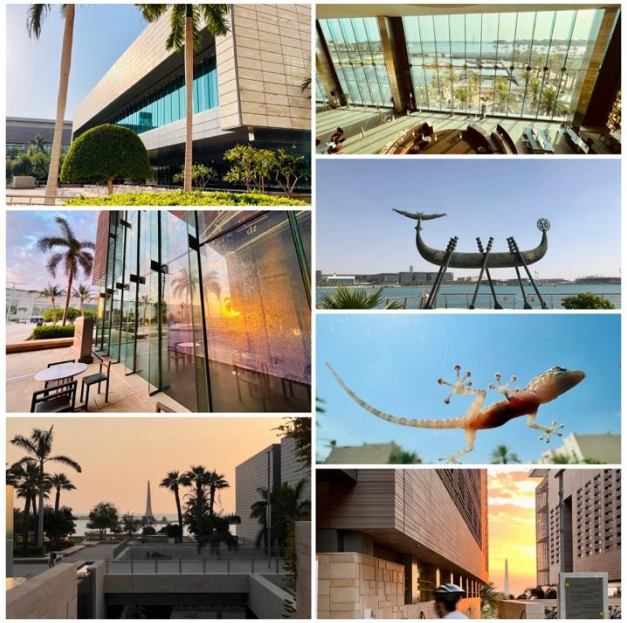
From the Broximal Point Method to Efficient Training of LLMs

Peter Richtárik
King Abdullah University of Science and Technology
Kingdom of Saudi Arabia

The 9th International Conference on Dynamics of Information Systems (DIS 2026)
Ústí nad Labem, Czech Republic
May 31-June 4, 2026

Optimization & Machine Learning Lab @ KAUST



Outline

Part 1: Broximal Point Method (BPM)

Part 2: From Adam to Muon: Orthogonalizing the Gradient

Part 3: Better Theory for Muon

Part 4: Muon: Compression and Error Feedback

Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Yin Tat Lee, Aaron Sidford, Kevin Tian
Acceleration with a ball optimization oracle
arXiv:2003.08078, 2020

Kaja Gruntkowska, Hanmin Li, Aadi Rane, P.R.
The ball-proximal (="broximal") point method: a new algorithm, convergence theory, and applications
arXiv:2502.02002, 2025

Kaja Gruntkowska, P.R.
Non-Euclidean broximal point method: a blueprint for geometry-aware optimization
arXiv:2510.00823, 2025

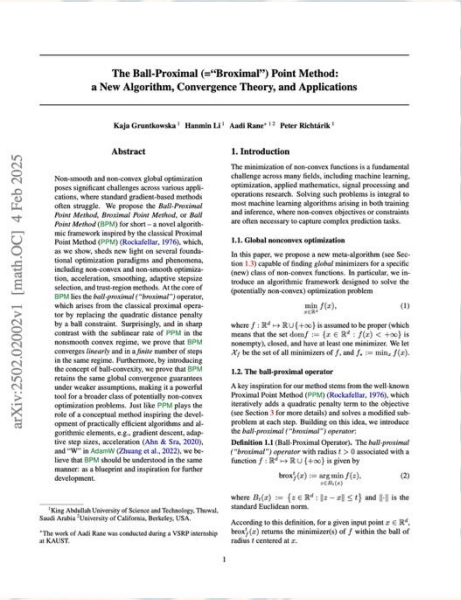
Kaja Gruntkowska, Hanmin Li, Xun Qian, P.R.
Broximal alignment for global non-convex optimization
Submitted (soon on arXiv)

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, P.R.
Gluon: Making Muon & Scion great again! (bridging theory and practice of LMO-based optimizers for LLMs)
arXiv:2505.13416, 2025

Xun Qian, Hussein Rammal, Dmitry Kovalev, P.R.
Muon is provably faster with momentum variance reduction
arXiv:2512.16598, 2025

Kaja Gruntkowska, Yassine Maziane, Zheng Qu, P.R.
Drop-Muon: Update less, converge faster
arXiv:2510.02239, 2025

Kaja Gruntkowska, Alexander Gaponov, Zhirayr Tovmasyan, P.R.
Error feedback for Muon and friends
ICLR 2026



The Ball-Proximal (="Broximal") Point Method:
a New Algorithm, Convergence Theory, and Applications

Kaja Gruntkowska[1] Hanmin Li[1] Aadi Rane*[1,2] Peter Richtárik[1]

Abstract

Non-smooth and non-convex global optimization poses significant challenges across various applications, where standard gradient-based methods often struggle. We propose the *Ball-Proximal Point Method*, *Broximal Point Method*, or *Ball Point Method* (BPM) for short – a novel algorithmic framework inspired by the classical Proximal Point Method (PPM) (Rockafellar, 1976), which, as we show, sheds new light on several foundational optimization paradigms and phenomena, including non-convex and non-smooth optimization, acceleration, smoothing, adaptive stepsize selection, and trust-region methods. At the core of BPM lies the *ball-proximal ("broximal") operator*, which arises from the classical proximal operator by replacing the quadratic distance penalty by a ball constraint. Surprisingly, and in sharp contrast with the sublinear rate of PPM in the nonsmooth convex regime, we prove that BPM converges *linearly* and in a *finite* number of steps in the same regime. Furthermore, by introducing the concept of ball-convexity, we prove that BPM retains the same global convergence guarantees under weaker assumptions, making it a powerful tool for a broader class of potentially non-convex optimization problems. Just like PPM plays the role of a conceptual method inspiring the development of practically efficient algorithms and algorithmic elements, e.g., gradient descent, adaptive step sizes, acceleration (Ahn & Sra, 2020), and "W" in AdamW (Zhuang et al., 2022), we believe that BPM should be understood in the same manner: as a blueprint and inspiration for further development.

[1]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia [2]University of California, Berkeley, USA.

*The work of Aadi Rane was conducted during a VSRP internship at KAUST.

1. Introduction

The minimization of non-convex functions is a fundamental challenge across many fields, including machine learning, optimization, applied mathematics, signal processing and operations research. Solving such problems is integral to most machine learning algorithms arising in both training and inference, where non-convex objectives or constraints are often necessary to capture complex prediction tasks.

1.1. Global nonconvex optimization

In this paper, we propose a new meta-algorithm (see Section 1.3) capable of finding *global* minimizers for a specific (new) class of non-convex functions. In particular, we introduce an algorithmic framework designed to solve the (potentially non-convex) optimization problem

$$\min_{x \in \mathbb{R}^d} f(x), \tag{1}$$

where $f : \mathbb{R}^d \mapsto \mathbb{R} \cup \{+\infty\}$ is assumed to be proper (which means that the set $\mathrm{dom} f := \{x \in \mathbb{R}^d : f(x) < +\infty\}$ is nonempty), closed, and have at least one minimizer. We let $\mathcal{X}_f$ be the set of all minimizers of f, and $f_* := \min_x f(x)$.

1.2. The ball-proximal operator

A key inspiration for our method stems from the well-known Proximal Point Method (PPM) (Rockafellar, 1976), which iteratively adds a quadratic penalty term to the objective (see Section 3 for more details) and solves a modified subproblem at each step. Building on this idea, we introduce the *ball-proximal ("broximal") operator*:

Definition 1.1 (Ball-Proximal Operator)**.** The *ball-proximal ("broximal") operator* with radius $t > 0$ associated with a function $f : \mathbb{R}^d \mapsto \mathbb{R} \cup \{+\infty\}$ is given by

$$\mathrm{brox}_t^f(x) := \arg\min_{z \in B_t(x)} f(z), \tag{2}$$

where $B_t(x) := \{z \in \mathbb{R}^d : \|z - x\| \leq t\}$ and $\|\cdot\|$ is the standard Euclidean norm.

According to this definition, for a given input point $x \in \mathbb{R}^d$, $\mathrm{brox}_t^f(x)$ returns the minimizer(s) of f within the ball of radius t centered at x.

1

Part 1
Broximal Point Method

Kaja Gruntkowska, Hanmin Li, Aadi Rane, P.R.
The ball-proximal (="broximal") point method: a new algorithm, convergence theory, and applications
arXiv:2502.02002, 2025

Kaja Gruntkowska, P.R.
Non-Euclidean broximal point method: a blueprint for geometry-aware optimization
arXiv:2510.00823, 2025

Problem Formulation

$$\min_{x \in \mathbb{R}^d} f(x)$$

f is proper, closed & convex

$\operatorname{Argmin} f \neq \emptyset$

$x_0 \in \operatorname{dom} f$

The Ball-Proximal (="Broximal") Point Method: a New Algorithm, Convergence Theory, and Applications

Kaja Gruntkowska [1] Hanmin Li [1] Aadi Rane* [1 2] Peter Richtárik [1]

Abstract

Non-smooth and non-convex global optimization poses significant challenges across various applications, where standard gradient-based methods often struggle. We propose the *Ball-Proximal Point Method*, *Broximal Point Method*, or *Ball Point Method* (BPM) for short – a novel algorithmic framework inspired by the classical Proximal Point Method (PPM) (Rockafellar, 1976), which, as we show, sheds new light on several foundational optimization paradigms and phenomena, including non-convex and non-smooth optimization, acceleration, smoothing, adaptive stepsize selection, and trust-region methods. At the core of BPM lies the *ball-proximal ("broximal")* operator, which arises from the classical proximal operator by replacing the quadratic distance penalty by a ball constraint. Surprisingly, and in sharp contrast with the sublinear rate of PPM in the nonsmooth convex regime, we prove that BPM converges *linearly* and in a *finite* number of steps in the same regime. Furthermore, by introducing the concept of ball-convexity, we prove that BPM retains the same global convergence guarantees under weaker assumptions, making it a powerful tool for a broader class of potentially non-convex optimization problems. Just like PPM plays the role of a conceptual method inspiring the development of practically efficient algorithms and algorithmic elements, e.g., gradient descent, adaptive step sizes, acceleration (Ahn & Sra, 2020), and "W" in AdamW (Zhuang et al., 2022), we believe that BPM should be understood in the same manner: as a blueprint and inspiration for further development.

[1]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia [2]University of California, Berkeley, USA.

*The work of Aadi Rane was conducted during a VSRP internship at KAUST.

1. Introduction

The minimization of non-convex functions is a fundamental challenge across many fields, including machine learning, optimization, applied mathematics, signal processing and operations research. Solving such problems is integral to most machine learning algorithms arising in both training and inference, where non-convex objectives or constraints are often necessary to capture complex prediction tasks.

1.1. Global nonconvex optimization

In this paper, we propose a new meta-algorithm (see Section 1.3) capable of finding *global* minimizers for a specific (new) class of non-convex functions. In particular, we introduce an algorithmic framework designed to solve the (potentially non-convex) optimization problem

$$\min_{x \in \mathbb{R}^d} f(x), \tag{1}$$

where $f : \mathbb{R}^d \mapsto \mathbb{R} \cup \{+\infty\}$ is assumed to be proper (which means that the set $\operatorname{dom} f := \{x \in \mathbb{R}^d : f(x) < +\infty\}$ is nonempty), closed, and have at least one minimizer. We let $\mathcal{X}_f$ be the set of all minimizers of f, and $f_\star := \min_x f(x)$.

1.2. The ball-proximal operator

A key inspiration for our method stems from the well-known Proximal Point Method (PPM) (Rockafellar, 1976), which iteratively adds a quadratic penalty term to the objective (see Section 3 for more details) and solves a modified subproblem at each step. Building on this idea, we introduce the *ball-proximal ("broximal") operator*:

Definition 1.1 (Ball-Proximal Operator)**.** The *ball-proximal ("broximal") operator* with radius $t > 0$ associated with a function $f : \mathbb{R}^d \mapsto \mathbb{R} \cup \{+\infty\}$ is given by

$$\operatorname{brox}_f^t(x) := \operatorname*{arg\,min}_{z \in B_t(x)} f(z), \tag{2}$$

where $B_t(x) := \{z \in \mathbb{R}^d : \|z - x\| \leq t\}$ and $\|\cdot\|$ is the standard Euclidean norm.

According to this definition, for a given input point $x \in \mathbb{R}^d$, $\operatorname{brox}_f^t(x)$ returns the minimizer(s) of f within the ball of radius t centered at x.

