WHAT SHAPE SHOULD A RULE BE? A GAUGE-THEORETIC FOUNDATION FOR ADAPTIVE ONLINE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adaptive first-order methods commit, each round, to a set of admissible step directions revised from past gradients; we ask what shape that set should take, and whether it must be an ellipsoid. We cast the proximal function as the squared gauge of a convex absorbing set C_t and prove a regret bound for any monotone sequence of such sets with strongly convex squared gauge. We give a rotated-ellipsoidal instance with an AdaGrad-type $O(\sqrt{T})$ guarantee and a consistent data-driven basis estimator, and study polytopal methods (rotated hyperrectangles, adaptive facets) empirically. Rotated geometries converge markedly faster than axis-aligned baselines with a line search and early in neural-network training.

1 INTRODUCTION

Motivation. Adaptive first-order methods – AdaGrad (Duchi et al., 2011a), RMSProp, and Adam (Kingma & Ba, 2014) – are the default choice for training machine learning models today, from convex online-learning problems to large-scale neural networks. Their defining idea is to let the optimizer revise, from the gradients it has already observed, a local notion of "distance" with respect to which the next step is taken, rather than committing to a single, isotropic geometry, whose uniform step scale causes the zigzagging characteristic of training under the curvature imbalance typical of deep networks (Aggarwal et al., 2018). This idea has inspired a large body of theoretical work – convergence guarantees, variance-reduced variants, and analyses that recover the best fixed geometry in hindsight – but almost all of it takes for granted *what kind of geometry* the method should adapt: a quadratic form, i.e. an ellipsoid, with theoretical effort spent on how to best set its parameters from data. Whether that geometric family is the right one to commit to at all has received comparatively little attention – a gap that has grown more visible as recent practical methods (Shampoo (Gupta et al., 2018), Muon (Jordan et al., 2024), and others) have begun adapting non-ellipsoidal geometries with real success, without a general theory to explain how this should work.

This paper asks that prior question directly: what shape should an adaptively-revised notion of distance take, and is it forced to be an ellipsoid? A simple example makes the question concrete. Ask what it means to project a point onto a line, and the answer that feels obvious — drop a perpendicular, find the one nearest point — depends on a choice nobody mentioned making. Under the Euclidean norm, that nearest point is unique. Under the ℓ_∞ norm, for a line lying along one of the norm's own axes, uniqueness disappears entirely: an entire segment, not one point, sits at the minimal distance (Deutsch, 2001). Nothing about the point or the line changed. What changed is the shape of the ball used to measure closeness. Every adaptive first-order method is, underneath its update rule, committing at each round to exactly this kind of choice – a set of directions it is willing to consider as an admissible next step – and revising that notion as it observes the consequence of each commitment. The central question of this paper is what shape that notion should take: not merely how far to step, but what *set* of directions counts as reasonable at all.

To see why this choice matters concretely, consider the elementary steepest-descent problem $d^\star = \arg\min_{\|d\|_2 \le 1} \langle \nabla f(x), d \rangle$, i.e., finding the direction of maximal local decrease of f at x under an ℓ_2 length budget. Writing $g = \nabla f(x)$, a first-order expansion gives $f(x+d) \approx f(x) + \langle g, d \rangle$,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

so we want $\langle g, d\rangle$ as negative as possible subject to $\|d\|_2 \le 1$. By Cauchy–Schwarz, $\langle g, d\rangle \ge -\|g\|_2\|d\|_2 \ge -\|g\|_2$, with equality iff d is a non-positive multiple of g and $\|d\|_2 = 1$; solving for the normalization gives $d^\star = -\frac{g}{\|g\|_2}$. The point of this classical computation is that the *shape* of the constraint set – here the Euclidean ball – is what turns "move opposite the gradient" into a specific, normalized direction. If the constraint set is instead an ellipsoid, a hyperrectangle, or a general polytope, the steepest-descent direction changes accordingly. Adaptive subgradient methods exploit exactly this fact: by reshaping the implicit unit ball used at each iteration to reflect the geometry of the gradients seen so far, they can achieve lower regret than methods that use a fixed, isotropic notion of distance throughout.

The object this paper studies is the squared *gauge* (Minkowski functional) of a convex absorbing set C: $\gamma_C(x) = \inf\{r > 0 : x \in rC\}$, which measures how much C must be scaled to reach x. Setting the proximal function at round t to $\psi_t(x) = \frac{1}{2}\gamma_{C_t}(x)^2$ lets the *shape* of C_t – not just its size or its orientation within a fixed family – be the object an adaptive method designs. When C is a polytope, γ_C is piecewise linear; when C is an ellipsoid, γ_C reduces to a quadratic-form norm $\sqrt{x^\top H x}$ for some $H \succ 0$ – and it is exactly this ellipsoidal case that AdaGrad uses, with $H_t = \delta I + \mathrm{diag}(G_t)^{1/2}$ (diagonal AdaGrad) or $H_t = \delta I + G_t^{1/2}$ (full-matrix AdaGrad), where $G_t = \sum_{\tau \le t} g_\tau g_\tau^\top$. We do not resolve, in general, which shape is best: Section 3.1 shows that the most natural way to pose that question – an unconstrained hindsight optimum over arbitrary C – is degenerate (Remark 3). Instead, we prove a general regret bound for any monotone sequence of convex absorbing C_t whose squared gauge is strongly convex (Theorem 2), and give a rotated-ellipsoidal instance with an AdaGrad-type $O(\sqrt{T})$ guarantee (Theorem 3). Guided by the dual-gauge term of Theorem 2 and by polarity (Theorem 1, Figure 1), we then study two polytopal methods – a rotated-hyperrectangle construction and a fully online facet-adaptive method (Sections 4–5), evaluating them empirically.

Two questions organize the paper: What regret bound holds for a general, monotonically shrinking sequence C_t, independent of its shape? How should C_t actually be designed from the observed subgradients, and in particular, can a polytopal C_t ever outperform an ellipsoid in practice?

For any monotone sequence of convex absorbing sets C_t with strongly convex squared gauge we prove a general regret bound (Theorem 2), with diagonal and full-matrix AdaGrad as ellipsoidal special cases. It covers a rotated-ellipsoidal construction – rotated diagonal AdaGrad – with an $O(\sqrt{T})$ regret bound (Theorem 3, Appendix B.4) and a consistent, data-driven estimator for its orientation (Proposition 1) via a one-time random-probe warm-up. We identify a degeneracy in the natural generalization of AdaGrad's hindsight-design argument to general absorbing sets (Remark 3). Finally, we study empirically a polytopal rotated-hyperrectangle method (Algorithm 1) and an algorithm that adapts a general polytope's facet normals online (Algorithm 3). Rotated geometries converge markedly faster than axis-aligned baselines when combined with a line search and a sufficiently long basis warm-up and early in neural-network training.

2 PRELIMINARIES

For $C \subseteq \mathbb{R}^d$, the *gauge* (Minkowski functional) of C is

$$\gamma_C(x) := \inf\{r > 0 : x \in rC\}, \qquad \gamma_C(x) := \infty \text{ if no such } r \text{ exists.}$$

C is *absorbing* if for every $x \in \mathbb{R}^d$ there is $t > 0$ with $\alpha x \in C$ for all $|\alpha| \le t$; for convex C this is equivalent to γ_C being finite everywhere. For an ellipsoid $C = \{x : x^\top Q x \le 1\}$ with $Q \succ 0$, $\gamma_C(x) = \sqrt{x^\top Q x}$.

For $C \ni 0$, the *polar* is $C^\circ = \{y \in \mathbb{R}^d : \langle y, x\rangle \le 1 \ \forall x \in C\}$, and the *support function* of C is $\sigma_C(y) = \sup_{x \in C}\langle y, x\rangle$. Since C° is the unit sublevel set of the positively homogeneous function σ_C,

$$\gamma_{C^\circ}(y) = \sigma_C(y) = \sup_{x \in C}\langle y, x\rangle.$$

We refer to γ_{C° as the *dual gauge* of C throughout.

For F convex and differentiable on a convex set Ω, the Bregman divergence is $B_F(p, q) = F(p) - F(q) - \langle \nabla F(q), p - q\rangle$. For the ellipsoidal gauge above, $F = \frac{1}{2}\gamma_C^2$ gives $B_F(x, y) = \frac{1}{2}(x - y)^\top Q(x - y)$, the identity underlying the rotated-ellipsoidal construction of Appendix B.4.

The following fact motivates why the design methods of Section 5 remain well defined even when a candidate primal shape is irregular during an online update.

Theorem 1 (Polarity convexifies; Bauschke et al. (2011)) *For any set $C \subseteq \mathbb{R}^d$ containing the origin, C° is closed and convex, whether or not C is convex.*

3 A General Regret Bound for Gauge-Based Mirror Descent

Let $X \subseteq R^d$ be convex, $\phi : X \to R$ a fixed (possibly zero) regularizer, and f_t a sequence of convex losses with $g_t \in \partial f_t(x_t)$. Let $\{C_t\}$ be a sequence of convex absorbing sets, and set $\psi_t(x) = \frac{1}{2}\gamma^2_{C_t}(x)$. Let $g_{1:t} = [g_1 \cdots g_t] \in \mathbb{R}^{d\times t}$ denote the matrix obtained by concatenating the subgradient sequence. For $i = 1, \ldots, d$, write $g_{1:t,i} := (g_{1,i}, \ldots, g_{t,i}) \in \mathbb{R}^t$ for its i-th row (the trajectory of coordinate i across rounds), and $\|g_{1:t,i}\|_2$ for its Euclidean norm – the notation of Duchi et al. (2011b), which we adopt throughout. The *composite mirror descent (CMD)* update (Tseng (2008); Duchi & Singer (2009); Duchi et al. (2010)) is

$$x_{t+1} = \arg\min_{x\in X} \Big\{\eta\langle g_t, x\rangle + \eta\phi(x) + B_{\psi_t}(x, x_t)\Big\}. \tag{CMD}$$

Define the regret $R_\phi(T) = \sum_{t=1}^T \big[f_t(x_t) + \phi(x_t) - f_t(x^*) - \phi(x^*)\big]$ for a comparator $x^* \in X$.

