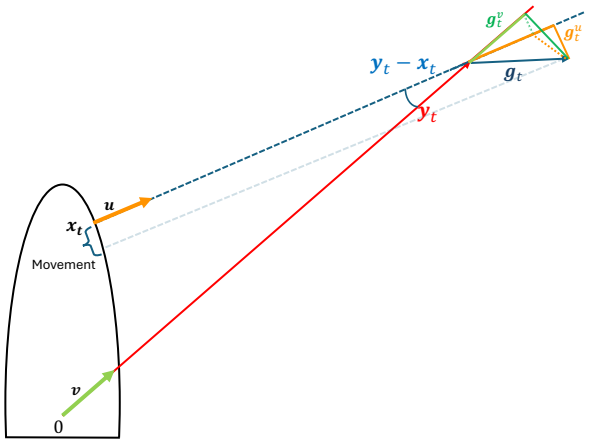

Figure 15: Geometric illustration for Prop. 3.

I.2 On the Contractive Property of the Projection Map

In the main text, we used a simple but useful property of the normalization map $\boldsymbol{x} \mapsto \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$, which is basically the Euclidean projection onto the unit ℓ_2 ball. For completeness, we restate it here with a proof.

Intuitively, the projection of points lying outside the ℓ_2 ball becomes *closer* as those points move farther away from the ball's boundary. This expresses the *contractive* behavior of the projection operator with respect to the distance from the feasible region. For clarity, we present the result for the unit ball ($R = 1$); the same argument applies to any radius $R > 0$ by simple scaling.

Lemma 12. *Let $f : \mathbb{R}^n \setminus \{0\} - \mathbb{R}^n$ be the mapping $f(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$. The Jacobian of f, denoted $J_f(\boldsymbol{x})$ or equivalently $Df(\boldsymbol{x})$, has an operator norm $\|J_f(\boldsymbol{x})\|_{op} = \frac{1}{\|\boldsymbol{x}\|}$.*

Proof. Write $f(\boldsymbol{x}) = \boldsymbol{x}\,\|\boldsymbol{x}\|^{-1} = g(\boldsymbol{x})\,h(\boldsymbol{x})$ with $g(\boldsymbol{x}) = \boldsymbol{x}$ and $h(\boldsymbol{x}) = \|\boldsymbol{x}\|^{-1}$. For any $\boldsymbol{v} \in \mathbb{R}^n$, the product rule gives

$$J_f(\boldsymbol{x})\boldsymbol{v} = (J_g(\boldsymbol{x})\boldsymbol{v})h(\boldsymbol{x}) + g(\boldsymbol{x})(J_h(\boldsymbol{x})\boldsymbol{v}).$$

Since $J_g(\boldsymbol{x}) = I$, we get $J_g(\boldsymbol{x})\boldsymbol{v} = \boldsymbol{v}$. The scalar function $h(\boldsymbol{x}) = (\langle \boldsymbol{x}, \boldsymbol{x}\rangle)^{-1\ 2}$ has gradient $\nabla h(\boldsymbol{x}) = -\boldsymbol{x}/\|\boldsymbol{x}\|^3$, so $J_h(\boldsymbol{x})\boldsymbol{v} = \langle \nabla h(\boldsymbol{x}), \boldsymbol{v}\rangle = -\frac{\langle \boldsymbol{x}\ \boldsymbol{v}\rangle}{\|\boldsymbol{x}\|^3}$. Substituting,

$$J_f(\boldsymbol{x})\boldsymbol{v} = \tfrac{1}{\|\boldsymbol{x}\|}\boldsymbol{v} - \tfrac{\boldsymbol{x}(\langle \boldsymbol{x}\ \boldsymbol{v}\rangle)}{\|\boldsymbol{x}\|^3} = \tfrac{1}{\|\boldsymbol{x}\|}\big(\boldsymbol{v} - \hat{\boldsymbol{x}}\langle \hat{\boldsymbol{x}}, \boldsymbol{v}\rangle\big),$$

where $\hat{\boldsymbol{x}} = \boldsymbol{x}/\|\boldsymbol{x}\|$. The map $\boldsymbol{v} \mapsto \boldsymbol{v} - \hat{\boldsymbol{x}}\langle \hat{\boldsymbol{x}}, \boldsymbol{v}\rangle$ is the orthogonal projection onto $\boldsymbol{x}^{\perp}$. Thus

$$\|J_f(\boldsymbol{x})\boldsymbol{v}\| = \tfrac{1}{\|\boldsymbol{x}\|}\|P_{x^{\perp}}\boldsymbol{v}\| \le \tfrac{1}{\|\boldsymbol{x}\|}\|\boldsymbol{v}\|.$$

Taking the supremum over $\|\boldsymbol{v}\| = 1$ yields $\|J_f(\boldsymbol{x})\|_{\text{op}} \le 1/\|\boldsymbol{x}\|$. Hence $\|J_f(\boldsymbol{x})\|_{\text{op}} = 1/\|\boldsymbol{x}\|$. □