Ball Proximal (“Broximal”) Operator

From Proximal to Broximal Operator:
From Penalty to Constraint

$$\mathrm{prox}_f^{\gamma}(x) = \mathrm{Arg}\min_{y\in\mathbb{R}^d} \left\{ f(y) + \frac{1}{2\gamma}\|y-x\|^2 \right\}$$

penalty

$$\mathrm{brox}_f^{t}(x) = \mathrm{Arg}\min_{y\in\mathbb{R}^d} \{ f(y) \ : \ \|y-x\| \le t \}$$

constraint

Notation for the constraint:

$$\mathcal{B}(x,t) := \{ y \in \mathbb{R}^d \ \mid \ \|y-x\| \le t \}$$

The First Brox Theorem (GLRR 2025)

A1. $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is proper, closed and convex

A2. $x \in \operatorname{dom} f$

$$\operatorname{brox}_f^t(x) \text{ is nonempty}$$

$$\operatorname{Argmin} f \cap \mathcal{B}(x, t) \neq \emptyset \quad \Rightarrow \quad \operatorname{brox}_f^t(x) \subseteq \operatorname{Argmin} f$$

$$\operatorname{Argmin} f \cap \mathcal{B}(x, t) = \emptyset \quad \Rightarrow \quad \begin{cases} \operatorname{brox}_f^t(x) \text{ is a singleton} \\ \|\operatorname{brox}_f^t(x) - x\| = t \end{cases}$$

The Second Brox Theorem (GLRR 2025)

A1. $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is proper, closed and convex

A2. $x \in \operatorname{dom} f$

Let $u \in \operatorname{brox}_f^t(x)$

There exists $c(x,t) \geq 0$ such that

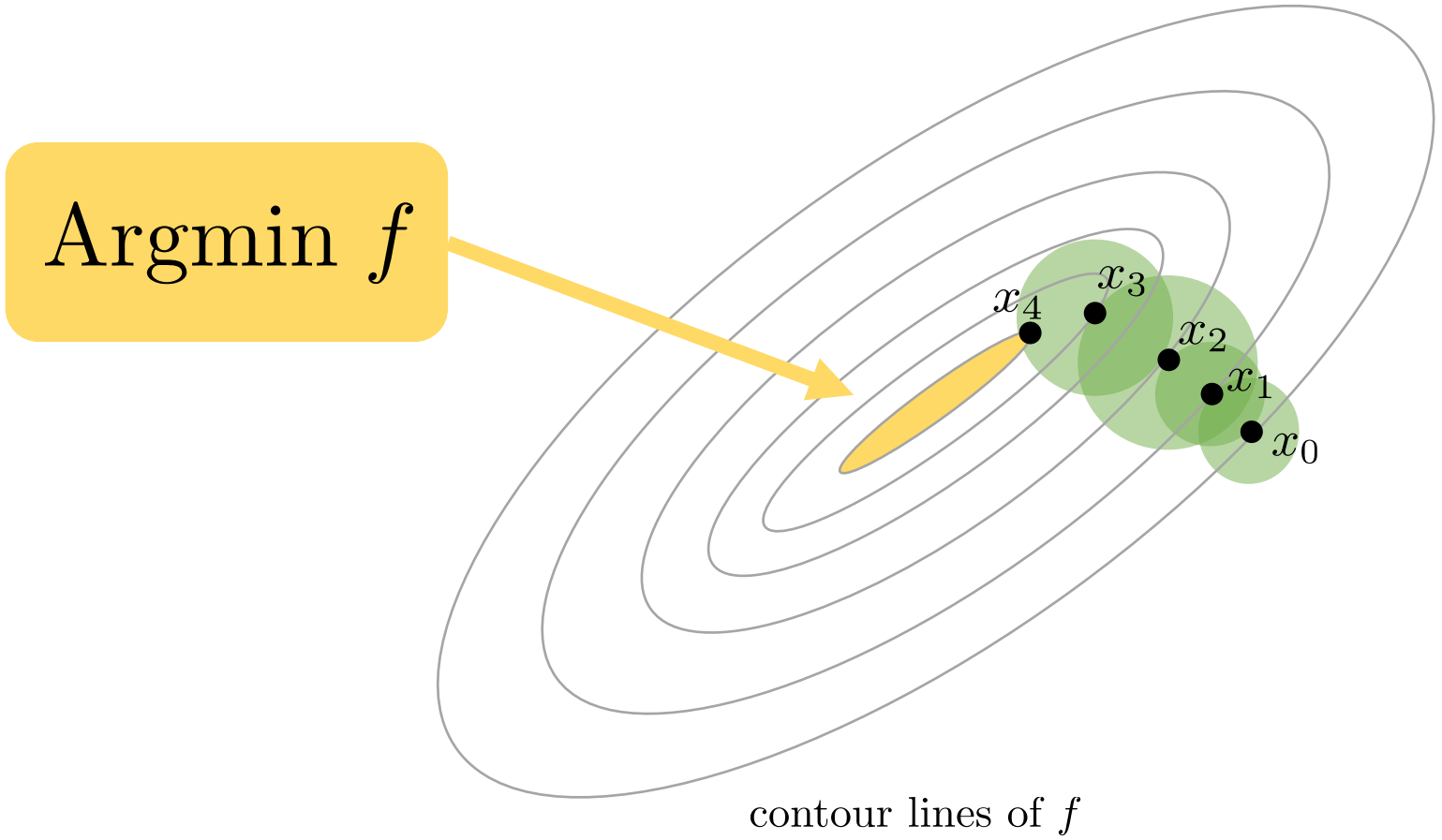
$$c(x,t) \cdot (x - u) \in \partial f(u)$$

$$f(y) - f(u) \geq c(x,t) \cdot \langle x - u, y - u \rangle \text{ for all } y \in \mathbb{R}^d$$

Broximal Point Method (BPM)

Broximal Point Method (BPM)

$$x_{k+1} = \operatorname{brox}_f^{t_k}(x_k) = \operatorname{argmin}\{f(x) \ : \ \|x - x_k\| \leq t_k\}$$



BPM = "Normalized" PPM

$f : \mathbb{R}^d \to \mathbb{R}$
convex & differentiable

$$x_{k+1} = \operatorname{brox}_f^{t_k}(x_k) = x_k - t_k \frac{\nabla f(x_{k+1})}{\|\nabla f(x_{k+1})\|} = \operatorname{prox}_f^{\gamma_k}(x_k)$$

$$\gamma_k = \frac{t_k}{\|\nabla f(x_{k+1})\|}$$

$$\operatorname{brox}_f^t(x) = \operatorname{Arg\,min}_{y \in \mathbb{R}^d} \{ f(y) \ : \ \|y - x\| \le t \}$$

normalized implicit gradient

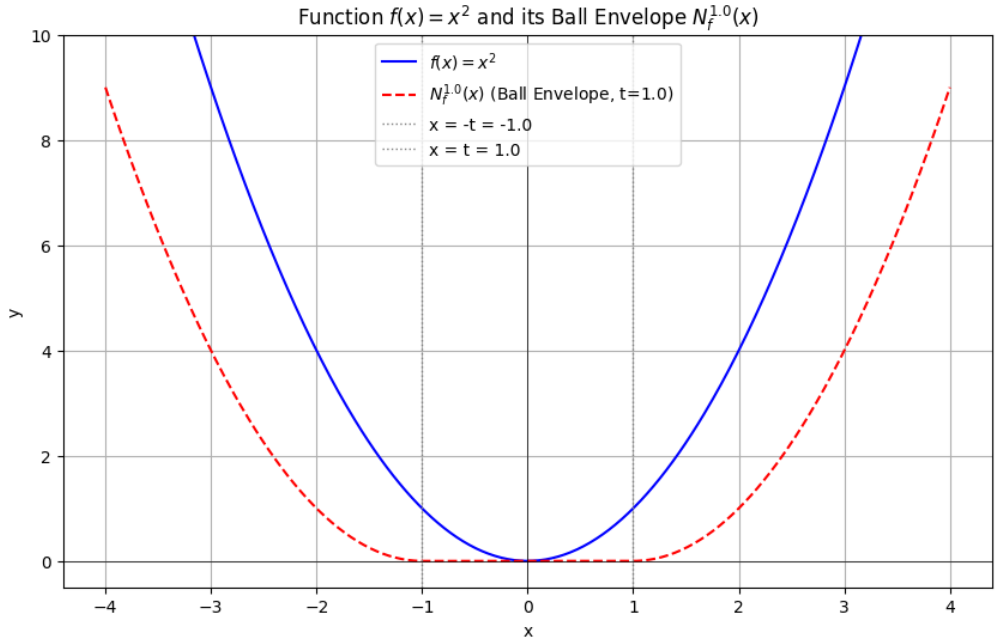
The Ball Envelope

$$\text{Ball envelope of } f\text{:}\quad N_f^t(x) := \min_{\|z-x\|\le t} f(z)$$

$$\text{Argmin } N_f^t = \text{Argmin } f + \{x \in \mathbb{R}^d \mid \|x\| \le t\}$$

Example: $f(x) = x^2$

$$N_f^t(x) = \begin{cases} (x+t)^2 & \text{if } x < -t \\ 0 & \text{if } -t \le x \le t \\ (x-t)^2 & \text{if } x > t \end{cases}$$



BPM = Normalized GD for Minimizing the Ball Envelope

$f : \mathbb{R}^d \to \mathbb{R}$ is convex and L-smooth

$x_{k+1} \notin \operatorname{Argmin} f$

$$x_{k+1} = \operatorname{brox}_f^{t_k}(x_k) = x_k - t_k \frac{\nabla N_f^{t_k}(x_k)}{\|\nabla N_f^{t_k}(x_k)\|}$$

Ball envelope of f:

$$N_f^t(x) := \min_{\|z-x\| \le t} f(z)$$

Seven (Convergence) Results

Assumptions (Convex Setup)

f is proper, closed & convex

$$\operatorname{Argmin} f \neq \emptyset$$

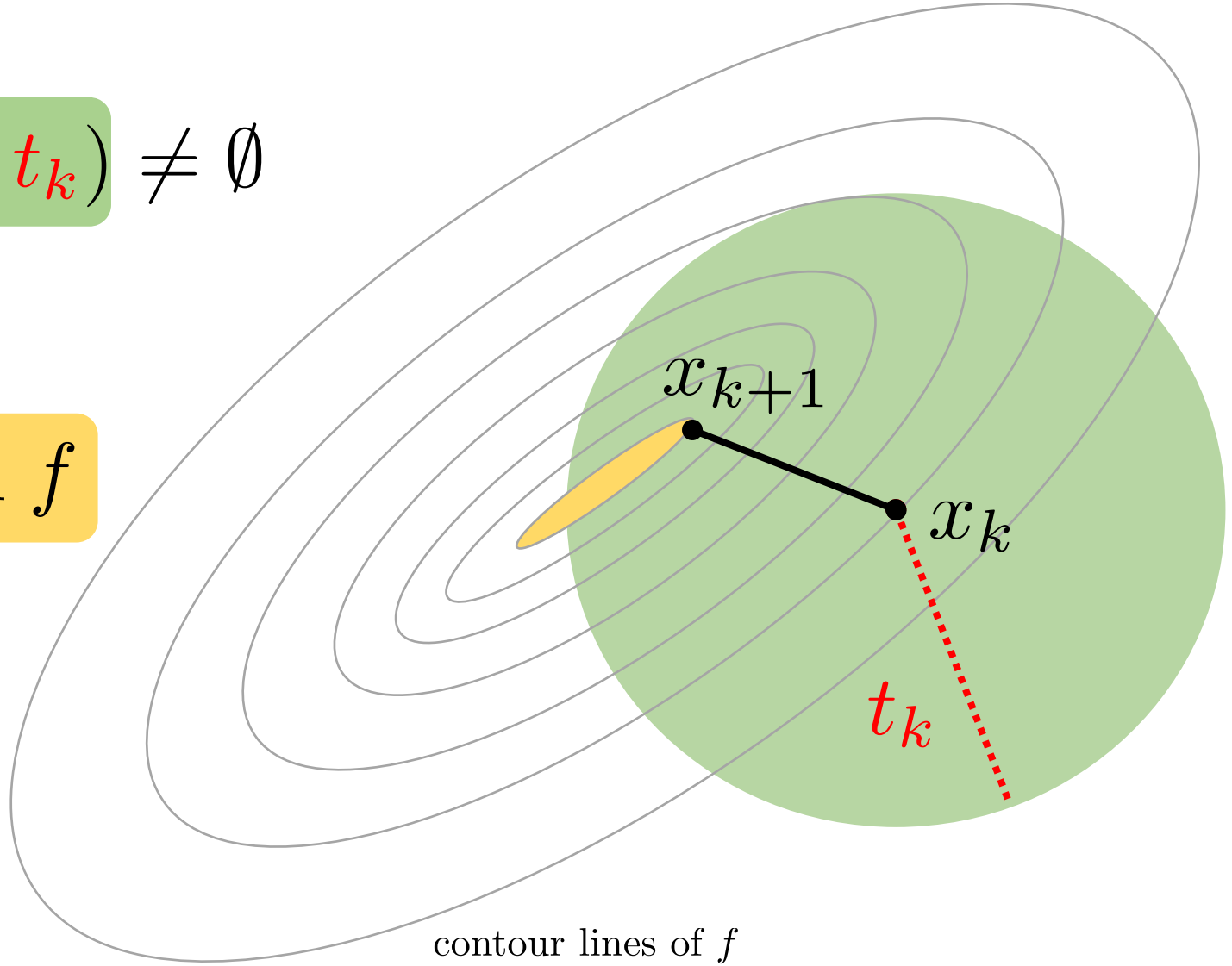
$$x_0 \in \operatorname{dom} f$$

We have **nonconvex theory**
in the paper
(not covered in this talk)