Theorem 2 (General gauge regret bound) *Suppose each $\psi_t = \frac{1}{2}\gamma^2_{C_t}$ is differentiable and 1-strongly convex with respect to γ_{C_t}, and that the sequence is monotonically non-decreasing, $\psi_{t+1}(x) \geq \psi_t(x)$ for all x (equivalently $C_{t+1} \subseteq C_t$). Then for any comparator $x^* \in X$, the iterates of (CMD) satisfy*

$$R_\phi(T) \leq \frac{1}{\eta}B_{\psi_1}(x^*, x_1) + \frac{1}{\eta}\sum_{t=1}^{T-1}\big[B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1})\big] + \eta\sum_{t=1}^{T}\gamma^2_{C_t^\circ}(g_t). \tag{$*$}$$

Proof. See Appendix B.3. □

Remark 1 *The bound $(*)$ makes no assumption on the shape of C_t beyond convexity, absorbency, monotonicity, and strong convexity of ψ_t. Diagonal and full-matrix AdaGrad are ellipsoidal special cases, $C_t = \{x : x^\top(\delta I + \mathrm{diag}(G_t)^{1/2})x \leq 1\}$ and $C_t = \{x : x^\top(\delta I + G_t^{1/2})x \leq 1\}$.*

Remark 2 *No polytope is strongly convex: a flat facet gives directions where the squared gauge's Bregman divergence vanishes while the gauge distance stays positive, and ψ_t is not differentiable across the ridges between facets. Theorem 2 therefore does* not *apply to polytopal C_t. Sections 4–5 study polytopal constructions as methods motivated by the dual-gauge term of $(*)$; the provably covered instance is the rotated ellipsoid of Appendix B.4.*

3.1 The Hindsight-Design Problem, and a Degeneracy to Avoid

Given the full gradient sequence $g_1, \ldots, g_T$ in hindsight, what is the best fixed absorbing set C to have used throughout? For quadratic forms, AdaGrad answers this with the closed form $s_i^\star = c\|g_{1:T,i}\|_2 / \sum_j \|g_{1:T,j}\|_2$, obtained by minimizing $\sum_t \langle g_t, \mathrm{diag}(s)^{-1} g_t\rangle$ over $\langle 1, s\rangle \leq c$.

Remark 3 *It is tempting to generalize this directly: replace s by an arbitrary absorbing set C, replace $\langle g, \mathrm{diag}(s)^{-1} g\rangle$ by $\gamma_{C^\circ}(g)^2$, and replace the trace budget by a gauge-diameter constraint $\sup_{x\in X}\gamma_C(x) \leq D$. This looks natural but is degenerate: the constraint $\sup_{x\in X}\gamma_C(x) \leq D$ is equivalent to $\frac{1}{D}X \subseteq C$ – a* lower *bound forcing C to be large enough to contain a fixed set. Since $\gamma_{C^\circ}(g) = \sup_{x\in C}\langle g, x\rangle$ is monotonically increasing in C (in the set-inclusion order), minimizing $\sum_t \gamma_{C^\circ}(g_t)^2$ subject to a pure lower bound on C is solved by shrinking C down to the smallest feasible set, $C^\star = \mathrm{conv}_{\mathrm{bal}}(\frac{1}{D}X)$ – a choice that does not depend on the gradients $g_1, \ldots, g_T$ at all. This directly contradicts the goal of shaping C around the observed gradient directions.*

The reason AdaGrad's own derivation does not hit this degeneracy is that its constraint, $\langle 1, s\rangle \leq c$, is an upper budget *on the primal matrix $\mathrm{diag}(s)$ itself – not a statement about containing a fixed*

reference set. In the special case X = the unit ℓ_∞ ball and $C = \{x^\top \mathrm{diag}(s)x \le 1\}$, one checks $\sup_{x\in X} \gamma_C(x)^2 = \langle 1, s\rangle$, so the two constraints happen to coincide numerically in that one case – which is presumably what motivated the generalization – but the equivalence is a coincidence of the hyperrectangle geometry of X, and it breaks for a general feasible set X or a general absorbing set C.

We resolve this by not attempting to solve a fully general hindsight problem over *arbitrary* C. Instead, in Section 4 we fix a *parametric family* of polytopes (rotated hyperrectangles) and design an *online* accumulation rule for the parameters directly.

4 Rotated Geometries

The reason we study a *hyperrectangle* is facet↔vertex duality (Figure 1): the gradient term of $(*)$ depends on C_t only through the dual gauge $\gamma_{C_t^\circ}(g_t)$, and a flat facet near the origin in direction g_t becomes a sharp, distant vertex of C_t° pointing at g_t – a large, direction-specific penalty reduction. An ellipsoid's polar is always another ellipsoid, so it can only trade off smoothly across *all* directions at once; only a polytope's corners make a sharp, direction-specific trade-off possible.

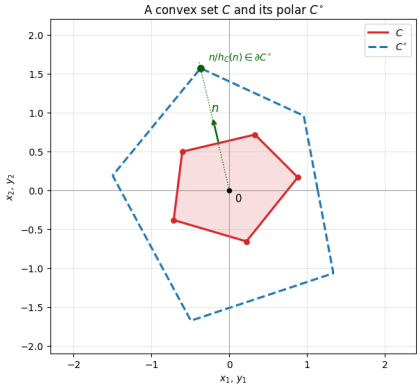


Figure 1: A convex set C (solid red pentagon) and its polar C° (dashed blue). Each facet of C, with outward normal n, contributes one constraint to C° and corresponds to the vertex $n/h_C(n)$ where that constraint is tight ($h_C(n) = \max_{x\in C}\langle n, x\rangle$); dually, each vertex of C becomes a facet of C°. C reaching farthest toward the right makes C° pull in closest there, and vice versa where C is thinnest (Remark 6).

A hyperrectangle is also the simplest polytope with more than one degree of freedom per axis: its steepest-descent direction is the closed-form sign vertex $d^\star = -\mathrm{sign}(g)$, no matrix solve required; its gauge and dual gauge are piecewise linear (Lemma 2), costing $O(d)$ once coordinates are expressed in the basis U (the rotation itself costs $O(d)$ for $U = I$, $O(dk)$ for a rank-k basis, and $O(d^2)$ for a dense one); and it is expressive enough to capture a rotation while still described by exactly d numbers $(\rho_1, \ldots, \rho_d)$ given a fixed basis, which is what makes an explicit online accumulation rule possible (Section 4.2), evaluated empirically (Section D). The fully general polytope of Section 5 trades this tractability for expressiveness; the hyperrectangle is the point on that trade-off curve where a rotated, non-ellipsoidal shape remains tractable to design online.

4.1 Construction

Fix an orthonormal basis $U = [u_1, \ldots, u_d] \in \mathbb{R}^{d\times d}$ (Section 4.3 discusses estimating U from data; for now it may be fixed, e.g. to the coordinate basis $U = I$, or set once from an initial batch of gradients). For radii $\rho = (\rho_1, \ldots, \rho_d)$, $\rho_i > 0$, define the hyperrectangle

$$C_{U,\rho} = \{x \in R^d : |\langle u_i, x\rangle| \le \rho_i,\ i = 1, \ldots, d\},$$

a genuine polytope with $2d$ facets (a rotated hyperrectangle; $U = I$ recovers the axis-aligned hyperrectangle used by diagonal AdaGrad's own D_∞ analysis). Since $\rho_i > 0$ for every i, $C_{U,\rho}$ is

absorbing (any $x \in \mathbb{R}^d$ lies in $rC_{U,\rho}$ once $r \geq \max_i |\langle u_i, x\rangle|/\rho_i$), so $\gamma_{C_{U,\rho}}$ is finite everywhere. Writing $\tilde{x}_i = \langle u_i, x\rangle$ for the coordinates of x in the basis U,

$$\gamma_{C_{U,\rho}}(x) = \max_i \frac{|\tilde{x}_i|}{\rho_i}, \qquad C^\circ_{U,\rho} = \operatorname{conv}\{\pm u_i/\rho_i : i = 1, \ldots, d\}, \qquad \gamma_{C^\circ_{U,\rho}}(g) = \sum_i \rho_i |\tilde{g}_i|,$$

where $\tilde{g} = U^\top g$ (see Lemma 2 in Appendix B.2). (The polar of a hyperrectangle is the cross-polytope on the same axes, and since U is an orthonormal basis the representation of g in that basis is unique, so the defining infimum for the gauge of the cross-polytope is attained by that unique representation.) The primal ball *shrinks* in direction u_i as $\rho_i \downarrow$, and correspondingly the dual gauge assigns *less* penalty to gradients aligned with directions of small ρ_i.

4.2 Online algorithm

Algorithm 1 Rotated-hyperrectangle Polytopal Adaptive Subgradient Method

1: **Input:** $\eta > 0$, $c > 0$, $\delta > 0$, orthonormal basis $U = [u_1, \ldots, u_d]$.
2: Initialize $x_1 = 0$, $r_{0,i} = 0$ for $i = 1, \ldots, d$.
3: **for** $t = 1, \ldots, T$ **do**
4: Suffer loss $f_t(x_t)$, receive $g_t \in \partial f_t(x_t)$.
5: $\tilde{g}_t \leftarrow U^\top g_t$
6: $r_{t,i} \leftarrow \sqrt{r^2_{t-1,i} + \tilde{g}^2_{t,i}}$ for $i = 1, \ldots, d$
7: $\rho_{t,i} \leftarrow \dfrac{c}{\delta + r_{t,i}}$ for $i = 1, \ldots, d$
8: $C_t \leftarrow C_{U,\rho_t}$
9: $x_{t+1} \leftarrow x_t - \eta\, \gamma_{C_t^\circ}(g_t)\, U\big(\rho_t \odot \operatorname{sign}(\tilde{g}_t)\big)$
10: **end for**

Line 9 is the steepest-descent step $x_t + \arg\min_d\{\langle g_t, d\rangle + \frac{1}{2\eta}\gamma_{C_t}(d)^2\}$ under the rotated hyperrectangle (Lemma 2), stated for $X = \mathbb{R}^d$, $\phi = 0$. It is not a Bregman step with ψ_t, which is not differentiable everywhere for a polytope. In this form η and c enter only through ηc^2. Since $r_{t,i}$ is non-decreasing, $\rho_{t,i}$ is non-increasing and $C_{t+1} \subseteq C_t$.

Remark 4 *Algorithm 1's step penalizes g_t through exactly the dual gauge $\gamma_{C_t^\circ}(g_t) = \sum_i \rho_{t,i}|\tilde{g}_{t,i}|$ appearing in $(*)$; we evaluate it empirically (Section D). The rotated ellipsoid of Appendix B.4, which replaces line 9 by a quadratic (CMD) step, is covered by Theorem 3.*

Remark 5 (Per-update cost relative to full-matrix AdaGrad) *Full-matrix AdaGrad does not dominate the rotated-hyperrectangle construction in computational cost. For a fixed basis U, the radius update and the step of Algorithm 1 require $O(d)$ operations in the U-coordinates. The change of basis $g \mapsto U^\top g$ adds $O(dk)$ operations for a rank-k basis and $O(d^2)$ for a dense one. By contrast, full-matrix AdaGrad requires $O(d^2)$ memory and the computation of a $d \times d$ matrix square root at every iteration. Consequently, whenever an informative low-rank basis is available, the construction recovers a substantial part of the benefit of full-matrix preconditioning at a fraction of its cost (Section D.2.2). The rotated-ellipsoidal variant of Appendix B.4 has the same per-update complexity and retains such a guarantee (Theorem 3).*

4.3 Estimating U from data

Algorithm 1 takes U as an input; we now make precise how to obtain it from data. The procedure below is shape-agnostic and applies equally to the ellipsoidal alternative of Appendix B.4.