Result 1: One Step Convergence

$$\text{Argmin } f \cap \mathcal{B}(x_k, t_k) \neq \emptyset$$

$$x_{k+1} \in \text{Argmin } f$$



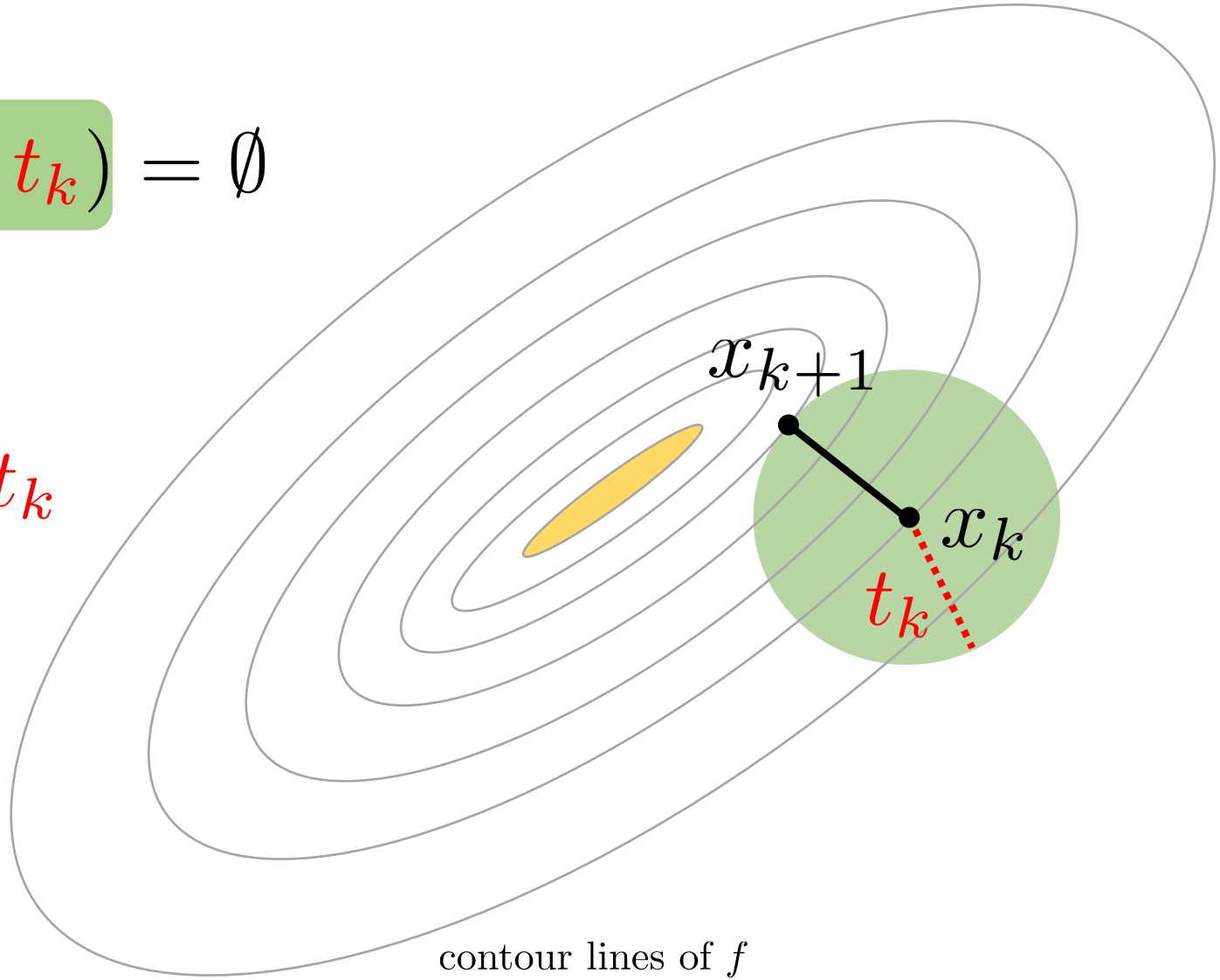
Broximal Point Method (BPM)

$$x_{k+1} \in \text{Argmin}\{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Result 2: Step to Boundary

$$\text{Argmin } f \cap \mathcal{B}(x_k, t_k) = \emptyset$$

$$\|x_{k+1} - x_k\| = t_k$$



Broximal Point Method (BPM)

$$x_{k+1} \in \text{Argmin}\{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Result 3: Super-Accelerated Linear Convergence in Distance to Minimizer

$$\text{Argmin } f \cap \mathcal{B}(x_k, t_k) = \emptyset$$

$x_\star \in \text{Argmin } f$

$$\|x_{k+1} - x_\star\|^2 \leq \|x_k - x_\star\|^2 - t_k^2$$

No need for strong convexity!

$$\left(1 - \frac{t_k^2}{\|x_k - x_\star\|^2}\right) \|x_k - x_\star\|^2$$

Broximal Point Method (BPM)

$$x_{k+1} \in \text{Argmin}\{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Result 4: Finite Convergence (i.e., BPM is a "Direct" Optimization Method)

$x_\star \in \operatorname{Argmin} f$

$$\sum_{k=0}^{K-1} t_k^2 \geq \|x_0 - x_\star\|^2 \quad \Rightarrow \quad x_K \in \operatorname{Argmin} f$$

Broximal Point Method (BPM)

$$x_{k+1} \in \operatorname{Argmin}\{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Corollary:

Constant stepsize regime: $t_k \equiv t$ for all $k \geq 0$

$$K \geq \frac{\|x_0 - x_\star\|^2}{t^2} \quad \Rightarrow \quad x_K \in \operatorname{Argmin} f$$

Result 5: Super-Accelerated Linear Convergence in Function Values

follows from

$$f(x_{k+1}) \leq f(x_k) - t_k \frac{f(x_{k+1}) - f(x_\star)}{\|x_{k+1} - x_\star\|}$$

$x_\star \in \operatorname{Argmin} f$

$$f(x_{k+1}) - f(x_\star) \leq \left(1 + \frac{t_k}{\|x_{k+1} - x_\star\|}\right)^{-1} (f(x_k) - f(x_\star))$$

No need for strong convexity!

Broximal Point Method (BPM)

$$x_{k+1} \in \operatorname{Argmin} \{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Result 6: Gradient Monotonicity

$$f \text{ is differentiable} \quad \& \quad x_0 \in \operatorname{dom} f$$

$$\|\nabla f(x_{k+1})\| \leq \|\nabla f(x_k)\|$$

Broximal Point Method (BPM)

$$x_{k+1} \in \operatorname{Argmin}\{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Result 7: Gradient Convergence

follows from

$$f(x_{k+1}) \leq f(x_k) - t_k \|\nabla f(x_{k+1})\|$$

$x_\star \in \text{Argmin } f$

$$\sum_{k=0}^{K-1} \frac{t_k}{\sum_{s=0}^{K-1} t_s} \|\nabla f(x_{k+1})\| \leq \frac{f(x_0) - f(x_\star)}{\sum_{k=0}^{K-1} t_k}$$

Corollary:

Constant stepsize regime: $t_k \equiv t$ for all $k \geq 0$

$$\|\nabla f(x_{K+1})\| \leq \frac{1}{K} \sum_{k=0}^{K-1} \|\nabla f(x_{k+1})\| \leq \frac{f(x_0) - f(x_\star)}{t \cdot K}$$

Broximal Point Method (BPM)

$$x_{k+1} \in \text{Argmin} \{f(x) \ : \ \|x - x_k\| \leq t_k\}$$

Nonconvex Problems

BPM Converges to Global Optimum for Some Nonconvex Functions with Suboptimal Local Minima

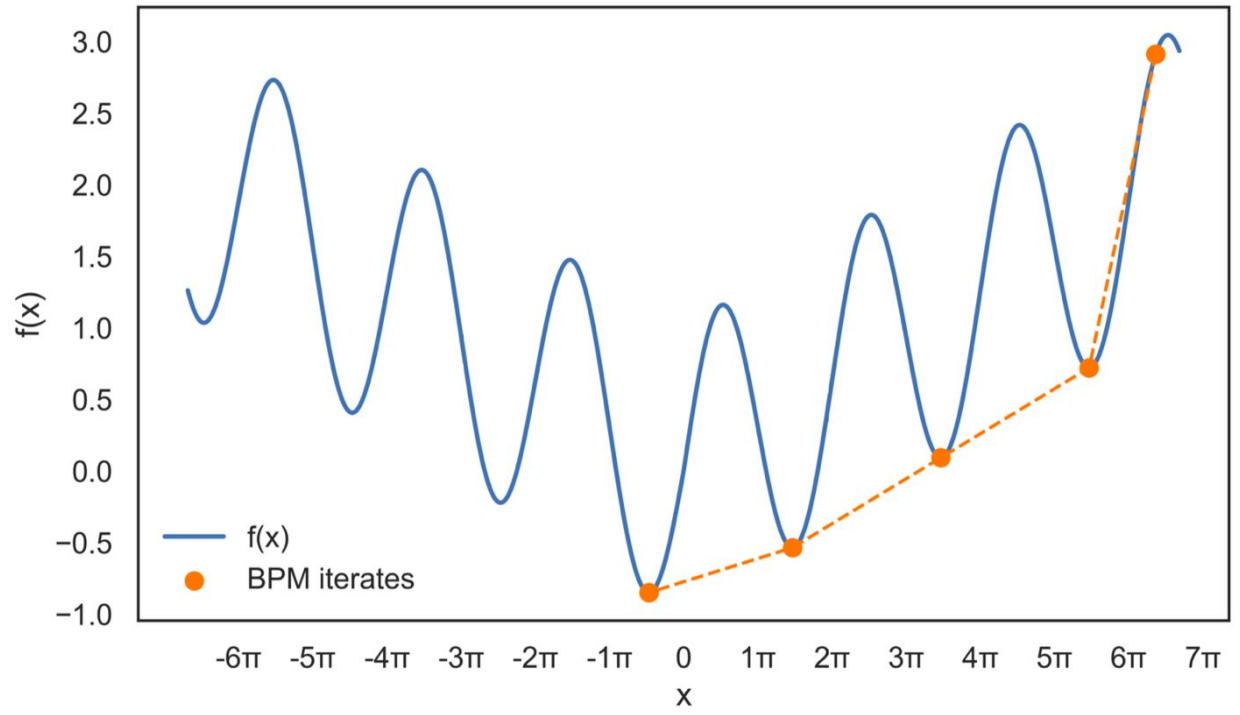


Figure 2. Trajectory of BPM with $t = 2\pi$ starting from $x_0 = 20$ applied to $f(x) = |x| + 10\sin(x)$ (Example 1).

Brox-Aligned Nonconvex Functions

Broximal Alignment for Global Non-Convex Optimization

Table 1. **Comparison of convex and non-convex function classes. Non-Convex** = does not require f to be convex; **Non-Differentiable** = does not require f to be differentiable; **Stationary pts** = allows existence of stationary points that are not global minima; **Non-unique min** = allows existence of multiple global minima; **Disconnected** $\mathcal{X}_f$: the set of minimizers need not be connected.

Function class	Non-Convex	Non-Differentiable	Stationary pts	Non-unique min	Disconnected $\mathcal{X}_f$
Strongly convex	✗	✓	✗	✗	✗
Convex	✗	✓	✗	✓	✗
Pseudoconvex (Mangasarian, 1975)	✓	✗	✗	✓	✗
Quasiconvex (Arrow & Enthoven, 1961)	✓	✓	✓	✓	✗
Star-convex (Nesterov & Polyak, 2006)	✓	✗	✗	✓	✗
Quasar convex (Hardt et al., 2018)	✓	✗	✗	✓	✗
Aiming (Liu et al., 2023)	✓	✗	✗	✓	✗
Polyak-Łojasiewicz (Polyak, 1963; Łojasiewicz, 1963)	✓	✗	✗	✗	✗
Quadratic growth (Bonnans & Ioffe, 1995)	✓	✓	✓	✓	✓
$\mathcal{F}_{\mathbf{X}}^{\mathrm{BA}}(t)$ (ours)	✓	✓	✓	✓	✓

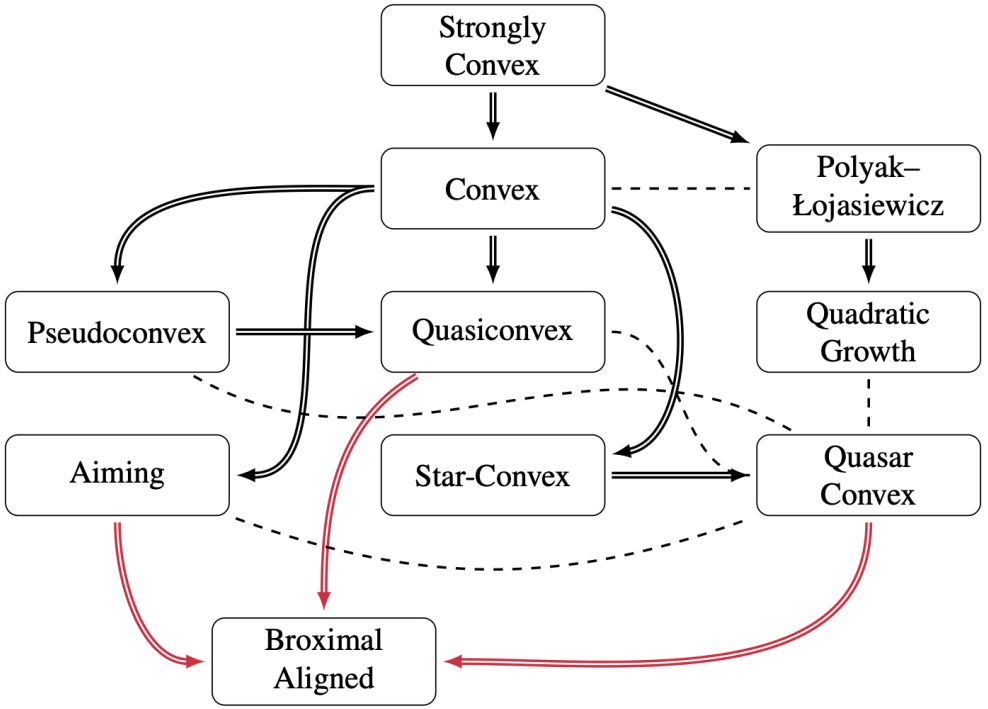


Figure 3. **Hierarchy of assumptions.** Solid arrows encode one-way implications, dashed lines denote mutual non-implications, and red arrows indicate conditions guaranteeing Broximal Alignment for any $t > 0$. For simplicity, we assume f is L-smooth (i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| \; \forall x, y \in \mathbb{R}^d$) with domain $\operatorname{dom} f = \mathbb{R}^d$. See the appendix and Karimi et al. (2016); Hinder et al. (2019); Khanh & Lara (2025) for proofs.

Kaja Gruntkowska, Hanmin Li, Xun Qian, P.R.
Broximal alignment for global non-convex optimization
Submitted (soon on arXiv)

Practical Considerations

BPM: Practical Considerations

1. Some results extend to **arbitrary non-Euclidean norms** [GR 2025]
   (choice of the norm is very important in deep learning, and training of LLMs!)

2. Can be made practical by **approximation:**

approximate f: apply BPM to a model of f

(gives rise to **non-Euclidean trust region methods**)

approximate brox_f^t: use an iterative technique, stop early

BPM: Prior Closely Related Works

Prior Related Work: Acceleration

Acceleration with a Ball Optimization Oracle

Yair Carmon* Arun Jambulapati* Qijia Jiang* Yujia Jin* Yin Tat Lee†
Aaron Sidford* Kevin Tian*

Abstract

Consider an oracle which takes a point x and returns the minimizer of a convex function f in an ℓ_2 ball of radius r around x. It is straightforward to show that roughly $r^{-1} \log \frac{1}{\epsilon}$ calls to the oracle suffice to find an ϵ-approximate minimizer of f in an ℓ_2 unit ball. Perhaps surprisingly, this is not optimal: we design an accelerated algorithm which attains an ϵ-approximate minimizer with roughly $r^{-2/3} \log \frac{1}{\epsilon}$ oracle queries, and give a matching lower bound. Further, we implement ball optimization oracles for functions with locally stable Hessians using a variant of Newton's method. The resulting algorithm applies to a number of problems of practical and theoretical import, improving upon previous results for logistic and ℓ_∞ regression and achieving guarantees comparable to the state-of-the-art for ℓ_p regression.

1 Introduction

We study unconstrained minimization of a smooth convex objective $f : \mathbb{R}^d \to \mathbb{R}$, which we access through a *ball optimization oracle* $\mathcal{O}_{\text{ball}}$, that when queried at any point x, returns the minimizer[1] of f restricted a ball of radius r around x, i.e.,

$$\mathcal{O}_{\text{ball}}(x) = \operatorname*{arg\,min}_{x' \text{ s.t. } \|x'-x\| \le r} f(x').$$

Such oracles underlie trust region methods [12] and, as we demonstrate via applications, encapsulate problems with local stability. Iterating $x_{k+1} \leftarrow \mathcal{O}_{\text{ball}}(x_k)$ minimizes f in $\widetilde{O}(R/r)$ iterations (see Appendix A), where R is the initial distance to the minimizer, x^*, and $\widetilde{O}(\cdot)$ hides polylogarithmic factors in problem parameters, including the desired accuracy.

Given the fundamental geometric nature of the ball optimization abstraction, the central question motivating our work is whether it is possible to improve upon this $\widetilde{O}(R/r)$ query complexity. It is natural to conjecture that the answer is negative: we require R/r oracle calls to observe the entire line from x_0 to the optimum, and therefore finding a solution using less queries would require jumping into completely unobserved regions. Nevertheless, we prove that the optimal query complexity scales as $(R/r)^{2/3}$. This result has positive implications for the complexity for several key regression tasks, for which we can efficiently implement the ball optimization oracles.

1.1 Our contributions

Here we overview the main contributions of our paper: accelerating ball optimization oracles (with a matching lower bound), implementing them under Hessian stability, and applying the resulting techniques to regression problems.

*Stanford University, {yairc,jmblpati,qjiang2,yujiajin,sidford,kjtian}@stanford.edu.
†University of Washington, yintat@uw.edu.

[1] In the introduction we discuss exact oracles for simplicity, but our results account for inexactness.

Prior Related Work: Non-Euclidean Balls

ℓ_∞ balls

Matrix Scaling and Balancing via Box Constrained Newton's Method and Interior Point Methods

Michael B. Cohen* MIT micohen@mit.edu
Aleksander Mądry† MIT madry@mit.edu
Dimitris Tsipras† MIT tsipras@mit.edu
Adrian Vladu‡ MIT avladu@mit.edu

Abstract

In this paper, we study matrix scaling and balancing, which are fundamental problems in scientific computing, with a long line of work on them that dates back to the 1960s. We provide algorithms for both these problems that, ignoring logarithmic factors involving the dimension of the input matrix and the size of its entries, both run in time $\widetilde{O}\left(m \log \kappa \log^2(1/\varepsilon)\right)$ where ε is the amount of error we are willing to tolerate. Here, κ represents the ratio between the largest and the smallest entries of the optimal scalings. This implies that our algorithms run in nearly-linear time whenever κ is quasi-polynomial, which includes, in particular, the case of strictly positive matrices. We complement our results by providing a separate algorithm that uses an interior-point method and runs in time $\widetilde{O}(m^{3/2} \log(1/\varepsilon))$.

In order to establish these results, we develop a new second-order optimization framework that enables us to treat both problems in a unified and principled manner. This framework identifies a certain generalization of linear system solving that we can use to efficiently minimize a broad class of functions, which we call *second-order robust*. We then show that in the context of the specific functions capturing matrix scaling and balancing, we can leverage and generalize the work on Laplacian system solving to make the algorithms obtained via this framework very efficient.

*This material is based upon work supported by the National Science Foundation under Grant No. 1111109 and Grant No. 1553428, and by the National Defense Science and Engineering Graduate Fellowship.
†This material is based upon work supported by the National Science Foundation under Grant No. 1553428.
‡This material is based upon work supported by the National Science Foundation under Grant No. 1111109 and Grant No. 1553428

ℓ_p balls

Convex optimization with p-norm oracles

Deeksha Adil
Institute for Theoretical Studies
ETH Zürich
deeksha.adil@eth-its@ethz.ch

Brian Bullins
Department of Computer Science
Purdue University
bbullins@purdue.edu

Arun Jambulapati
Department of Computer Science
University of Michigan
jmblpati@umich.edu

Aaron Sidford
Department of Computer Science
Stanford University
sidford@stanford.edu

November 1, 2024

Abstract

In recent years, there have been significant advances in efficiently solving ℓ_s-regression using linear system solvers and ℓ_2-regression [Adil-Kyng-Peng-Sachdeva, J. ACM'24]. Would efficient ℓ_p-norm solvers lead to even faster rates for solving ℓ_s-regression when $2 \leq p < s$? In this paper, we give an affirmative answer to this question and show how to solve ℓ_s-regression using $\widetilde{O}(n^{\frac{1}{1+\nu}})$ iterations of solving smoothed ℓ_s regression problems, where $\nu := \frac{1}{p} - \frac{1}{s}$. To obtain this result, we provide improved accelerated rates for convex optimization problems when given access to an $\ell_p^s(\lambda)$-*proximal oracle*, which, for a point c, returns the solution of the regularized problem $\min_x \boldsymbol{f}(\boldsymbol{x}) + \lambda \|\boldsymbol{x} - \boldsymbol{c}\|_p^s$. Additionally, we show that the rates we establish for the $\ell_p^s(\lambda)$-proximal oracle are near-optimal.

Contents

Prior Related Work: Parallel Optimization

Closing the Computational-Query Depth Gap in Parallel Stochastic Convex Optimization

Arun Jambulapati
University of Michigan
jmblpati@gmail.com

Aaron Sidford
Stanford University
sidford@stanford.edu

Kevin Tian
University of Texas at Austin
kjtian@cs.utexas.edu

Abstract

We develop a new parallel algorithm for minimizing Lipschitz, convex functions with a stochastic subgradient oracle. The total number of queries made and the query depth, i.e., the number of parallel rounds of queries, match the prior state-of-the-art, [CJJ+23], while improving upon the computational depth by a polynomial factor for sufficiently small accuracy. When combined with previous state-of-the-art methods our result closes a gap between the best-known query depth and the best-known computational depth of parallel algorithms.

Our method starts with a *ball acceleration* framework of previous parallel methods, i.e., [CJJ+20, ACJ+21], which reduce the problem to minimizing a regularized Gaussian convolution of the function constrained to Euclidean balls. By developing and leveraging new stability properties of the Hessian of this induced function, we depart from prior parallel algorithms and reduce these ball-constrained optimization problems to stochastic unconstrained quadratic minimization problems. Although we are unable to prove concentration of the asymmetric matrices that we use to approximate this Hessian, we nevertheless develop an efficient parallel method for solving these quadratics. Interestingly, our algorithms can be improved using fast matrix multiplication and use nearly-linear work if the matrix multiplication exponent is 2.

Contents

ReSQueing Parallel and Private Stochastic Convex Optimization

Yair Carmon* Arun Jambulapati† Yujia Jin‡ Yin Tat Lee§ Daogao Liu†

Aaron Sidford‡ Kevin Tian§

Abstract

We introduce a new tool for stochastic convex optimization (SCO): a Reweighted Stochastic Query (ReSQue) estimator for the gradient of a function convolved with a (Gaussian) probability density. Combining ReSQue with recent advances in *ball oracle acceleration* [CJJ+20, ACJ+21], we develop algorithms achieving state-of-the-art complexities for SCO in parallel and private settings. For a SCO objective constrained to the unit ball in $\mathbb{R}^d$, we obtain the following results (up to polylogarithmic factors).