Proposition 1 (Consistent basis recovery) *Suppose the gradients $g_1, g_2, \ldots$ available for estimation are i.i.d. with $\|g_t\|_2 \leq B$ almost surely, and population second moment $M := \mathbb{E}[gg^\top]$ exactly diagonalized by an orthonormal basis U, $M = U\operatorname{diag}(\lambda_1, \ldots, \lambda_d)\,U^\top$, with eigen-gap $\operatorname{gap} := \min_{i \neq j} |\lambda_i - \lambda_j| > 0$. Let $\widehat{M}_m := \frac{1}{m}\sum_{t=1}^m g_t g_t^\top$ and let $\hat{u}_i$ be the i-th eigenvector of*

$\widehat{M}_m$. *Then for any* $\delta \in (0,1)$*, with probability at least* $1-\delta$*, simultaneously for every* i*,*

$$\sin \angle(\hat{u}_i, u_i) \;\le\; \frac{CB^2}{\mathrm{gap}} \sqrt{\frac{\log(d/\delta)}{m}}$$

for a universal constant C*, where* $\angle(\cdot,\cdot)$ *is taken up to sign.*

Proof. See Appendix B.5. □

Algorithm 2 Warm-up Basis Estimation

1: **Input:** warm-up length m, dimension d.
2: Collect m gradients $g_1, \ldots, g_m$ by probing at independent, isotropically random points.
3: $\widehat{M} \leftarrow \frac{1}{m}\sum_{t=1}^{m} g_t g_t^\top$
4: $\hat{U} \leftarrow$ eigenvectors of $\widehat{M}$, sorted by eigenvalue.
5: **Return** $\hat{U}$, for use as U in Algorithm 1.

Algorithm 2 costs $O(md^2)$ to accumulate $\widehat{M}$ (or $O(m \cdot \mathrm{nnz})$ for sparse gradients, where nnz is the number of nonzero entries per gradient), plus $O(d^3)$ for the eigen decomposition.

5 Fully Adaptive Facet Normals

Both the rotated-hyperrectangle construction of Section 4 and the ellipsoidal alternative of Appendix B.4 fix their generating directions in advance – an orthonormal basis U, estimated once and held fixed – so only per-axis radii adapt to the data, never the directions themselves. A more ambitious goal is to let a general polytope's facet normals adapt online directly, so that the shape responds to the data with no orthogonality or fixed-basis constraint at all. It is motivated by the sensitivity of the dual gauge.

5.1 Sensitivity of the dual gauge

Let $C = \{x : \langle a_i, x\rangle \le 1,\ i = 1,\ldots,m\}$ be a polytope with facet normals $a_1, \ldots, a_m$, so $\gamma_C(x) = \max_i \langle a_i, x\rangle$ and $C^\circ = \mathrm{conv}\{a_1, \ldots, a_m\}$, with

$$\gamma_{C^\circ}(g) = \min\Big\{ \textstyle\sum_i \mu_i : \mu \succeq 0,\ \sum_i \mu_i a_i = g \Big\}.$$

Let $\mu^\star(g)$ be an optimal solution of this linear program and $v^\star(g) \in \arg\max_{x \in C}\langle g, x\rangle$ an optimal solution of its dual, a vertex of C with $\langle g, v^\star\rangle = \gamma_{C^\circ}(g)$. A standard LP sensitivity (envelope) argument gives, for each active facet i (i.e. $\mu_i^\star > 0$),

$$\frac{\partial \gamma_{C^\circ}(g)}{\partial a_i} = -\mu_i^\star\, v^\star,$$

and 0 for inactive facets. Only the facets currently used to represent g in the polar receive a nonzero sensitivity – the polytopal analogue of AdaGrad's diagonal accumulator only updating coordinates with nonzero gradient.

Remark 6 *Because Theorem 1 guarantees* C° *is convex regardless of* C*'s shape, the facet↔vertex principle of Figure 1 remains sound even while Algorithm 3's online facet adaptation (Section 5) explores primal shapes that are momentarily irregular or non-convex.*

The design intuition (Remark 6) is: to make the dual penalty small in a gradient direction g, the primal facet should be *flat and close to the origin* in that direction, which by polarity produces a *sharp vertex* of C° pointing toward g. Conversely a sharp corner of C in a direction produces a flat, high-penalty face of C° perpendicular to it. Algorithm 3 operationalizes this by nudging only the facets actively representing each observed g_t, using $g_t/\|g_t\|$ as a cheap proxy for the exact sensitivity direction $v^\star$ (the two have positive inner product, since $\langle g_t, v^\star\rangle = \gamma_{C_t^\circ}(g_t) > 0$).

Algorithm 3 Adaptive Facet Normals

1: **Input:** step size $\eta_a > 0$, target facet scale D, blending weight $\beta \in (0, 1)$, initial facet normals $a_1^{(0)}, \ldots, a_m^{(0)}$ (e.g. $\pm e_i$).
2: Set $a_i \leftarrow a_i^{(0)}$ for all i.
3: **for** $t = 1, \ldots, T$ **do**
4: Observe g_t; solve the small LP $\gamma_{C_t^\circ}(g_t) = \min\{\sum_i \mu_i : \mu \succeq 0,\ \sum_i \mu_i a_i = g_t\}$ for $\mu^\star$ and its dual vertex $v^\star \in \arg\max_{x \in C_t} \langle g_t, x \rangle$.
5: $x_{t+1} \leftarrow x_t - \alpha_t\, v^\star / \|v^\star\|$, with α_t from an Armijo line search
6: **for** each active facet i (i.e. $\mu_i^\star > 0$) **do**
7: $a_i \leftarrow a_i + \eta_a\, \mu_i^\star\, g_t / \|g_t\|$ ▷ tilt the facet toward the gradient it represents
8: **end for**
9: Rescale $\{a_i\}$ uniformly so that their mean norm is D.
10: $a_i \leftarrow (1 - \beta)\, a_i + \beta\, \bar{a}_i^{(0)}$, where $\bar{a}_i^{(0)}$ is $a_i^{(0)}$ rescaled to mean norm D.
11: **end for**

Remark 7 *We present Algorithm 3 as a practical design tool, not as a method with a regret guarantee. Its facet update tilts each active facet toward the gradient it represents, so facets rotate as well as rescale, and the resulting sequence C_t need not be monotone. The final blending step keeps the facets close to the $\pm e_i$ configuration, which positively spans $\mathbb{R}^d$; this keeps C_t bounded and the LP of line 4 feasible.*

Remark 8 (Step size is a hindsight quantity) *The step size that optimizes the bound of Theorem 3, $\eta c^2 = D_\infty/\sqrt{2}$, depends on the diameter D_∞ of the trajectory, known only at the end of the horizon T. This is not a gap specific to our construction: AdaGrad's own step size (Duchi et al. (2011a)) has the identical structure – a global scalar η that sits outside the per-coordinate accumulator and is not set from a closed-form online prescription either, in the original paper or in any standard implementation. The same holds even for methods built specifically around adaptive stepsize theory: the recent universal AdaGrad-stepsize framework of Rodomanov et al. (2024) retains an analogous un-adapted global constant – the feasible-set diameter D – which their own analysis does not remove, despite adapting every other problem-dependent quantity automatically. Selecting this scalar without hindsight is the subject of a separate, actively studied line of work on parameter-free online optimization (e.g. Orabona (2019)), which we do not attempt to resolve here. Algorithms 1 and 3 instead take their step sizes as fixed input hyperparameters, tuned empirically (Section D.1), consistent with standard practice throughout this literature.*

6 Experiment

A Neural Network Training Task: We evaluate Algorithm 1 and the rotated-ellipsoidal variant of Appendix B.4 on a non-convex problem: 10-class classification of scikit-learn's `digits` images, against standard and current optimizers. Convex experiments are reported in Appendix D: an evaluation of basis estimation (Algorithm 2), sparse text classification, least-squares regression, and a non-stationary objective on which we evaluate the adaptive facet normals of Algorithm 3. We use a 2-layer multilayer perceptron ($64 \to 16 \to 10$, ReLU hidden layer, softmax cross-entropy loss, $d = 1{,}210$ parameters), trained for $1{,}000$ mini-batch steps of size 64. Every optimizer acts on the network's flattened parameter vector, as AdaGrad and its variants are used in practice.

The basis $\hat{U}$ is estimated by a direct analogue of Algorithm 2: we probe $m = 100$ isotropic Gaussian perturbations ($\sigma = 0.05$) of a freshly initialized parameter vector and take the eigenvectors of the empirical second-moment matrix of the resulting mini-batch gradients. Three methods share this basis. *Algorithm 1* uses the radii $\rho_{t,i} = c/(\delta + r_{t,i})$ and the steepest-descent step under the rotated hyperrectangle $C_t = C_{\hat{U},\rho_t}$ (Algorithm 1, line 9),

$$x_{t+1} \;=\; x_t - \eta\, \gamma_{C_t^\circ}(g_t)\, \hat{U}\big(\rho_t \odot \operatorname{sign}(\hat{U}^\top g_t)\big), \qquad \gamma_{C_t^\circ}(g_t) = \textstyle\sum_i \rho_{t,i}\, |\tilde{g}_{t,i}|, \tag{1}$$

which is the minimizer of $\langle g_t, d\rangle + \frac{1}{2\eta}\gamma_{C_t}(d)^2$ by Lemma 2. In this form η and c enter only through ηc^2, so we fix $c = 1$. The *rotated-ellipsoid variant* of Appendix B.4 ($\rho_{t,i} = c/\sqrt{\delta + r_{t,i}}$) takes

the step $x_{t+1} = x_t - \eta c^2 \hat{U} \left(\tilde{g}_t/(\delta + r_t)\right)$. As an ablation, the *sign-only* method takes $x_{t+1} = x_t - \eta \hat{U} \operatorname{sign}(\hat{U}^\top g_t)$, i.e. a direction with a fixed step and without the radii or the dual-gauge scaling of equation 1. We compare against Plain SGD, SGD with momentum (0.9), Diagonal AdaGrad, Adam, and Muon (Jordan et al., 2024), which orthogonalizes momentum updates for matrix-shaped parameters via a Newton–Schulz iteration, approximating steepest descent under the spectral norm – the closest practical analogue to our construction, for a different choice of shape. Muon is applied to the two weight matrices and Adam to the bias vectors. Full-matrix AdaGrad is omitted: its $O(d^3)$ per-step eigendecomposition makes 1,000 steps impractical at $d = 1{,}210$.

The data are split (stratified) into 1,149 training, 288 validation, and 360 test images. Every method is tuned by the same procedure: its learning rate is searched over $\{10^{-8}, 10^{-7}, \ldots, 10^{1}\}$ and then refined twice around the best value (factors $10^{\pm 1/2}$ and $10^{\pm 1/4}$), separately for each value of its second hyperparameter ($\delta \in \{10^{-8}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10\}$ for Diagonal AdaGrad and both rotated methods; $\epsilon \in \{10^{-8}, 10^{-6}, 10^{-4}\}$ for Adam; the bias learning rate $\in \{10^{-3}, 10^{-2}, 10^{-1}\}$ for Muon). The selection criterion is validation accuracy, averaged over the last three logged steps and over two tuning seeds that are disjoint from the evaluation seeds. The test set is used only in the final runs, over 5 evaluation seeds; a given seed fixes the initialization and the mini-batch sequence for every optimizer, so all comparisons are paired.