1. We give a parallel algorithm obtaining optimization error ϵ_{opt} with $d^{1/3}\epsilon_{\text{opt}}^{-2/3}$ gradient oracle query depth and $d^{1/3}\epsilon_{\text{opt}}^{-2/3} + \epsilon_{\text{opt}}^{-2}$ gradient queries in total, assuming access to a bounded-variance stochastic gradient estimator. For $\epsilon_{\text{opt}} \in [d^{-1}, d^{-1/4}]$, our algorithm matches the state-of-the-art oracle depth of [BJL+19] while maintaining the optimal total work of stochastic gradient descent.
2. Given n samples of Lipschitz loss functions, prior works [BFTT19, BFGT20, AFKT21, KLL21] established that if $n \gtrsim d\epsilon_{\text{dp}}^{-2}$, $(\epsilon_{\text{dp}}, \delta)$-differential privacy is attained at no asymptotic cost to the SCO utility. However, these prior works all required a superlinear number of gradient queries. We close this gap for sufficiently large $n \gtrsim d^2\epsilon_{\text{dp}}^{-3}$, by using ReSQue to design an algorithm with near-linear gradient query complexity in this regime.

*Tel Aviv University, ycarmon@tauex.tau.ac.il.
†University of Washington, {jmblpati, dgliu}@uw.edu.
‡Stanford University, {yujiajin, sidford}@stanford.edu.
§Microsoft Research, {yintatlee, tiankevin}@microsoft.com.

Prior Related Work: Minimize Maximum Loss

Stochastic Bias-Reduced Gradient Methods

Hilal Asi Yair Carmon Arun Jambulapati Yujia Jin Aaron Sidford
{asi,jmblpati,yujiajin,sidford}@stanford.edu ycarmon@cs.tau.ac.il

Abstract

We develop a new primitive for stochastic optimization: a low-bias, low-cost estimator of the minimizer $x_\star$ of any Lipschitz strongly-convex function. In particular, we use a multilevel Monte-Carlo approach due to Blanchet and Glynn [8] to turn any optimal stochastic gradient method into an estimator of $x_\star$ with bias δ, variance $O(\log(1/\delta))$, and an expected sampling cost of $O(\log(1/\delta))$ stochastic gradient evaluations. As an immediate consequence, we obtain cheap and nearly unbiased gradient estimators for the Moreau-Yoshida envelope of any Lipschitz convex function, allowing us to perform dimension-free randomized smoothing.

We demonstrate the potential of our estimator through four applications. First, we develop a method for minimizing the maximum of N functions, improving on recent results and matching a lower bound up to logarithmic factors. Second and third, we recover state-of-the-art rates for projection-efficient and gradient-efficient optimization using simple algorithms with a transparent analysis. Finally, we show that an improved version of our estimator would yield a nearly linear-time, optimal-utility, differentially-private non-smooth stochastic optimization method.

1 Introduction

Consider the fundamental problem of minimizing a μ-strongly convex function $F : \mathcal{X} \to \mathbb{R}$ given access to a stochastic (sub-)gradient estimator $\hat{\nabla}F$ satisfying $\mathbb{E}\hat{\nabla}F(x) \in \partial F(x)$ and $\mathbb{E}\|\hat{\nabla}F(x)\|^2 \le G^2$ for every $x \in \mathcal{X}$. Is it possible to transform the unbiased estimator $\hat{\nabla}F$ into a (nearly) unbiased estimator of the minimizer $x_\star := \operatorname{argmin}_{x\in\mathcal{X}} F(x)$? In particular, can we improve upon the $O(G/(\mu\sqrt{T}))$ bias achieved by T iterations of stochastic gradient descent (SGD)?

In this paper, we answer this question in the affirmative, proposing an *optimum estimator* $\hat{x}_\star$, which (for any fixed $\delta > 0$) has

$$\text{bias } \|\mathbb{E}\hat{x}_\star - x_\star\| = \delta \text{ and variance } \mathbb{E}\|\hat{x}_\star - \mathbb{E}\hat{x}_\star\|^2 = O\left(\frac{G^2}{\mu^2}\log\left(\frac{G}{\mu\delta}\right)\right),$$

and, *in expectation*, costs $O(\log(\frac{G}{\mu\delta}))$ evaluations of $\hat{\nabla}F$.[1] Setting $\delta = G/(\mu\sqrt{T})$, we obtain the same bias bound as T iterations of SGD, but with expected cost of only $O(\log T)$ stochastic gradient evaluations (the worst-case cost is T). Further, the bias can be made arbitrarily small with only logarithmic increase in the variance and the stochastic gradient evaluations of our estimator, and therefore—paralleling the term "nearly linear-time" [27]—we call $\hat{x}_\star$ *nearly unbiased*.

Our estimator is an instance of the multilevel Monte Carlo technique for de-biasing estimator sequences [25] and more specifically the method of Blanchet and Glynn [8]. Our key observation is that this method is readily applicable to strongly-convex variants of SGD, or indeed any stochastic optimization method with the same (optimal) rate of convergence.

[1] When $\mathcal{X} = \mathbb{B}_R(x_0) \subset \mathbb{R}^d$, $F(x) = \frac{1}{n}\sum_{i\in[n]} \hat{F}(x;i)$, and $\hat{\nabla}F$ is the subgradient of a uniformly random $\hat{F}(x;i)$ we can also get an estimator with bias 0 and expected cost $O(\log(nd))$. See Appendix A.1 for details.

Thinking Inside the Ball: Near-Optimal Minimization of the Maximal Loss

Yair Carmon Arun Jambulapati Yujia Jin Aaron Sidford
ycarmon@cs.tau.ac.il, {jmblpati,yujiajin,sidford}@stanford.edu

Abstract

We characterize the complexity of minimizing $\max_{i\in[N]} f_i(x)$ for convex, Lipschitz functions $f_1, \ldots, f_N$. For non-smooth functions, existing methods require $O(N\epsilon^{-2})$ queries to a first-order oracle to compute an ϵ-suboptimal point and $\widetilde{O}(N\epsilon^{-1})$ queries if the f_i are $O(1/\epsilon)$-smooth. We develop methods with improved complexity bounds of $\widetilde{O}(N\epsilon^{-2/3}+\epsilon^{-8/3})$ in the non-smooth case and $\widetilde{O}(N\epsilon^{-2/3}+\sqrt{N}\epsilon^{-1})$ in the $O(1/\epsilon)$-smooth case. Our methods consist of a recently proposed ball optimization oracle acceleration algorithm (which we refine) and a careful implementation of said oracle for the softmax function. We also prove an oracle complexity lower bound scaling as $\Omega(N\epsilon^{-2/3})$, showing that our dependence on N is optimal up to polylogarithmic factors.

1 Introduction

Consider the problem of approximately minimizing the maximum of N convex functions: given $f_1, \ldots, f_N$ such that for every $i \in [N]$ the function $f_i : \mathbb{R}^d \to \mathbb{R}$ is convex, Lipschitz and possibly smooth, and a target accuracy ϵ,

$$\text{find a point } x \text{ such that } F_{\max}(x) - \inf_{x_\star\in\mathbb{R}^d} F_{\max}(x_\star) \le \epsilon \text{ where } F_{\max}(x) := \max_{i\in[N]} f_i(x)\,. \quad (1)$$

Problems of this form play significant roles in optimization and machine learning. The maximum of N functions is a canonical example of structured non-smoothness and several works develop methods for exploiting it [31, 30, 36, 9, 12]. The special case where the f_i's are linear functions is particularly important for machine learning, since it is equivalent to hard-margin SVM training (with f_i representing the negative margin on the ith example) [38, 13, 21]. Going beyond the linear case, Shalev-Shwartz and Wexler [36] argue that minimizing the maximum classification loss can have advantageous effects on training speed and generalization in the presence of rare informative examples. Moreover, minimizing the worst-case objective is the basic paradigm of robust optimization [4, 27]. In particular, since $F_{\max}(x) = \max_{p\in\Delta^N} \sum_{i\in[N]} p_i f_i(x)$ the problem corresponds to an extreme case of distributionally robust optimization [5] with an uncertainty set that encompasses the entire probability simplex Δ^N.

The goal of this paper is to characterize the complexity of this fundamental problem. We are particularly interested in the regime where the number of data points N and the problem dimension d are large compared to the desired level of accuracy $1/\epsilon$, as is common in modern machine learning. Consequently, we focus on dimension-independent first-order methods (i.e., methods which only rely on access to $f_i(x)$ and a (sub)gradient $\nabla f_i(x)$ as opposed to higher-order derivatives), and report complexity in terms of the number of function/gradient evaluations required to solve the problem.

Part 2
From Adam to Muon: Orthogonalizing the Gradient

Shampoo (2018)

Algorithm 1 Shampoo, matrix case.

Initialize $W_1 = \mathbf{0}_{m \times n}$; $L_0 = \epsilon I_m$; $R_0 = \epsilon I_n$

for $t = 1, \ldots, T$ **do:**

Receive loss function $f_t : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$

Compute gradient $G_t = \nabla f_t(W_t)$ // $G_t \in \mathbb{R}^{m \times$

Update preconditioners:

$$L_t = L_{t-1} + G_t G_t^\mathsf{T}$$

$$R_t = R_{t-1} + G_t^\mathsf{T} G_t$$

Update parameters:

$$W_{t+1} = W_t - \eta L_t^{-1/4} G_t R_t^{-1/4}$$

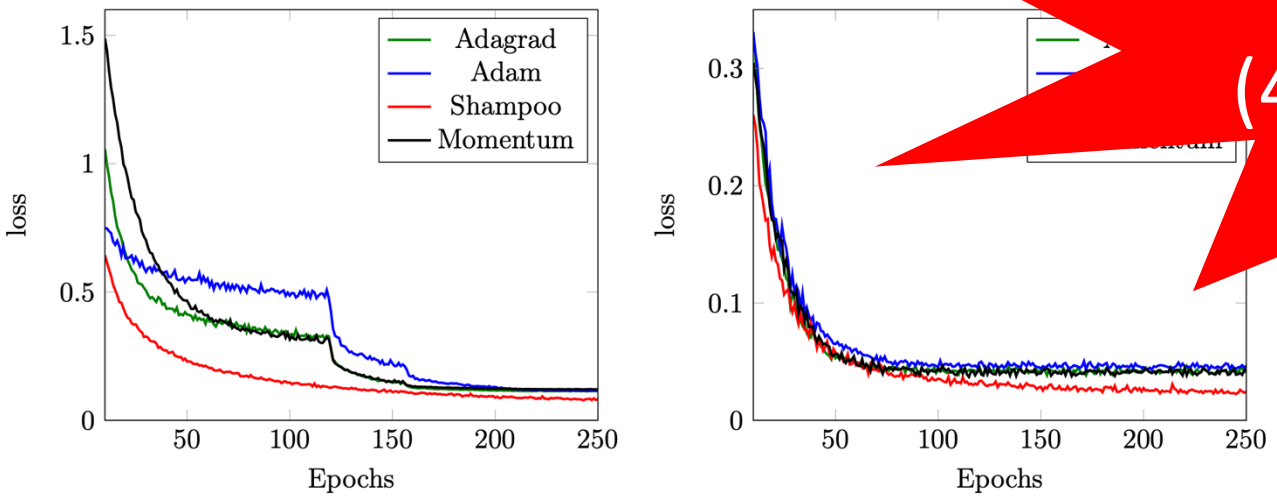


Figure 2: Training loss for a residual network and an inception network on CIFAR-10.

Promising, but did not replace Adam (441 citations "only")

Shampoo: Preconditioned Stochastic Tensor Optimization

Vineet Gupta [1] Tomer Koren [1] Yoram Singer [2 1]

Abstract

Preconditioned gradient methods are am
most general and powerful tools in op
However, preconditioning require
manipulating prohibitively large
he and analyze a new struc
algorithm, called S
on over tens
preco

Nocedal, 1980) that can be used whenever second-order information is unavailable or too expensive to compute. Newer additions to this family are preconditioned online algorithms, most notably AdaGrad (Duchi et al., 2011), that use the covariance matrix of the accumulated gradients to form a preconditioner.

While preconditioned methods often lead to improved con-
vergence pro
ality of typical problems
out-of-the-box use of full-
mitigate this issue, specialized
d in which the full preconditioner
gonal approximation (Duchi et al.,
2014), a sketched version (Gonen
ight 2017).
the
anks to their
other approaches
as one typically requires
of the preconditioner that
computation.

recondi-
servation that in numerous machine
parameter space entertains a more
monolithic vector in Euclidean
ms the parameters form a matrix
e number of features and n is
ural networks, the parameters
form an $m \times n$ matrix with n
es and m is the number of
s of convolutional neural
of 4 dimensional tensors
of the form input-depth × width × height × output-depth.
As a matter of fact, machine learning software tools such as Torch and TensorFlow are designed with tensor structure in mind.

Our algorithm, which we call Shampoo,[1] retains the tensor structure of the gradient and maintains a separate preconditioner matrix for each of its dimensions. An illustration of Shampoo is provided in Figure 1. The set of preconditioners

[1]Google Brain, Mountain View, CA, USA [2]Princeton University, Princeton, NJ, USA. Correspondence to: Tomer Koren <tkoren@google.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

[1]We call it Shampoo because it has to do with pre-conditioning.