Method	Final test accuracy	Steps to 90%	Steps to 95%
Plain SGD	0.9628 ± 0.0057	105	275
SGD + momentum	0.9594 ± 0.0084	75	205
Diagonal AdaGrad	0.9617 ± 0.0064	80	315
Adam	0.9656 ± 0.0057	85	235
Muon	0.9611 ± 0.0018	190	280
Sign-only ablation	0.9639 ± 0.0063	130	280
Rotated-ellipsoid variant (App. B.4)	0.9650 ± 0.0057	**50**	145
Algorithm 1	0.9617 ± 0.0062	60	**140**

Table 1: Final test accuracy after 1,000 mini-batch steps (mean $\pm$ std over 5 seeds), and the number of steps needed to first reach a given test accuracy (mean over seeds; every method reached both thresholds on every seed; accuracy is logged every 25 steps). Step counts exclude the $m = 100$ gradient evaluations of the basis warm-up used by the last three rows.

Method	Step 50	Step 100	Step 250	Step 500	Step 1,000
Plain SGD	0.8539	0.9000	0.9444	0.9572	0.9628
SGD + momentum	0.8811	0.9250	0.9500	0.9572	0.9594
Diagonal AdaGrad	0.8800	0.9122	0.9389	0.9522	0.9617
Adam	0.8517	0.9128	0.9506	0.9611	0.9656
Muon	0.4811	0.7911	0.9417	0.9650	0.9611
Sign-only ablation	0.6628	0.8756	0.9450	0.9611	0.9639
Rotated-ellipsoid variant	**0.9194**	**0.9422**	**0.9583**	0.9611	0.9650
Algorithm 1	0.9061	0.9350	0.9556	0.9611	0.9617

Table 2: Test accuracy at fixed training checkpoints, mean over 5 seeds.

With every method tuned by the same protocol, final test accuracy does not separate the optimizers on this task (Table 1): all eight methods end between 0.9594 and 0.9656, a spread of about two test images, and no paired difference between either rotated method and any baseline is significant (paired t-test, $p \geq 0.16$). We therefore make no claim that Algorithm 1 reaches a better final model here; the task saturates at roughly 96% for every method.

The rotated methods differ from the baselines in how quickly they get there. Algorithm 1 and the rotated-ellipsoid variant need the fewest steps to reach both 90% (60 and 50 steps, vs. 75 for the fastest baseline, SGD with momentum) and 95% (140 and 145 steps, vs. 205). At step 50, both rotated methods are more accurate than every baseline on each of the 5 seeds individually, not only on average (Table 2). Algorithm 1 also ends with the lowest training loss of all methods except

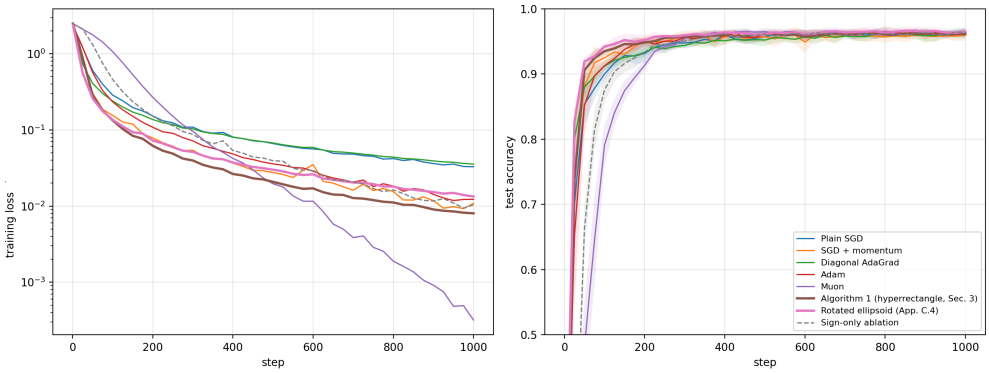


Figure 2: Training loss on the training set (left, log scale) and test accuracy (right) over training, mean over 5 seeds; the shaded band is $\pm$ one standard deviation.

Muon (Figure 2); Muon drives training loss more than an order of magnitude lower still, without a corresponding gain in test accuracy. Muon is also the slowest method early in training at its selected learning rate; since every method was tuned for final validation accuracy rather than speed, we do not read this as a general statement about Muon.

The sign-only ablation, which keeps the estimated basis but drops the radii ρ_t and the scalar $\gamma_{C_t^\circ}(g_t)$ of equation 1, is much slower early on (0.66 accuracy at step 50, vs. 0.91 for Algorithm 1), even though all three rotated methods reach similar final accuracy. The early advantage on this task therefore comes from the accumulated radii and the gauge-induced step, not from the rotation alone; on sparse text (Section D.2.2) no such benefit appears.

Two costs offset this advantage and must be counted. First, the basis warm-up uses $m = 100$ gradient evaluations; counting them, the rotated methods reach 95% after 240–245 gradient evaluations, comparable to Adam (235) and behind SGD with momentum (205). Second, a dense $\hat{U}$ makes each rotation $O(d^2)$: on our CPU, the optimizer update alone takes about 0.9 ms per step for the rotated methods, vs. 0.02 ms for Adam and 0.14 ms for Muon. The advantage we observe is thus in the number of optimizer iterations, not in gradient evaluations or wall-clock time at this scale.

We present this experiment as empirical evidence that the method's geometric mechanism transfers to a non-convex task, not as a claim covered by this paper's theory: the basis-estimation procedure used here is a direct analogue of Algorithm 2, not an instance of Proposition 1's guarantee, which assumes i.i.d. gradients with a fixed population second moment – an assumption a neural network's loss landscape does not satisfy globally. Neither Algorithm 1 nor Muon is covered by a regret guarantee in this paper; convergence-rate results for Muon-family methods are given by Pethick et al. (2025), and the rotated-ellipsoid variant is covered by Theorem 3 in the convex setting.

7 Conclusion

We asked what shape an adaptively-revised rule should take, formalized "shape" as the squared gauge of a convex absorbing set C_t, and proved a regret bound (Theorem 2) for any monotone sequence of such sets with strongly convex squared gauge, with AdaGrad's diagonal and full-matrix methods as ellipsoidal special cases. We showed that the natural generalization of AdaGrad's hindsight-design argument to arbitrary C is degenerate (Remark 3), that strong convexity – and with it the standard mirror-descent analysis – fails for every polytope (Remark 2), and gave a rotated-ellipsoidal instance, rotated diagonal AdaGrad with a data-driven basis, with an $O(\sqrt{T})$ guarantee (Theorem 3, Proposition 1). Empirically, rotated geometries converge markedly faster than axis-aligned baselines with a line search and a sufficiently long basis warm-up, and early in neural-network training; but they do not win uniformly. Reconciling flat facets, which make polytopes attractive, with the strong convexity that regret analysis requires is the main open problem this paper leaves.

REFERENCES

Charu C Aggarwal et al. *Neural networks and deep learning*, volume 10. springer Cham, 2018.

Kang An, Yuxing Liu, Rui Pan, Yi Ren, Shiqian Ma, Donald Goldfarb, and Tong Zhang. ASGO: Adaptive structured gradient optimization, 2025.

Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. In *Workshop on Optimization for Machine Learning (OPT)*, 2024. arXiv preprint arXiv:2409.20325.

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

K. Crammer, M. Dredze, and A. Kulesza. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 23*, 2009.

Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970. doi: 10.1137/0707001.

Frank Deutsch. *Best Approximation in Inner Product Spaces*, volume 7 of *CMS Books in Mathematics*. Springer, New York, 2001.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011a.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011b.

John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *Colt*, volume 10, pp. 14–26, 2010.

R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. `https://kellerjordan.github.io/posts/muon/`, 2024. Technical report / blog post; not peer-reviewed.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ken Lang. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, pp. 331–339. Elsevier, 1995.

Tim Tsz-Kit Lau, Qi Long, and Weijie Su. PolarGrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective, 2025.

Ning Liu and Benjamin Grimmer. Gauges and accelerated optimization over smooth and/or strongly convex sets. *arXiv preprint arXiv:2303.05037*, 2023.

Zhou Lu, Nataly Brukhim, Paula Gradu, and Elad Hazan. Projection-free adaptive regret with membership oracles. In *International Conference on Algorithmic Learning Theory*, pp. 1055–1073. PMLR, 2023.

Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49069–49104. PMLR, 2025.

Anton Rodomanov, Xiaowen Jiang, and Sebastian Stich. Universality of adagrad stepsizes for stochastic optimization: Inexact oracle, acceleration and variance reduction. *Advances in Neural Information Processing Systems*, 37:26770–26813, 2024.

N. Z. Shor. Utilization of the operation of space dilation in the minimization of convex functions. *Cybernetics and Systems Analysis*, 6(1):7–15, 1972.

Joel A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015. doi: 10.1561/2200000048.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2(3), 2008.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Shuo Xie, Tianhao Wang, Sashank Reddi, Sanjiv Kumar, and Zhiyuan Li. Structured preconditioners in adaptive optimization: A unified analysis. In *International Conference on Machine Learning*. PMLR, 2025a.

Shuo Xie, Tianhao Wang, Beining Wu, and Zhiyuan Li. A tale of two geometries: Adaptive optimizers and non-Euclidean descent. *arXiv preprint arXiv:2511.20584*, 2025b.

Yi Yu, Tengyao Wang, and Richard J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015. doi: 10.1093/biomet/asv008.

A Related work

Adapting a first-order method's local metric to the data traces back to variable-metric and space-dilation methods (Fletcher, 1970; Shor, 1972) and, in machine learning, to confidence-weighted and second-order online classifiers (Crammer et al., 2009; Cesa-Bianchi et al., 2005). Almost universally, *what shape* that metric should take has been answered the same way: a quadratic form, varying only in how its matrix is set. Duchi et al. (2011a)'s AdaGrad, the instance we engage with most directly, introduced diagonal and full-matrix proximal functions with regret bounds as good as the best fixed quadratic form in hindsight; both are ellipsoidal special cases of our general bound (Section 3), and we add the machinery and constructions for shapes that are not ellipsoids at all.

Three recent threads bear directly on this paper. Lu et al. (2023) extend a membership-oracle approach to *adaptive* regret (measured against every sub-interval, not just $[1, T]$) via a Minkowski regularization – the same gauge object as ours, applied to a different performance measure; extending our polytopal constructions to their adaptive-regret setting is a natural next step we do not pursue. Liu & Grimmer (2023) build projection-free methods from the Minkowski gauge as well, but for smooth/strongly-convex sets (curved boundaries, no corners) – the opposite regime from our polytopal, piecewise-linear constructions, though the same underlying mirror-map object; our Remark 2 explains why the strongly convex regime is the one in which regret analysis goes through. Xie et al. (2025a) give a unified analysis covering diagonal and full-matrix AdaGrad, AdaGrad-Norm, and Shampoo (Gupta et al., 2018), indexed by a matrix subalgebra – the closest project to ours in spirit, but their family always remains a subalgebra of quadratic forms; their finding that more structured, cheaper preconditioners can *outperform* less structured ones is an independent data point for the accuracy/cost trade-off we discuss in Section 4. Xie et al. (2025b) relate adaptive optimizers to *normalized steepest descent*, the family including Muon and Lion that reduces, in our language, to steepest descent under an ℓ_∞- or spectral-norm ball; this is the closest practical analogue to our axis-aligned construction, with our contribution being the *rotated* case.

A concurrent, spectral-norm answer to the same "what shape" question has emerged in practice. Vyas et al. (2025) propose SOAP, stabilizing Shampoo with Adam-style second-moment tracking; Pethick et al. (2025) propose Scion, unifying several optimizers under a single convergence guarantee via the linear minimization oracle of a norm ball – the closest existing theoretical analogue to Theorem 2, and one that, unlike strongly convex mirror descent, accommodates non-smooth norm balls. Both target matrix-shaped parameters via the spectral norm rather than a polytope, a perspective popularized by Bernstein & Newhouse (2024) and realized practically in Muon (Jordan et al., 2024), which orthogonalizes momentum updates and has seen wide adoption despite not yet being formally peer-reviewed; An et al. (2025) and Lau et al. (2025) propose related structured, matrix-aware preconditioners in the same spectral-norm spirit. Our rotated-hyperrectangle construction and this spectral-norm family answer the same question – what non-ellipsoidal shape an adaptive method's unit ball should take – for different parameter classes and via different mechanisms: facet-based accumulation versus matrix orthogonalization.

B Proofs

B.1 Proof of Lemma Scalar Accumulation Inequality

Lemma 1 *Let $b_1, \dots, b_T \geq 0$ and set $B_t := \big(\sum_{\tau=1}^{t} b_\tau^2\big)^{1/2}$ for $t = 0, 1, \dots, T$, with $B_0 := 0$ and the convention $0/0 := 0$. Then $\sum_{t=1}^{T} \frac{b_t^2}{B_t} \;\leq\; 2B_T$.*

Proof. By induction on T. If $b_1 = 0$ the $T = 1$ case reads $0 \leq 0$; otherwise $b_1^2/B_1 = b_1 \leq 2b_1 = 2B_1$, so the base case holds.

Assume the claim for $T-1$. If $b_T = 0$ then $B_T = B_{T-1}$ and

$$\sum_{t=1}^{T} \frac{b_t^2}{B_t} = \sum_{t=1}^{T-1} \frac{b_t^2}{B_t} \leq 2B_{T-1} = 2B_T,$$

so assume $b_T > 0$, hence $B_T > B_{T-1} \geq 0$. By the inductive hypothesis,

$$\sum_{t=1}^{T} \frac{b_t^2}{B_t} \leq 2B_{T-1} + \frac{b_T^2}{B_T},$$

so it suffices to show $2B_{T-1} + b_T^2/B_T \leq 2B_T$, i.e. $b_T^2 \leq 2B_T(B_T - B_{T-1})$. Since $B_T^2 - B_{T-1}^2 = b_T^2$ factors as $(B_T - B_{T-1})(B_T + B_{T-1})$, this is equivalent to

$$(B_T - B_{T-1})(B_T + B_{T-1}) \;\leq\; 2B_T(B_T - B_{T-1}),$$

and since $B_T - B_{T-1} > 0$ this reduces to $B_T + B_{T-1} \leq 2B_T$, i.e. $B_{T-1} \leq B_T$ – true because $B_T = \sqrt{B_{T-1}^2 + b_T^2} \geq B_{T-1}$. This completes the induction. □

B.2 Proof of Lemma: Gauge and polar of a rotated hyperrectangle

Lemma 2 *Let $U = [u_1, \dots, u_d]$ be orthonormal, $\rho_1, \dots, \rho_d > 0$, and $C_{U,\rho} = \{x \in R^d : |\langle u_i, x\rangle| \leq \rho_i,\ i = 1, \dots, d\}$. Writing $\tilde{x}_i := \langle u_i, x\rangle$, $\tilde{g}_i := \langle u_i, g\rangle$,*

$$\gamma_{C_{U,\rho}}(x) = \max_i \frac{|\tilde{x}_i|}{\rho_i}, \qquad C_{U,\rho}^{\circ} = \mathrm{conv}\{\pm u_i/\rho_i : i = 1, \dots, d\}, \qquad \gamma_{C_{U,\rho}^{\circ}}(g) = \sum_i \rho_i |\tilde{g}_i|.$$

Moreover, $\arg\min_d\{\langle g, d\rangle + \frac{1}{2\eta}\gamma_{C_{U,\rho}}(d)^2\} \ni -\eta\, \gamma_{C_{U,\rho}^{\circ}}(g)\, U(\rho \odot \mathrm{sign}(\tilde{g}))$.

Proof. Primal gauge. For $r > 0$, $x \in rC_{U,\rho}$ iff $x/r \in C_{U,\rho}$ iff $|\tilde{x}_i|/r \leq \rho_i$ for every i, iff $r \geq |\tilde{x}_i|/\rho_i$ for every i, iff $r \geq \max_i |\tilde{x}_i|/\rho_i$. Hence $\{r > 0 : x \in rC_{U,\rho}\} = [\max_i |\tilde{x}_i|/\rho_i, \infty)$ and $\gamma_{C_{U,\rho}}(x) = \max_i |\tilde{x}_i|/\rho_i$.

Dual gauge. By Section 2, $\gamma_{C^\circ_{U,\rho}}(g) = \sigma_{C_{U,\rho}}(g) = \sup_{x \in C_{U,\rho}} \langle g, x \rangle$. Since U is orthonormal, $x = \sum_i \tilde{x}_i u_i$ and $\langle g, x \rangle = \sum_i \tilde{x}_i \langle g, u_i \rangle = \sum_i \tilde{x}_i \tilde{g}_i$, so the objective separates coordinatewise over the box constraint $|\tilde{x}_i| \le \rho_i$: each term $\tilde{x}_i \tilde{g}_i$ is maximized independently at $\tilde{x}_i = \rho_i \operatorname{sign}(\tilde{g}_i)$, giving

$$\sigma_{C_{U,\rho}}(g) = \sum_i \rho_i \operatorname{sign}(\tilde{g}_i)\, \tilde{g}_i = \sum_i \rho_i |\tilde{g}_i|.$$

Polar set. Let $K := \operatorname{conv}\{\pm u_i/\rho_i : i = 1, \dots, d\}$. Since $\{u_i\}$ is an orthonormal basis, every $g \in R^d$ has the *unique* representation $g = \sum_i \mu_i (u_i/\rho_i)$ with $\mu_i = \rho_i \tilde{g}_i$. The gauge of a convex hull of a symmetric point set is $\gamma_K(g) = \inf\{\sum_i |\mu_i| : g = \sum_i \mu_i (u_i/\rho_i)\}$, and since the representation is unique here, the infimum is attained at the only candidate:

$$\gamma_K(g) = \sum_i |\mu_i| = \sum_i \rho_i |\tilde{g}_i| = \gamma_{C^\circ_{U,\rho}}(g),$$

using the dual-gauge computation above. Two closed, convex, absorbing sets with 0 in their interior and identical gauge functions coincide (a closed convex absorbing set A satisfies $A = \{x : \gamma_A(x) \le 1\}$, so γ_A determines A), hence $K = C^\circ_{U,\rho}$.

Steepest-descent step. Fix $s = \gamma_{C_{U,\rho}}(d) \ge 0$. Over $\{d : \gamma_{C_{U,\rho}}(d) = s\} = s\,\partial C_{U,\rho}$, $\langle g, d \rangle$ is minimized at $d = -s\,U(\rho \odot \operatorname{sign}(\tilde{g}))$ with value $-s\,\gamma_{C^\circ_{U,\rho}}(g)$, by the dual-gauge computation. Minimizing $-s\,\gamma_{C^\circ_{U,\rho}}(g) + s^2/(2\eta)$ over $s \ge 0$ gives $s = \eta\,\gamma_{C^\circ_{U,\rho}}(g)$. □

B.3 Proof of Theorem 2

Inequality $(*)$ is the standard telescoped composite-mirror-descent bound (see Duchi et al. (2011a), Corollary 3). Its derivation uses only convexity of each f_t, differentiability of ψ_t (so that B_{ψ_t} is defined and the optimality condition of (CMD) holds), and 1-strong convexity of ψ_t with respect to γ_{C_t}, which bounds $\eta\langle g_t, x_t - x_{t+1} \rangle - B_{\psi_t}(x_{t+1}, x_t)$ by $\frac{\eta^2}{2}\gamma^2_{C_t^\circ}(g_t)$; it therefore holds verbatim under the theorem's hypotheses. Both hypotheses fail for polytopal C_t (Remark 2).

B.4 The Rotated-Ellipsoidal Alternative

For radii $\rho = (\rho_1, \dots, \rho_d)$, $\rho_i > 0$, define the rotated ellipsoidal set

$$C^{\mathrm{ell}}_{U,\rho} = \Big\{ x \in \mathbb{R}^d : \sum_{i=1}^d \frac{\langle u_i, x \rangle^2}{\rho_i^2} \le 1 \Big\}$$

and the proximal function

$$\psi_t(x) = \frac{1}{2}\gamma_{C^{\mathrm{ell}}_{U,\rho_t}}(x)^2 = \frac{1}{2}\sum_{i=1}^d \frac{\langle u_i, x \rangle^2}{\rho_{t,i}^2}.$$

Lemma 3 (Ellipsoid gauge and dual gauge) *With $\tilde{x}_i = \langle u_i, x \rangle$, $\tilde{g}_i = \langle u_i, g \rangle$,*

$$\gamma_{C^{\mathrm{ell}}_{U,\rho}}(x) = \Big(\sum_i \tilde{x}_i^2/\rho_i^2 \Big)^{1/2}, \qquad \gamma_{(C^{\mathrm{ell}}_{U,\rho})^\circ}(g) = \Big(\sum_i \rho_i^2 \tilde{g}_i^2 \Big)^{1/2}.$$

Proof. Primal gauge. Since U is orthonormal, $x \in rC^{\mathrm{ell}}_{U,\rho}$ iff $\sum_i (\tilde{x}_i/r)^2/\rho_i^2 \le 1$ iff $r^2 \ge \sum_i \tilde{x}_i^2/\rho_i^2$ iff $r \ge (\sum_i \tilde{x}_i^2/\rho_i^2)^{1/2}$, giving the stated gauge directly.