Orthogonal SGDM (2022)

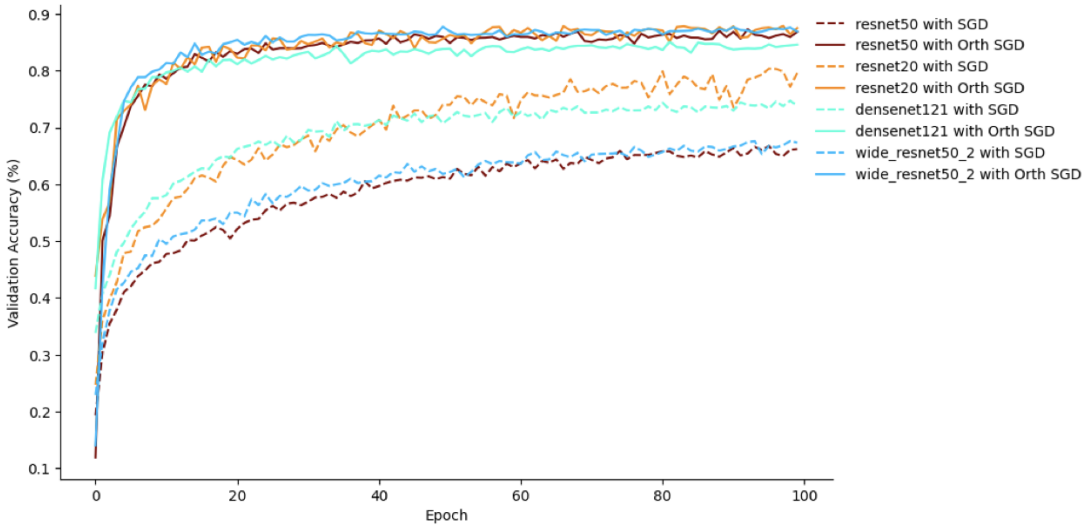


Figure 2: Validation accuracy from one run of SGDM vs Orthogonal-SGDM for a selection of models. Full plot in Appendix C. Best viewed in colour.

Orthogonalising gradients to speedup neural network optimisation

Mark Tuddenham [1] **Adam Prügel-Bennett** [1] **Jonathon Hare** [1]

Abstract

The optimisation of neural networks can be sped up by orthogonalising the gradients before the optimisation step, ensuring the diversification of the learned representations. We orthogonalise the gradients of the layer's components/-filters with respect to each other to separate out the intermediate representations. Our method of orthogonalisation allows the weights to be used more flexibly, in contrast to restricting the weights to an orthogonalised sub-space. We tested this method on ImageNet and CIFAR-10 resulting in a large decrease in learning time, and also obtain a speed-up on the semi-supervised learning BarlowTwins. We obtain similar accuracy to SGD without fine-tuning and better accuracy for naïvely chosen hyper-parameters.

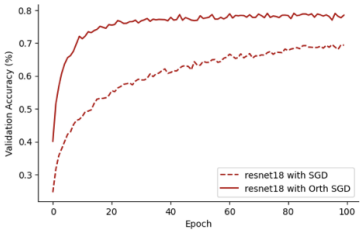


Figure 1: An example of the speed-up obtained by orthogonalising the gradients on CIFAR-10.

1. Introduction

Neural network layers are made up of several identical, but differently parametrised, components, *e.g.* filters in a convolutional layer, or heads in a multi-headed attention layer. Layers consist of several components so that they can provide a diverse set of intermediary representations to the next layer, however, there is no constraint or bias, other than the implicit bias from the cost function, to learning different parametrisations. We introduce this diversification bias in the form of orthogonalised gradients and find a resultant speed-up in learning and sometimes improved performance, see Figure 1.

Our novel contributions include this new optimisation method, thorough testing on CIFAR-10 and ImageNet, additional testing on a semi-supervised learning method, and experiments to support our hypothesis.

In Section 2 we detail the method and results to give an understanding of how this method works and its capabilities. Then, in Section 3, we provide experimental justifications and supporting experiments for this method along with finer details of the implementation and limitations.

2. Overview of new method and results

2.1. Related works

Gradient orthogonalisation has been explored in the domain of multi-task learning (Yu et al., 2020) to keep the different tasks separate and relevant. However in this work we focus on orthogonalisation for improving single task performance.

Weight orthogonalisation has been extensively explored with both empirical (Bansal et al., 2018; Jia et al., 2017) and theoretical (Jia et al., 2019) justifications. However, modifying the weights during training is unstable, and, in addition, it limits the weights to a tiny subspace. Deep learning is know to work well despite the immense size of the weight space, and as such we do not view this as an advantage. Xie et al. (2017) obtain improved performance over Stochastic Gradient Descent (SGD) via weight orthogonalisation and allows them to train very deep networks, we aim to achieve the same results while being more flexible with model and optimisation method choice. We do this by orthogonalising the gradients before they are used by an optimisation method rather than modifying the weights themselves. This, in effect, biases the training towards learning orthogonal representations and we show this to be the case;

[1]Department of Electronics and Computer Science, University of Southampton, Southampton, UK. Correspondence to: Mark Tuddenham <mark.tuddenham@southampton.ac.uk>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Orthogonal SGDM (2022)

No theory
16 citations only as of today!

2.2. Orthogonalising Gradients

Given a neural netw[illegible] f, with L layers m[illegible] from components, c,

$$f = \circ_{i=1}^{L} (f_i),$$
$$f_l(x) = [illegible]$$

where $\circ$ is the compositio[illegible] of components in laye[illegible] parametrised function and c_{li} deno[illegible] $\theta_{li} \in \mathbb{R}^{P_l}$ giving $f_l : \mathbb{R}^{S_{l-1} \times N_{l-1}}$ [illegible] by $\theta_l \in \mathbb{R}^{P_l \times N_l}$.

Let

$$G_l = [\nabla c_{l1}, \nabla c_{l2}, \ldots, \nabla c_{l N_l}], \tag{3}$$

be the $P_l \times N_l$ matrix of the components' gradients.

Then the nearest orthonormal matrix, *i.e.* the orthonormal matrix, O_l, that minimises the Frobenius norm of its difference from G_l

$$\min_{O_l} \|O_l - G_l\| \quad \text{subject to} \quad \forall i, j : \langle O_{li}, O_{lj} \rangle = \delta_{ij},$$

where δ_{ij} is the Kronecker delta function, is the product of the left and right singular vector matricies from the Singular Value Decomposition (SVD) of G_l (Trefethen & Bau III, 1997),

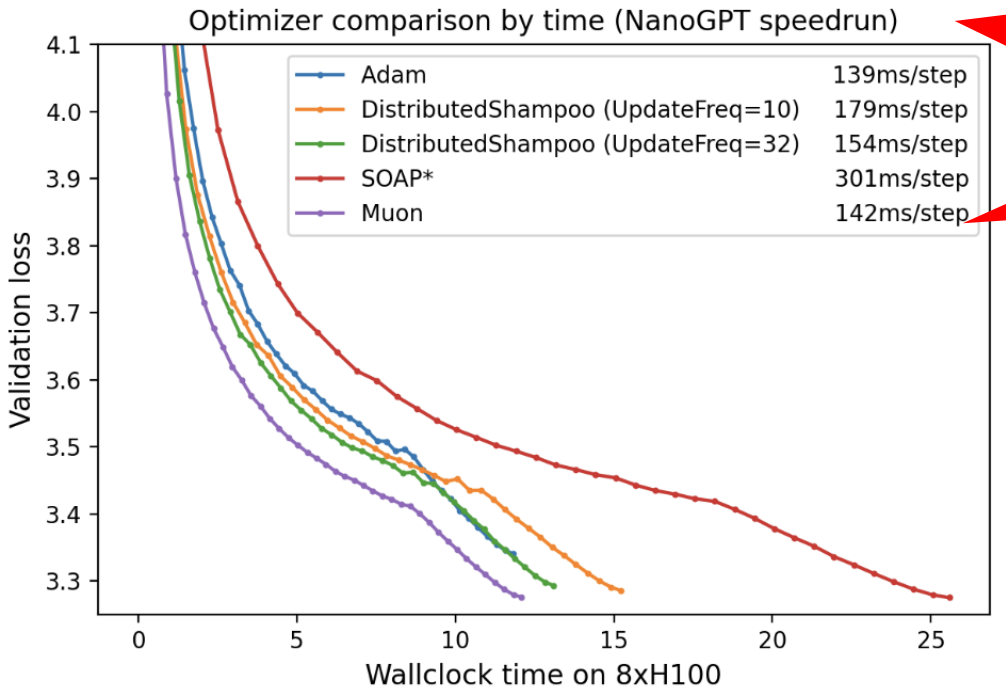
$$G_l = U_l \Sigma_l V_l^{\mathsf{T}}, \tag{4}$$
$$O_l = U_l V_l^{\mathsf{T}}. \tag{5}$$

[illegible]just a first-order gradient descent method, [illegible]Stochastic Gradient Descent with Momentum (Polyak, 1964), to make steps where the components [illegible]ushed in orthogonal directions,

$$v_l^{(t+1)} = \gamma v_l^{(t)} + \eta O_l^{(t)}, \text{ and} \tag{6}$$
$$\theta_l^{(t+1)} = \theta_l^{(t)} - v_l^{(t+1)}, \tag{7}$$

where v_l is the velocity matrix, $t \in \mathbb{Z}^{0+}$ is the time, γ is the momentum decay term, and η is the step size. We call this method Orthogonal Stochastic Gradient Descent with Momentum (Orthogonal-SGDM). This modification can clearly be applied to any first-order optimisation algorithm by replacing the gradients with $O_l^{(t)}$ before the calculation of the next iterate.

Muon (12/2024)



*SOAP is under active development. Future versions will significantly improve the wallclock overhead.

Figure 2. Optimizer comparison by wallclock time.

Keller Jordan blog

About

Muon: An optimizer for hidden layers in neural networks

December 8, 2024 · 16 min

Muon is an o zer for the hidden lay l networks. It is used in the current training sp for both Na AR-10 speedrunning.

writeup will ide an overview of

mization

ers. It is defined

Algor n 2 Muon

Requi e: Learning rate η, momentum μ

1: Initialize $B_0 \leftarrow 0$
2: **for** $t = 1, \ldots$ **do**
3: Compute gradient $G_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta_{t-1})$
4: $B_t \leftarrow \mu B_{t-1} + G_t$
5: $O_t \leftarrow \text{NewtonSchulz5}(B_t)$
6: Update parameters $\theta_t \leftarrow \theta_{t-1} - \eta O_t$
7: **end for**
8: **return** θ_t

Consistently beats Adam!

Efficient Orthonormalization in Muon

Newton-Schulz 5

```python
# Pytorch code
def newtonschulz5(G, steps=5, eps=1e-7):
    assert G.ndim == 2
    a, b, c = (3.4445, -4.7750, 2.0315)
    X = G.bfloat16()
    X /= (X.norm() + eps)
    if G.size(0) > G.size(1):
        X = X.T
    for _ in range(steps):
        A = X @ X.T
        B = b * A + c * A @ A
        X = a * X + B @ X
    if G.size(0) > G.size(1):
        X = X.T
    return X
```

By Keller Jordan (blog; 12/2024)

Optimal method (6/2025)

The Polar Express: Optimal Matrix Sign Methods and Their Application to the Muon Algorithm

Noah Amsel* David Persson† Christopher Musco‡ Robert M. Gower§

June 4, 2025

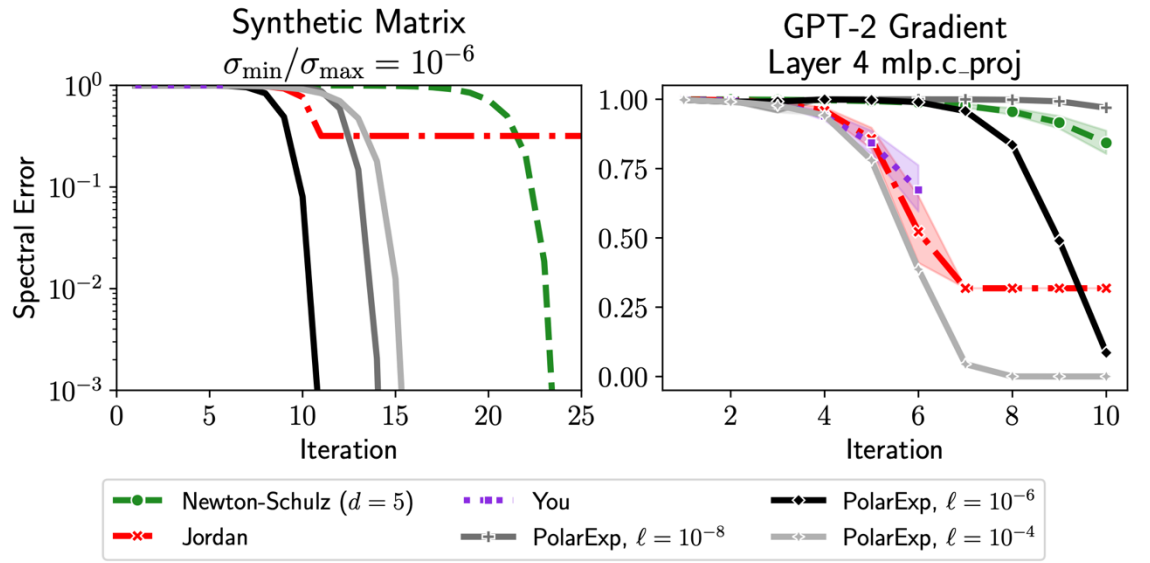


Figure 4: Convergence of various degree-5 polynomial methods in the spectral norm. When tuned properly, Polar Express attains outperforms the other methods at every iteration. Left panel: synthetic matrix with $\sigma_{\max} = 1$, $\sigma_{\min} = 10^{-6}$. Right panel: gradient of a certain weight matrix of a randomly-initialized GPT-2 architecture on a batch of language modeling data, normalized by the Frobenius norm.

Muon Theory

A Note on the Convergence of Muon

Jiaxiang Li * Mingyi Hong †

Abstract

In this note, we inspect the convergence of a new optimizer for pretraining LLMs, namely the Muon optimizer. Such an optimizer is closely related to a specialized steepest descent method where the update direction is the minimizer of the quadratic approximation of the objective function under spectral norm (Bernstein and Newhouse, 2024). We provide the convergence analysis on both versions of the optimizer and discuss its implications.

1 Introduction

Recently, a new optimizer named Muon[1] has drawn attentions due to its success in training large models. Consider the following stochastic optimization problem:

$$\min_X f(X) = \mathbb{E}_\xi[F(x,\xi)] \tag{1.1}$$

where the variable $X \in \mathbb{R}^{m\times n}$ is a matrix. Without loss of generality we assume $m \geq n$. Muon optimizer writes:

$$\begin{aligned} &G_t \leftarrow \nabla F(X_t, \xi_t)\\ &B_t \leftarrow \mu B_{t-1} + G_t\\ &O_t \leftarrow \operatorname*{argmin}_O\{\|O - B_t\|_F : O^\top O = I \text{ or } OO^\top = I\}\\ &X_{t+1} \leftarrow X_t - \eta_t O_t \end{aligned} \tag{1.2}$$

where the O step can be equivalently written as $O_t = UV^\top$ where $B_t = U\Sigma V^\top$ is the singular value decomposition.

On the other hand, a closely related formula of (1.2) is the following spectral optimizer:

$$\begin{aligned} &G_t \leftarrow \nabla F(X_t, \xi_t)\\ &B_t \leftarrow \mu B_{t-1} + G_t\\ &\Delta_t \leftarrow \operatorname*{argmin}_\Delta\{\operatorname{tr}(B_t^\top \Delta) + \frac{1}{2\eta_t}\|\Delta\|_2^2\}\\ &X_{t+1} \leftarrow X_t + \Delta_t \end{aligned} \tag{1.3}$$

When $\|\cdot\|_2$ is the matrix 2-norm (spectral norm), i.e. the largest singular value, the solution of the third line of (1.3) is $\Delta_t = -\eta_t\|B_t\|_* UV^\top$, where $B_t = U\Sigma V^\top$ is the singular value decomposition (see Bernstein and Newhouse (2024, Proposition 5)). In this case (1.3) only differs from (1.2) by the norm $\|B_t\|_*$.

*Department of Electrical and Computer Engineering, University of Minnesota. li003755@umn.edu
†Department of Electrical and Computer Engineering, University of Minnesota. mhong@umn.edu

[1] see https://kellerjordan.github.io/posts/muon/.

February 2025

Understanding Gradient Orthogonalization for Deep Learning via Non-Euclidean Trust-Region Optimization

A Preprint

Dmitry Kovalev
Yandex Research
dakovalev1@gmail.com

April 9, 2025

Abstract

Optimization with matrix gradient orthogonalization has recently demonstrated impressive results in the training of deep neural networks (Jordan et al., 2024; Liu et al., 2025). In this paper, we provide a theoretical analysis of this approach. In particular, we show that the orthogonalized gradient method can be seen as a first-order trust-region optimization method, where the trust-region is defined in terms of the matrix spectral norm. Motivated by this observation, we develop the stochastic non-Euclidean trust-region gradient method with momentum, which recovers the Muon optimizer (Jordan et al., 2024) as a special case, along with normalized SGD and signSGD with momentum (Cutkosky and Mehta, 2020; Sun et al., 2023). In addition, we prove state-of-the-art convergence results for the proposed algorithm in a range of scenarios, which involve arbitrary non-Euclidean norms, constrained and composite problems, and non-convex, star-convex, first- and second-order smooth functions. Finally, our theoretical findings provide an explanation for several practical observations, including the practical superiority of Muon compared to the Orthogonal-SGDM algorithm of Tuddenham et al. (2022) and the importance of weight decay in the training of large-scale language models.

1 Introduction

Over the past couple of decades, a substantial amount of optimization research has been dedicated to adaptive gradient optimization algorithms (Duchi et al., 2011; Tieleman, 2012; Kingma and Ba, 2014; Gupta et al., 2018; Reddi et al., 2019), with the primary application being the training of deep neural networks (LeCun et al., 2015). One of the most notable results of this line of research is the development of the AdamW optimizer (Loshchilov and Hutter, 2017; Zhuang et al., 2022), which has become the standard algorithm of choice for training large language models (LLMs) (Achiam et al., 2023; Liu et al., 2024; Grattafiori et al., 2024; Anil et al., 2023). However, recently, Jordan et al. (2024); Liu et al. (2025) have made significant progress in the ambitious task of surpassing AdamW in training LLMs using the idea of neural network optimization with orthogonalized gradients (Tuddenham et al., 2022; Jordan et al., 2024). In our paper, we aim to establish theoretical foundations for this promising research direction. Formally speaking, we consider the following composite optimization problem:

$$\min_{x\in\mathcal{X}}[F(x) = f(x) + R(x)], \tag{1}$$

where $\mathcal{X}$ is a finite-dimensional vector space endowed with the inner product $\langle\cdot,\cdot\rangle\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$, $f(\cdot)\colon \mathcal{X}\to\mathbb{R}$ is a bounded from below and differentiable objective function, and $R(\cdot)\colon \mathcal{X}\to\mathbb{R}\cup\{+\infty\}$ is a proper,[1] closed, and convex regularizer. Our goal is to study the convergence properties of gradient methods for solving problem (1) in the *stochastic non-Euclidean smooth* setting. We provide a formal description of this setting below and justify our interest in this setting in the upcoming Sections 1.1 and 1.2.

Stochastic gradient estimator. We assume access to a stochastic estimator $g(\cdot\,;\xi)\colon \mathcal{X}\to\mathcal{X}$ of the gradient $\nabla f(\cdot)$, where $\xi\sim\mathcal{D}$ is a random variable sampled from a probability distribution $\mathcal{D}$. We assume that the stochastic gradient

[1] Function $R(x)$ is called proper if there exists $x\in\mathcal{X}$ such that $R(x)$ is finite.

March 2025

Scion (2/2025)

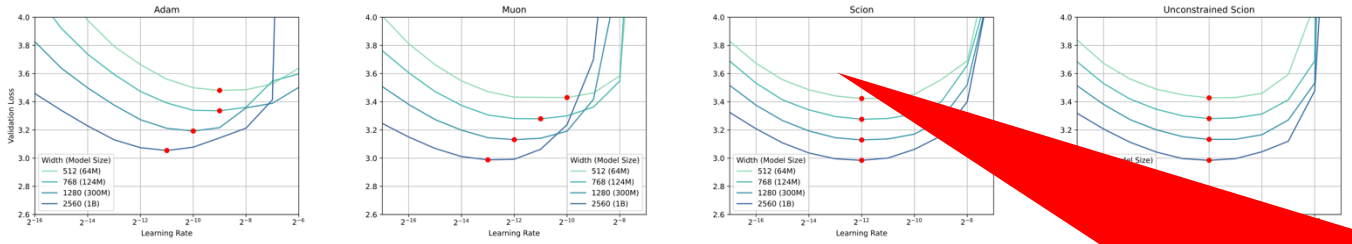


Figure 1. Performance on NanoGPT with between 64M and 1B parameters. The optimal learning rate of SCION is

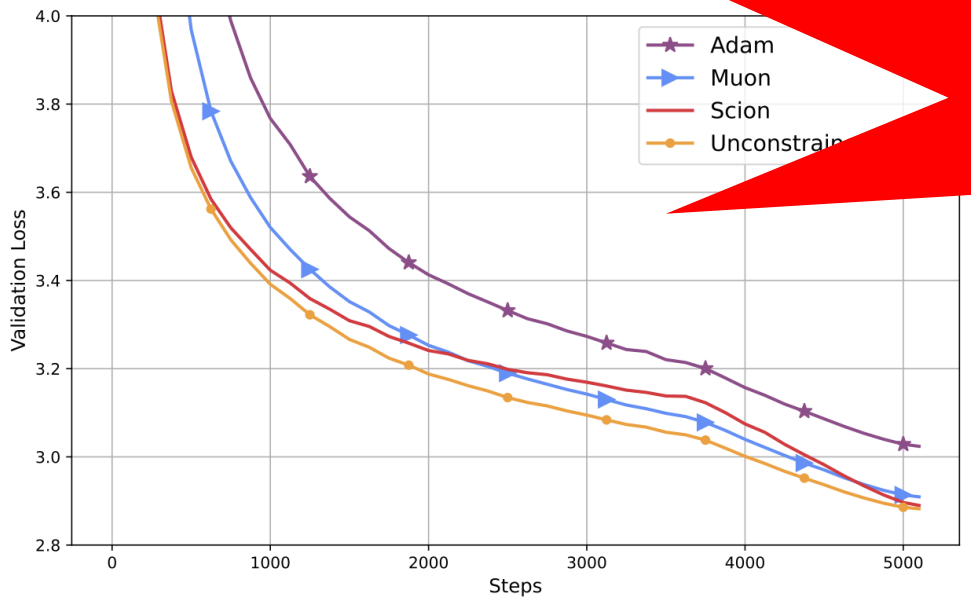


Figure 5. Validation loss curve for NanoGPT 3B.

Training Deep Learning Models with Norm-Constrained LMOs

Thomas Pethick [1] Wanyun Xie [1] Kimon An... [1] Zhenyu Zhu [1] Antonio Silveti-Falls [2]

Abstract

...ing, with adaptive algorithms, such as Adam (Kingma, ...14), dominating the deep learning model training.