Dual gauge. $\gamma_{(C^{\mathrm{ell}}_{U,\rho})^\circ}(g) = \sup_{x \in C^{\mathrm{ell}}_{U,\rho}} \langle g, x \rangle$. Writing $y_i := \tilde{x}_i/\rho_i$, the constraint set is $\sum_i y_i^2 \le 1$ and the objective is $\sum_i \tilde{x}_i \tilde{g}_i = \sum_i \rho_i y_i \tilde{g}_i$. By Cauchy–Schwarz, $\sum_i \rho_i y_i \tilde{g}_i \le (\sum_i \rho_i^2 \tilde{g}_i^2)^{1/2} (\sum_i y_i^2)^{1/2} \le (\sum_i \rho_i^2 \tilde{g}_i^2)^{1/2}$, with equality attained at $y_i = \rho_i \tilde{g}_i / (\sum_j \rho_j^2 \tilde{g}_j^2)^{1/2}$, which satisfies $\sum_i y_i^2 = 1$. □

Since $C^{\mathrm{ell}}_{U,\rho}$ is an ellipsoid for every ρ, ψ_t is an exact quadratic form, hence differentiable and 1-strongly convex with respect to its own gauge, with modulus independent of ρ; Theorem 2's hypotheses therefore hold for any monotone sequence ρ_t. We take

$$\rho_{t,i} = \frac{c}{\sqrt{\delta + r_{t,i}}}, \qquad \text{so that} \qquad \psi_t(x) = \frac{1}{2c^2} \sum_{i=1}^{d} (\delta + r_{t,i})\, \tilde{x}_i^2,$$

which is exactly diagonal AdaGrad's proximal function in the rotated coordinates $\tilde{x} = U^\top x$. Running Algorithm 1 with C_{U,ρ_t} replaced by $C^{\mathrm{ell}}_{U,\rho_t}$, line 7 by the radii above, and line 9 by the (CMD) step with this ψ_t gives, for $X = \mathbb{R}^d$ and $\phi = 0$, the update $x_{t+1} = x_t - \eta c^2\, U\big(\tilde{g}_t/(\delta + r_t)\big)$, and in general:

Theorem 3 (Regret bound for the rotated-ellipsoidal variant) *Let U be orthonormal and fixed, let $\{x_t\}$ be generated by the rotated-ellipsoidal variant above with $\delta \geq \max_t \|U^\top g_t\|_\infty$, and let $D_\infty := \max_{t \leq T} \|U^\top (x^* - x_t)\|_\infty$. Then*

$$R_\phi(T) \;\leq\; \frac{D_\infty^2}{2\eta c^2} \sum_{i=1}^{d} \|\tilde{g}_{1:T,i}\|_2 + \eta c^2 \sum_{i=1}^{d} \|\tilde{g}_{1:T,i}\|_2,$$

and the choice $\eta c^2 = D_\infty/\sqrt{2}$ gives $R_\phi(T) \leq \sqrt{2}\, D_\infty \sum_{i=1}^{d} \|\tilde{g}_{1:T,i}\|_2$. Since $\sum_i \|\tilde{g}_{1:T,i}\|_2 \leq \sqrt{d \sum_t \|g_t\|_2^2}$, this is $O(\sqrt{T})$ for bounded gradients.

Proof. Write $\tilde{x} = U^\top x$, $\tilde{X} = U^\top X$, $\tilde{\phi}(\tilde{x}) = \phi(U\tilde{x})$ and $\tilde{f}_t(\tilde{x}) = f_t(U\tilde{x})$. Since U is orthonormal, $\tilde{g}_t = U^\top g_t$ is a subgradient of $\tilde{f}_t$ at $\tilde{x}_t$, $\langle g_t, x\rangle = \langle \tilde{g}_t, \tilde{x}\rangle$, and the regret is unchanged by the change of variables. With $H_t = \delta I + \mathrm{diag}(r_t)$, we have $\psi_t(x) = \frac{1}{2c^2}\langle \tilde{x}, H_t \tilde{x}\rangle$, so the (CMD) objective equals $\frac{1}{c^2}\big[\eta c^2 \langle \tilde{g}_t, \tilde{x}\rangle + \eta c^2 \tilde{\phi}(\tilde{x}) + \frac{1}{2}\langle \tilde{x} - \tilde{x}_t, H_t(\tilde{x} - \tilde{x}_t)\rangle\big]$. Its minimizer is therefore exactly diagonal AdaGrad's composite-mirror-descent iterate in the coordinates $\tilde{x}$, with step size ηc^2 and accumulators $r_{t,i} = \|\tilde{g}_{1:t,i}\|_2$. The stated bound is Duchi et al. (2011a, Theorem 5) applied to $\{\tilde{f}_t\}$; its key step is Lemma 1. The choice of ηc^2 minimizes $a/(\eta c^2) + b\,\eta c^2$ with $a = \frac{1}{2} D_\infty^2 \sum_i \|\tilde{g}_{1:T,i}\|_2$ and $b = \sum_i \|\tilde{g}_{1:T,i}\|_2$. The final inequality is Cauchy–Schwarz over i. □

B.5 Proof of Proposition 1

Step 1 (concentration of $\widehat{M}_m$). Each summand $g_t g_t^\top - M$ is a mean-zero, symmetric $d \times d$ random matrix with operator norm bounded by $\|g_t g_t^\top\|_{\mathrm{op}} + \|M\|_{\mathrm{op}} \leq B^2 + \lambda_1 \leq 2B^2$ almost surely (since $\lambda_1 \leq \mathbb{E}\|g\|_2^2 \leq B^2$), and matrix variance $\big\|\sum_{t=1}^m \mathbb{E}[(g_t g_t^\top - M)^2]\big\|_{\mathrm{op}} \leq m B^4$. The matrix Bernstein inequality (Tropp, 2015, Theorem 1.6.2) then gives, for any $s > 0$,

$$\Pr\Big[\big\|\textstyle\sum_{t=1}^m (g_t g_t^\top - M)\big\|_{\mathrm{op}} \geq s\Big] \;\leq\; 2d \exp\Big(\frac{-s^2/2}{mB^4 + 2B^2 s/3}\Big).$$

Setting the right-hand side to δ and solving for s gives, with probability at least $1 - \delta$,

$$\|\widehat{M}_m - M\|_{\mathrm{op}} \;=\; \frac{1}{m}\Big\|\textstyle\sum_{t=1}^m (g_t g_t^\top - M)\Big\|_{\mathrm{op}} \;\leq\; C_1 B^2 \sqrt{\frac{\log(d/\delta)}{m}}$$

for a universal constant C_1, once $m \gtrsim \log(d/\delta)$ (so the linear-in-s term in the Bernstein bound is dominated by the $\sqrt{m}$ term).

Step 2 (Davis–Kahan). Since M has eigen-gap gap, the $\sin\Theta$ theorem (Davis & Kahan, 1970), in the per-eigenvector form of Yu et al. (2015, Corollary 3), gives

$$\sin \angle(\hat{u}_i, u_i) \;\leq\; \frac{2\,\|\widehat{M}_m - M\|_{\mathrm{op}}}{\mathrm{gap}}$$

for every i, provided $\|\widehat{M}_m - M\|_{\mathrm{op}} \leq \mathrm{gap}/2$ (guaranteed once m is large enough that the Step 1 bound is below $\mathrm{gap}/2$).

Combining. Substituting the Step 1 bound into Step 2 and taking a union bound over the d eigenvectors (absorbed into the $\log(d/\delta)$ term) gives the stated bound, with $C = 2C_1$.

C TOWARD A SHAPELESS CONSTRUCTION

Sections 4–5 answer the question of shape by fixing a family in advance – a rotated hyperrectangle (Algorithm 1) or a polytope with a fixed number of facets whose normals adapt online (Algorithm 3). Neither is forced to be an ellipsoid, but both still commit to a shape *before* seeing the data: a hyperrectangle in some basis, or a polytope with m facets. We close by describing a construction that commits to nothing – one where the shape of C_t is not chosen from a family at all, but is exactly the shape the observed subgradients themselves trace out, together with the obstacles that keep it from being a proof rather than a proposal.

C.1 A POLYTOPE GENERATED BY THE GRADIENTS THEMSELVES

Given subgradients $g_1, \dots, g_t$ and a sequence of positive scalars $\rho_1, \dots, \rho_t$, define C_t directly by its polar:

$$C_t^\circ := \operatorname{conv}\Big\{ \pm \frac{g_\tau}{\rho_\tau} : \tau = 1, \dots, t \Big\}, \qquad C_t := \big\{ x \in \mathbb{R}^d : \langle g_\tau, x \rangle \le \rho_\tau,\ \tau = 1, \dots, t \big\}.$$

Every facet of C_t is normal to an observed gradient; no basis, rotation, or fixed facet count is chosen in advance. C_t is convex regardless of how the g_τ arose (Theorem 1), and $C_{t+1} \subseteq C_t$ whenever ρ_τ is non-increasing in t – the same monotone-accumulator mechanism as Algorithm 1's radii, applied here to a shape with no fixed orientation at all. Since every facet of C_t is flat, by construction, ψ_t is not strongly convex, so this construction falls outside Theorem 2, like the hyperrectangle and the facet-adaptive method.

This construction nonetheless makes the gradient term of $(*)$ nearly free. Because g_t/ρ_t is by construction one of the generators of C_t°,

$$\gamma_{C_t^\circ}(g_t) \le \rho_t$$

holds immediately, with no argument beyond the definition of the gauge of a convex hull: the round's own gradient is automatically cheap to penalize under the polytope that round just built to contain it. Summing gives $\sum_{t \le T} \gamma_{C_t^\circ}(g_t)^2 \le \sum_{t \le T} \rho_t^2$, controlling exactly the gradient term of $(*)$ – the same role Algorithm 1's accumulator r_t plays, but here derived from the construction directly rather than imposed. Algorithm 1 is recovered as the special case where the generators are restricted to lie along a fixed orthonormal basis U rather than the raw gradients; Algorithm 3's online facet adaptation can be read as an approximate, practical construction at this same object, updating a fixed-size facet set toward the gradients it observes rather than generating facets from them directly.

Absorbency, needed for the gauge to be finite, can be weakened from absorption on all of $\mathbb{R}^d$ to absorption relative only to the feasible set X – $\gamma_{C_t}(x) < \infty$ for $x \in X$. Convexity, by contrast, is not an assumption to relax at all: it holds automatically for any set built as an intersection of halfspaces, however those halfspaces are chosen.

What remains open is what happens before $t \ge d$: while the observed gradients $g_1, \dots, g_t$ span a proper subspace of $\mathbb{R}^d$, C_t is an unbounded prism in the unspanned directions, and $\gamma_{C_t}(x) = \infty$ for any $x \in X$ with a component there. C_t is not absorbing relative to X during this burn-in period unless X itself is contained in the span of the gradients seen so far – a condition that need not hold, and that we do not know how to guarantee without reintroducing some fixed structure (e.g. a warm-up phase in the spirit of Algorithm 2, which is exactly the kind of externally-imposed choice this construction was meant to avoid). This is not a peripheral technicality: it is the same degeneracy, in a purer form, that facet-adaptation method exhibits when its facets lose positive span (Section D.2.4), there resolved only by an ad hoc regularization toward a fixed, guaranteed-absorbing reference polytope. Together with the strong-convexity obstruction, it is why a regret guarantee for this construction remains open. Whether a data-driven fix exists – one that lets the gradients themselves determine when and how to guard against these failures, rather than falling back on a fixed family the moment they threaten to – is, to our mind, the right version of the question this paper opened with: not which shape is best, but whether a method can be trusted to find out for itself, with nothing given in advance but what it has seen.

D ADDITIONAL EXPERIMENTS

This appendix reports the convex experiments, complementing the neural-network experiment of Section 6. We ask four questions. First, how well does Algorithm 2 recover the true basis, and how much of an oracle rotation's benefit does the estimated basis deliver (Section D.2.1)? Second, where do Algorithm 1 and the rotated ellipsoid of Appendix B.4 sit on the accuracy/cost frontier on high-dimensional sparse data (Section D.2.2)? Third, does the advantage of a rotated geometry on real data depend on a line search (Section D.2.3)? Fourth, does Algorithm 3's continuous online adaptation offer any advantage over a fixed, once-estimated basis when the problem itself is non-stationary (Section D.2.4)?

Methods and baselines. Our methods are Algorithm 1 and the rotated ellipsoid of Appendix B.4 (Sections D.2.2 and D.2.3); Algorithm 1-LS, its line-search variant (Sections D.2.1, D.2.3 and D.2.4); and Algorithm 3 (Section D.2.4). In every experiment, the basis $\hat{U}$ is estimated by Algorithm 2 (in Section D.2.2, by a variant that collects the warm-up gradients along a short SGD trajectory), and Section D.2.1 also evaluates Algorithm 2 in isolation. Where a true rotation exists by construction, the oracle basis $U = Q$, fixed throughout, serves as a reference. Baselines are Plain SGD (or GD), and Diagonal and Full-matrix AdaGrad (Duchi et al., 2011a); Algorithm 3 is compared directly against the fixed-basis Algorithm 1-LS under a distribution shift (Section D.2.4). All methods share the same loss sequence and initialization within a run.

D.1 PROTOCOL

Every reported number is a mean $\pm$ one standard deviation over 10 random seeds, unless stated otherwise. In Section D.2.2 and the online comparison of Section D.2.3, every method's step size (and its second hyperparameter, where it has one) is selected on held-out data or tuning seeds disjoint from the evaluation seeds, and then held fixed. Sections D.2.1 and D.2.3 otherwise use a shared line search instead of tuned step sizes, as described there.

D.2 RESULTS

D.2.1 EVALUATING BASIS ESTIMATION (ALGORITHM 2) ON A ROTATED QUADRATIC

This experiment isolates the effect of the *shape* of the unit ball, and tests whether a basis $\hat{U}$ estimated via Algorithm 2 still delivers most of the advantage an oracle rotation would provide. Then, additional experiments in the following of this section assess Algorithms 1 and 3.

We minimize $f(x) = \frac{1}{2}x^\top Ax$ over $x \in \mathbb{R}^d$, $d = 30$, where $A = Q\,\mathrm{diag}(\lambda_1, \ldots, \lambda_d)\,Q^\top$ for a uniformly random rotation Q[1] and curvatures λ_i geometrically spaced from 1 to $\kappa = 200$, so κ is exactly A's condition number; crucially, A's eigenvectors are *not* the coordinate axes, so the problem's curvature is rotated away from any fixed basis. The minimizer is $x^\star = 0$; the start x_1 is a fixed random unit vector. At each iteration we take the *exact* steepest-descent step $d_t^\star = \arg\min_{\|d\|_C \le 1} \langle g_t, d\rangle$ under the given unit ball C, with step size α_t set by a shared Armijo line search[2], so that the shape of C is the only thing that differs between runs. For symmetric positive-definite M, the exact steepest-descent direction under the ellipsoid $\{d : d^\top Md \le 1\}$ is $d^\star \propto -M^{-1}g_t$; under a hyperrectangle with (rotated) axes U, it is $d^\star = -U\,\mathrm{sign}(U^\top g_t)$ (Section 4). We write *Algorithm 1-LS* for this direction combined with the shared line search: it keeps Algorithm 1's rotated-hyperrectangle geometry and basis U, but replaces its radius schedule and step with the line search. We compare Plain GD, both AdaGrad directions, and Algorithm 1-LS under both a fixed oracle rotation ($U = Q$, given its geometry directly as a best-case reference) and a basis $\hat{U}$ estimated from an m-step isotropic warm-up via Algorithm 2, swept over $m \in \{0, d/2, d, 2d, 5d\}$. All methods share the same starting point, problem instance, and line-search protocol, so any difference in performance is attributable only to the choice of preconditioning geometry. We report, over the same 10 random seeds throughout: (i) iterations to reach $f(x_t) - f^\star < 10^{-3}$, (ii) total wall-clock time to reach that target, including any

[1] Sampled via QR decomposition of a $d \times d$ Gaussian random matrix, the standard construction for a Haar-uniform orthogonal matrix.

[2] Backtracking from a unit trial step by factors of 0.5; sufficient-decrease constant 10^{-4}.

one-time warm-up cost, (iii) suboptimality at two fixed iteration counts, and (iv) Algorithm 2's own cost and basis-recovery quality in isolation.

Method	Iterations to target
Plain GD	208.4 ± 70.8
Diagonal AdaGrad (line search)	164.5 ± 45.4
Full-matrix AdaGrad (line search)	84.4 ± 10.9
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = 0$ $(= I)$	164.3 ± 22.1
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = d/2$	96.9 ± 47.9
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = d$	63.7 ± 50.7
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = 2d$	20.1 ± 8.6
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = 5d$	12.9 ± 0.8
Alg. 1-LS, oracle $U = Q$ (fixed; no Algorithm 2)	12.6 ± 2.4

Table 3: Iterations to reach $f(x_t) - f^\star < 10^{-3}$, mean $\pm$ std over 10 seeds. All rows use the same Armijo line search.

Method	Total wall-clock time to target (ms)
Diagonal AdaGrad (line search)	3.541 ± 1.047
Full-matrix AdaGrad (line search)	10.856 ± 1.469
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = d$	3.099 ± 2.508
Alg. 1-LS, $\hat{U}$ via Algorithm 2, $m = 2d$	1.173 ± 0.324

Table 4: Total wall-clock time to reach $f(x_t) - f^\star < 10^{-3}$, including the one-time Algorithm 2 warm-up cost, mean $\pm$ std over the same 10 seeds as Table 3.

Method	Suboptimality at $t = 30$	Suboptimality at $t = 219$
Plain GD	$1.061 \times 10^{-1} \pm 6.79 \times 10^{-2}$	$6.135 \times 10^{-4} \pm 7.82 \times 10^{-4}$
Diagonal AdaGrad (line search)	$1.049 \times 10^{-1} \pm 4.87 \times 10^{-2}$	$3.909 \times 10^{-4} \pm 2.78 \times 10^{-4}$
Full-matrix AdaGrad (line search)	$8.346 \times 10^{-2} \pm 3.46 \times 10^{-2}$	$1.706 \times 10^{-7} \pm 3.77 \times 10^{-7}$
Alg. 1-LS, est. $\hat{U}$ (Alg. 2, $m = d$)	$4.223 \times 10^{-3} \pm 4.24 \times 10^{-3}$	$7.882 \times 10^{-5} \pm 1.52 \times 10^{-4}$
Alg. 1-LS, oracle $U = Q$ (fixed)	$5.255 \times 10^{-9} \pm 1.55 \times 10^{-8}$	$1.494 \times 10^{-34} \pm 5.79 \times 10^{-35}$

Table 5: Suboptimality $f(x_t) - f^\star$ at two fixed iteration counts, mean $\pm$ std over the same 10 seeds as Tables 3 and 4.

The rotated geometry helps only once the basis is estimated well enough. With a warm-up of $m = 2d$ gradient probes, Algorithm 1-LS needs about $8\times$ fewer iterations than the Diagonal AdaGrad direction (20.1 vs. 164.5) and about $4\times$ fewer than the Full-matrix AdaGrad direction (84.4, Table 3), and reaches the target about $3\times$ and $9\times$ faster in wall-clock time, including the warm-up (Table 4). At $m = 5d$ it essentially matches the oracle rotation (12.9 vs. 12.6 iterations). With a shorter warm-up the gain is small and unreliable: at $m = d$, Algorithm 1-LS needs 63.7 ± 50.7 iterations, only modestly fewer than Full-matrix AdaGrad, with very high variance across seeds, and at $m = 0$ $(U = I)$ it is no better than the Diagonal AdaGrad direction.

The suboptimality at fixed iteration counts tells the same story (Table 5, Figure 3). With the oracle rotation, Algorithm 1-LS reaches the limit of floating-point precision within about 90 iterations. With the $m = d$ estimate, it is well ahead of every baseline at iteration 30 (4.2×10^{-3}, vs. 1.05×10^{-1} for Diagonal AdaGrad and 8.35×10^{-2} for Full-matrix AdaGrad), but it then slows down, and by iteration 219 Full-matrix AdaGrad is more accurate (1.71×10^{-7} vs. 7.88×10^{-5}). An inaccurate basis leaves residual curvature that the sign direction cannot correct.

Warm-up length m	Time (ms)	Mean principal angle (deg)
0 ($= I$, no estimation)	0.0046 ± 0.0017	81.82 ± 1.34
15 ($d/2$)	0.2243 ± 0.0577	71.04 ± 2.89
30 (d)	0.2841 ± 0.0196	60.50 ± 4.44
60 ($2d$)	0.4697 ± 0.0861	45.05 ± 2.42
150 ($5d$)	0.9378 ± 0.1417	26.06 ± 3.33

Table 6: Algorithm 2 in isolation: one-time wall-clock cost and basis-recovery quality (mean principal angle to the true rotation Q, matched eigenvector-by-eigenvector as in Proposition 1), as a function of warm-up length m, mean $\pm$ std over the same 10 seeds used throughout this section. A principal angle of 0° indicates exact recovery; 90° indicates an uninformative estimate.

Table 6 isolates Algorithm 2 itself. Basis-recovery quality improves monotonically with warm-up length, from 81.82° at $m = 0$ (an essentially uninformative angle, since a fixed basis has no relationship to a random rotation Q) down to 26.06° at $m = 5d$, consistent with the recovery guarantee of Proposition 1, and this improvement tracks the iteration counts of Table 3. The one-time cost is small in absolute terms: even the largest warm-up tested ($m = 5d = 150$) costs under 1 ms. This advantage relies on the shared line search: on a real least-squares problem, the ordering reverses when Algorithm 1 is run online without one (Section D.2.3).

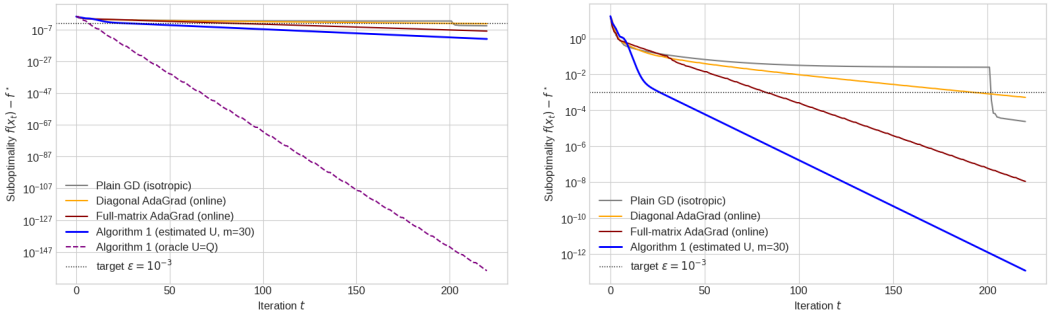


Figure 3: Suboptimality $f(x_t) - f^\star$ vs. iteration t (log scale) for a single representative seed, $d = 30$, $\kappa = 200$. All methods use the shared line search; Algorithm 1-LS's estimated basis uses $m = d$. Only the dashed curve (oracle $U = Q$) is given privileged information, shown as a reference. The horizontal dotted line marks the target $\epsilon = 10^{-3}$ used in Table 3. Panel (b) repeats panel (a) without the oracle-U curve, to show the differences between the other methods.

D.2.2 Real Sparse Text Classification

We evaluate the methods on 20 Newsgroups (Lang, 1995) (one class vs. rest, $d = 2^{18} \approx 262{,}000$ after feature hashing). Plain SGD and Diagonal AdaGrad operate in the full sparse feature space. The rank-256 methods operate on data projected once onto the top 256 singular vectors of 600 gradients collected along a short SGD warm-up, a variant of Algorithm 2. We report held-out log-loss and wall-clock cost per update.

Diagonal AdaGrad, in the full feature space, attains the best held-out log-loss. The four rank-256 methods all reach 0.163–0.165. In particular, the sign-only ablation matches Algorithm 1, so the benefit of the radii and dual-gauge scaling seen in Section 6 does not appear here; the gap to Diagonal AdaGrad reflects the rank-256 restriction rather than the choice of shape. Per update, Algorithm 1 is about $41\times$ cheaper than Diagonal AdaGrad and $170\times$ cheaper than rank-256 full-matrix AdaGrad. This saving does not repay the one-time basis estimation and projection (89.5 s) within the 2,500 steps run here: break-even against Diagonal AdaGrad is about 2×10^4 steps. Even plain SGD in the full feature space (0.1457) beats every rank-256 method, which confirms that the projection, not the shape, limits their accuracy. Table 7 therefore shows a cost–accuracy trade-off, not an advantage for Algorithm 1.

Method	Held-out log-loss	μs/update
Plain SGD	0.1457 ± 0.0018	3479.5 ± 495.9
Diagonal AdaGrad	$\mathbf{0.1354 \pm 0.0007}$	4504.9 ± 440.6
Algorithm 1 (rank-256)	0.1651 ± 0.0025	109.3 ± 19.5
Rotated ellipsoid, App. B.4 (rank-256)	0.1627 ± 0.0005	74.9 ± 7.5
Sign-only ablation (rank-256)	0.1646 ± 0.0011	79.1 ± 6.8
Full-matrix AdaGrad (rank-256)	0.1638 ± 0.0005	18664.7 ± 696.9

Table 7: 20 Newsgroups (one-vs-rest): held-out log-loss after 2,500 mini-batch steps (batch 64), mean $\pm$ std over 10 seeds; learning rate and δ tuned on a validation split. μs/update is wall-clock per step, including the mini-batch gradient. Rank-256 methods act on data projected once onto a basis estimated from 600 warm-up gradients; this one-time cost (89.5 s) is not included in μs/update. Plain SGD's selected learning rate lies at the top of the search grid, so its row may understate it.

D.2.3 Real Least-Squares Regression (Diabetes Dataset)

We next evaluate rotated geometries on a real, moderately ill-conditioned least-squares problem: the scikit-learn diabetes dataset ($n = 442$ samples, $d = 10$ features after standardization). The objective is the standard quadratic loss

$$f(x) = \frac{1}{2}\mathbb{E}\big[(x^\top a - b)^2\big],$$

and we measure the number of iterations required to reach a fixed sub-optimality target of 10^{-4} together with wall-clock time. All methods use the same Armijo line search, which here also tries enlarging the unit trial step by factors of 1.7 while the objective keeps decreasing; the rotated rows are Algorithm 1-LS (Section D.2.1). We report both an oracle basis (eigenvectors of the empirical Gram matrix) and bases estimated by Algorithm 2 with varying warm-up lengths.

Method	Iters to target	Wall-clock (ms)
Plain GD (normalized)	444.1 ± 99.2	314.42
Diag. AdaGrad direction	265.0 ± 54.2	93.87
Full AdaGrad direction	33.1 ± 6.5	27.85
Alg. 1-LS (oracle U, fixed)	10.9 ± 3.8	6.27
Alg. 1-LS (est. $\hat{U}$, $m = 0$)	308.4 ± 45.4	243.90
Alg. 1-LS (est. $\hat{U}$, $m = 10$)	184.8 ± 142.7	160.27
Alg. 1-LS (est. $\hat{U}$, $m = 20$)	12.4 ± 5.3	7.59
Alg. 1-LS (est. $\hat{U}$, $m = 40$)	12.1 ± 7.1	6.42

Table 8: Iterations to reach sub-optimality 10^{-4} and wall-clock time on the diabetes least-squares problem (iterations: mean $\pm$ std over 10 seeds; wall-clock: mean). Every row uses the same Armijo line search; wall-clock excludes the one-time Algorithm 2 warm-up.

Algorithm 1-LS reaches the target fastest with an oracle basis, and with only $m = 20$ probes of Algorithm 2 essentially matches it (12.4 vs. 10.9 iterations), requiring fewer iterations than the Full-matrix AdaGrad direction. The advantage depends on the line search. In a single-pass online version of the same problem (442 rounds, squared loss, no line search, step size and δ tuned on seeds disjoint from the 10 evaluation seeds), Algorithm 1 incurs regret 18.0 ± 2.5, versus 11.9–12.3 for plain SGD, diagonal and full-matrix AdaGrad, and the rotated ellipsoid of Appendix B.4. We therefore attribute the gains in Table 8 to line-search access, not to the quadratic loss alone.

D.2.4 Adaptive Facet Normals Under a Non-Stationary Objective

Algorithm 1-LS's basis $\hat{U}$ is estimated once and held fixed; Algorithm 3 instead adapts its facet normals continuously online, with no separate warm-up phase. Both take the same kind of step: the steepest-descent vertex of their current polytope, with an Armijo line search. This experiment isolates the setting where that distinction should matter: a rotated quadratic $f(x) = \frac{1}{2}(x-c)^\top A(x-$

c) whose rotation *and* minimizer change partway through the run – the same deterministic rotated-quadratic construction as in Section D.2.1. We use $d = 16$, condition number $\kappa = 50$, $T = 3{,}000$, switching at $t = 1{,}500$ to a new random rotation Q_2 and a relocated minimizer c_2 (sampled once, $\|c_2\| \approx 2\sqrt{d}$). Algorithm 1-LS's basis is estimated via Algorithm 2 ($m = d$) on phase 1 only and held fixed throughout; it is not re-estimated at the switch.

Algorithm 3's per-round facet update (Section 5) can, without safeguards, drive the facet normals toward near-degeneracy over many rounds – in an early implementation this caused the polytope to stop positively spanning $\mathbb{R}^d$, producing an unbounded dual-gauge LP on 2 of 10 seeds. We add a small regularization, blending the facets 2% per round back toward their original axis-aligned configuration (which positively spans $\mathbb{R}^d$ by construction), fully resolving this failure mode across all 10 seeds while leaving the facets free to adapt substantially the rest of the time.

Rounds after switch	Alg. 1-LS ($\hat{U}$ from Alg. 2, fixed)	Alg. 3 (adaptive)
10	$\mathbf{6.67 \pm 3.60}$	126.2 ± 117.5
50	$\mathbf{3.83 \times 10^{-2} \pm 3.07 \times 10^{-2}}$	10.78 ± 18.98
200	$\mathbf{4.00 \times 10^{-8} \pm 6.72 \times 10^{-8}}$	$1.53 \times 10^{-6} \pm 4.48 \times 10^{-6}$
500	$4.60 \times 10^{-18} \pm 1.38 \times 10^{-17}$	$\mathbf{1.35 \times 10^{-20} \pm 2.07 \times 10^{-20}}$
1000	$8.13 \times 10^{-30} \pm 7.71 \times 10^{-30}$	$1.77 \times 10^{-29} \pm 4.85 \times 10^{-30}$
1499	$8.13 \times 10^{-30} \pm 7.71 \times 10^{-30}$	$1.77 \times 10^{-29} \pm 4.85 \times 10^{-30}$

Table 9: Suboptimality after a switch in rotation and minimizer at $t = 1{,}500$, mean $\pm$ std over 10 seeds, $d = 16$, $\kappa = 50$.

Continuous adaptation gives no practical advantage here. Immediately after the switch, the fixed-basis method is far ahead (rounds 10–200): although its basis was estimated for phase 1's geometry, the stale rotation combined with the line search still recovers quickly, while Algorithm 3 must first relearn its facets. Algorithm 3 is ahead only at round 500, and by round 1,000 both methods have reached the limit of floating-point precision. In this setting, then, a once-estimated rotated basis is enough, and the extra cost of an LP per round buys nothing. Neither method is covered by Theorem 2 (Remark 2), and this experiment compares the two with each other, not with AdaGrad.

E Computational Complexity Summary

Table 10 collects the per-update time and space complexity claims stated throughout the paper (Section 4.1, Remark 5, Section 4.3) in one place, together with whether each method uses a rotated geometry. This is per-step cost only, not total cost to reach a target accuracy: none of the regret bounds in this paper translate directly into an iteration count without further assumptions, so the closest empirical measures of total complexity (cost per step times steps to target) are the wall-clock comparisons of Table 4 and Section D.2.2. Costs are per mini-batch update, in the d-dimensional parameter space, unless otherwise noted.

Table 11 separately reports the one-time cost of Algorithm 2, which is paid once before training rather than per update.

F Use of Large Language Models

We used a large language model (LLM) as an assistant during this work: to polish writing and draft parts of the text.

Method	Time/step	Space	Rotation
Plain SGD / GD	$O(d)$	$O(d)$	No
Diagonal AdaGrad	$O(d)$	$O(d)$	No
Full-matrix AdaGrad	$O(d^3)$	$O(d^2)$	Implicit (full matrix)
Algorithm 1 / rotated ellipsoid, $U = I$	$O(d)$	$O(d)$	No
Algorithm 1 / rotated ellipsoid, rank-k U	$O(dk)$	$O(dk)$	Yes (U)
Algorithm 1 / rotated ellipsoid, dense U	$O(d^2)$	$O(d^2)$	Yes (U)
Algorithm 3	not characterized†	$O(md)$	Yes (facet normals)
Muon	not characterized†	$O(d)$	Yes (spectral)

Table 10: Per-update cost and use of rotation for every method compared in the paper. Full-matrix AdaGrad's cost is that of a $d \times d$ matrix square root per step. †Algorithm 3's per-round cost is dominated by an m-variable LP solve (Section 5); Muon's cost is dominated by a Newton–Schulz orthogonalization of matrix-shaped parameters.

Method	One-time cost	Notes
Algorithm 2 (dense gradients)	$O(md^2)$	m warm-up probes
Algorithm 2 (sparse gradients)	$O(m \cdot \mathrm{nnz})$	nnz = nonzeros per gradient
+ eigendecomposition	$O(d^3)$	sorted by eigenvalue

Table 11: One-time warm-up cost of basis estimation (Algorithm 2), paid once before training begins; see Table 6 and Section D.2.2 for measured wall-clock values.