...rk, we study optimizati... linear minimiz... We pr... he possible way for ad... which we fo... derlying norm ...eter space. There ...e early work by Carl- ...he stochastic spectral de- ...rms steepest descent in

...SCG)

...momen-

...$\alpha_k \in$ ... steps ... $(0, 1)$

1: **for** $k =$ ... $, n$ **do**
2: Sam... $\xi_k \sim \mathcal{P}$
3: $d^k \leftarrow \alpha_k \nabla f(x^k, \xi_k) + (1 - \alpha_k) d^{k-1}$
4: $x^{k+1} \leftarrow (1 - \gamma_k) x^k + \gamma_k \operatorname{lmo}(d^k)$
5: Choose $\bar{x}^n$ uniformly at random from $\{x^1, \ldots, x^n\}$

Return $\bar{x}^n$

Improves upon Muon, exhibits hyperparameter transfer

Distributed Muon-Style Optimizers

MUON IS SCALABLE FOR LLM TRAINING

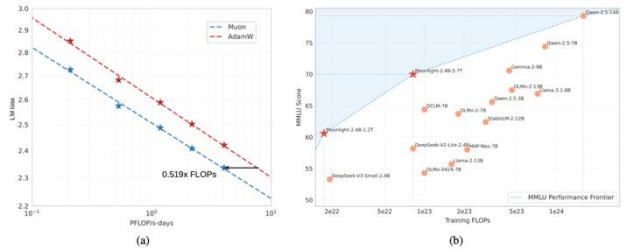
TECHNICAL REPORT

Jingyuan Liu[1] Jianlin Su[1] Xingcheng Yao[2] Zhejun Jiang[1] Guokun Lai[1] Yulun Du[1]
Yidao Qin[1] Weixin Xu[1] Enzhe Lu[1] Junjie Yan[1] Yanru Chen[1] Huabin Zheng[1]
Yibo Liu[1] Shaowei Liu[1] Bohong Yin[1] Weiran He[1] Han Zhu[1] Yuzhi Wang[1]
Jianzhou Wang[1] Mengnan Dong[1] Zheng Zhang[1] Yongsheng Kang[1] Hao Zhang[1]
Xinran Xu[1] Yutao Zhang[1] Yuxin Wu[1] Xinyu Zhou[1] * Zhilin Yang[1]

[1] Moonshot AI [2] UCLA

ABSTRACT

Recently, the Muon optimizer (K. Jordan et al. 2024) based on matrix orthogonalization has demonstrated strong results in training small-scale language models, but the scalability to larger models has not been proven. We identify two crucial techniques for scaling up Muon: (1) adding weight decay and (2) carefully adjusting the per-parameter update scale. These techniques allow Muon to work out-of-the-box on large-scale training without the need of hyper-parameter tuning. Scaling law experiments indicate that Muon achieves $\sim 2\times$ computational efficiency compared to AdamW with compute optimal training. Based on these improvements, we introduce Moonlight, a 3B/16B-parameter Mixture-of-Expert (MoE) model trained with 5.7T tokens using Muon. Our model improves the current Pareto frontier, achieving better performance with much fewer training FLOPs compared to prior models. We open-source our distributed Muon implementation that is memory optimal and communication efficient. We also release the pretrained, instruction-tuned, and intermediate checkpoints to support future research.



(a) (b)

Figure 1: Scaling up with Muon. **(a)** Scaling law experiments comparing Muon and Adam. Muon is $\sim 2\times$ more computational efficient than Adam with compute optimal training. **(b)** The MMLU performance of our Moonlight model optimized with Muon and other comparable models. Moonlight advances the Pareto frontier of performance vs training FLOPs.

*Corresponding author: zhouxinyu@moonshot.cn

February 2025

Dion: Distributed Orthonormalized Updates

Kwangjun Ahn[1] Byron Xu[1] Natalie Abreu[1,2] John Langford[1]
[1]Microsoft Research, AI Frontiers
[2]Harvard University
Correspondence: {kwangjunahn, byronxu}@microsoft.com

Abstract

Recent work has shown that orthonormal matrix updates speed up neural network optimization, improve training stability, and offer better hyperparameter transfer across model sizes. Applying these updates efficiently when model weights and optimizer states are sharded across a large-scale distributed LLM training system remains a major challenge. We introduce Dion (DIstributed OrthoNormalization), a scalable and communication-efficient orthonormalizing optimizer. Dion leverages low-rank approximation and decoupled momentum buffers, eliminating the need for full gradient synchronization while producing numerically equivalent results. It is compatible with simultaneous DDP, FSDP, and TP parallelism, and it computes an orthonormalized update without unsharding a full parameter matrix on any single device. We evaluate Dion on language models from 120M to 3B parameters and find that its benefits improve with increasing model size and batch size.

1 Introduction

Training state-of-the-art AI models consumes millions of GPU-hours, so improvements to the optimizer's *update rule* directly reduce cost and increase iteration speed. In this work, we propose a new update rule for *matrix-valued* parameters—such as the attention and MLP matrices in Transformer models—which constitute the bulk of modern neural network weights.

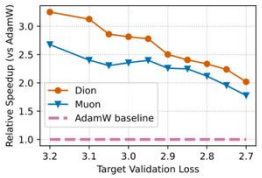


Figure 1: **Speed-up to reach target loss.** We report wall-clock time (relative to AdamW) for training a 3B parameter model to reach each given validation loss. Dion consistently outperforms Muon and demonstrates 2–3× training efficiency over AdamW. See Section 5.4 for experiment details.

The Muon optimizer [Jordan et al., 2024b] has recently shown that orthonormalizing the weight update matrix can give large speedups for neural network training. By conditioning weight updates to induce consistent changes in hidden states [Bernstein, 2025], orthonormalized updates yield faster convergence, greater training stability, and more robust hyperparameter transfer across model scales [Bernstein and Newhouse, 2024a, Large et al., 2024, Pethick et al., 2025]. A large-scale study by Moonshot AI [Liu et al., 2025a] has shown that Muon achieves nearly twice the computational efficiency of AdamW [Loshchilov and Hutter, 2019] when training 16B parameter models. Similarly, Essential AI [Shah et al., 2025] has found improvements with larger batch sizes.

Yet a fundamental obstacle remains. Muon's orthonormalization procedure relies on Newton-Schulz iteration [Bernstein and Newhouse, 2024a,b, Jordan et al., 2024b], involving dense matrix-matrix multiplications that do not respect the ways in which modern LLM training shards parameters across devices. The Moonshot AI reference implementation of Muon [Liu et al., 2025a] therefore executes

Preprint. Under review.

May 2025

MuLoCo: Muon is a practical inner optimizer for DiLoCo

Benjamin Thérien [1 2] Xiaolong Huang [3 2] Irina Rish [1 2] Eugene Belilovsky [3 2]

Abstract

DiLoCo is a powerful framework for training large language models (LLMs) under networking constraints with advantages for increasing parallelism and accelerator utilization in data center settings. Despite significantly reducing communication frequency, however, DiLoCo's communication steps still involve all-reducing a complete copy of the model's parameters. While existing works have explored ways to reduce communication in DiLoCo, the role of error feedback accumulators and the effect of the inner-optimizer on compressibility remain under-explored. In this work, we investigate the effectiveness of standard compression methods—including Top-k sparsification and quantization for reducing the communication overhead of DiLoCo when paired with two local optimizers (AdamW and Muon). Our experiments pre-training decoder-only transformer language models (LMs) reveal that leveraging Muon as the inner optimizer for DiLoCo along with an error-feedback accumulator allows to aggressively compress the communicated delta to 2-bits with next to no performance degradation. Crucially, MuLoCo (Muon inner optimizer DiLoCo) significantly outperforms DiLoCo while communicating 8× less and having identical memory complexity.

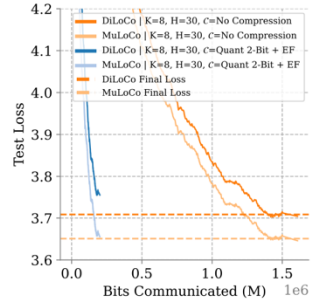


Figure 1: **MuLoCo (Muon inner optimizer) with 2-bit quantization and error feedback outperforms standard AdamW-DiLoCo with 8× less communication while having identical memory complexity.** The figure reports the test loss (y-axis) measured at each communication step during pre-training. The x-axis reports the total number of bits communicated for a 220M parameter transformer LM. We use $K = 8$ workers and $H = 30$ local steps.

1. Introduction

It is now well established that increasing the model and dataset size improves performance of foundation models. Under this paradigm, training state-of-the-art models is infeasible on a single accelerator. Distributed computation across multiple accelerators is, therefore, required. However, standard data parallel training incurs communication costs proportional to the model size at each optimization step. For large models or slow networks, communication can become a significant bottleneck, leading to the development of many communication-efficient training algorithms.

Common approaches to reduce communication costs generally fall into three categories: those that reduce the frequency of communication, decrease the size of data communicated, or combine both approaches. The idea of reducing communication frequency in favor of more local computation was originally popularized in the context of federated learning, where it is known as FedAVG (McMahan et al., 2017). Local SGD (Stich, 2018) is an equivalent algorithm for non-federated settings. More recently, DiLoCo (Douillard et al., 2023) (a variant of Local SGD) has recently gained attention due to its competitive performance with data-parallel pre-training of LLMs and larger effective batch sizes (Charles et al., 2025). Another direction for reducing communication is to compress the com-

[1]DIRO, Université de Montréal, Montréal, Canada [2]Mila – Quebec AI Institute, Montréal, Canada [3]Concordia University, Montréal, Canada. Correspondence to: Benjamin Thérien <benjamin.therien@umontreal.ca>.

May 2025

Part 3
Better Theory for Muon (and Efficient Training of LLMs)

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, P.R.
Gluon: Making Muon & Scion great again! (bridging theory and practice of LMO-based optimizers for LLMs)
arXiv:2505.13416, 2025

Issues with Prior Theory

1. **Global vs layer-wise linear minimization**
   - **Practice:** linear minimization over non-Euclidean balls is performed layer-by layer
   - **Theory:** linear minimization over all parameters

2. **Learning rates (stepsizes)**
   - **Practice:** trial & error tuning
   - **Theory:** impractically tiny stepsizes

Gluon: Making Muon & Scion Great Again!
(Bridging Theory and Practice of LMO-based Optimizers for LLMs)

Artem Riabinin Egor Shulgin Kaja Gruntkowska Peter Richtárik

King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia

Abstract

Recent developments in deep learning optimization have brought about radically new algorithms based on the Linear Minimization Oracle (LMO) framework, such as Muon [12] and Scion [24]. After over a decade of Adam's dominance, these LMO-based methods are emerging as viable replacements, offering several practical advantages such as improved memory efficiency, better hyperparameter transferability, and most importantly, superior empirical performance on large-scale tasks, including LLM training. However, a significant gap remains between their practical use and our current theoretical understanding: prior analyses (1) overlook the layer-wise LMO application of these optimizers in practice, and (2) rely on an unrealistic smoothness assumption, leading to impractically small stepsizes. To address both, we propose a new LMO-based method called Gluon, capturing prior theoretically analyzed methods as special cases, and introduce a new refined generalized smoothness model that captures the layer-wise geometry of neural networks, matches the layer-wise practical implementation of Muon and Scion, and leads to convergence guarantees with strong practical predictive power. Unlike prior results, our theoretical stepsizes closely match the fine-tuned values reported by Pethick et al. [24]. Our experiments with `NanoGPT` and `CNN` confirm that our assumption holds along the optimization trajectory, ultimately closing the gap between theory and practice.

1 Introduction

The success of deep learning models across a wide range of challenging domains is inseparable from the optimization algorithms used to train them. As neural networks have grown deeper and datasets larger, optimization has quietly become one of the most consequential components of modern machine learning (ML). Nowhere is this more evident than in the training of large language models (LLMs), which routinely consume thousands of GPU-hours. Adam [15] (and lately AdamW [22])—being effective, relatively reliable, and widely adopted—has for over a decade served as the default choice for this task. While this reliance has powered much of deep learning's progress, it has also exposed the shortcomings of adaptive moment estimation as a one-size-fits-all solution–namely, sensitivity to learning rate schedules, heavy tuning requirements [28], and poor generalization when not carefully calibrated [31]. However, a shift may now be underway. Recent optimizers, such as Muon [12] and Scion [24], represent a significant departure from Adam-type methods: they forgo the adaptive moment estimation in favor of a geometry-aware approach inspired by Frank-Wolfe

Gluon: The Setup

$$\min_{X \in \mathcal{S}} \{ f(X) := \mathbb{E}_{\xi \sim \mathcal{D}} [f_\xi(X)] \}$$

$$\mathcal{S} := \mathcal{S}_1 \otimes \cdots \otimes \mathcal{S}_p$$

$$X = [X_1, \cdots, X_p]$$

$$X_i \in \mathcal{S}_i := \mathbb{R}^{n_i \times m_i}$$

Gluon: The Algorithm

$$\min_{X \in \mathcal{S}} \{ f(X) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[f_\xi(X) \right] \}$$

Stochastic gradient + momentum

$$M_i^k = \beta^k M_i^{k-1} + (1 - \beta^k) \nabla_i f_{\xi^k}(X^k)$$

Trace inner product on $\mathcal{S}_i := \mathbb{R}^{m_i \times n_i}$

$$\langle X_i, Y_i \rangle := \mathrm{Tr}\left(X_i^\top Y_i \right)$$

Stepsize/radius for layer i

For all layers $i = 1, 2, \ldots, p$ do:

$$X_i^{k+1} = \mathrm{argmin}_{X_i \in \mathcal{S}_i} \left\{ \langle M_i^k, X_i \rangle_{(i)} \;\middle|\; \| X_i - X_i^k \|_{(i)} \leq t_i^k \right\}$$

$$X^{k+1} = [X_1^{k+1}, \ldots, X_i^{k+1}, \ldots, X_p^{k+1}]$$

Trainable parameters belonging to layer i

$$X_i \in \mathcal{S}_i := \mathbb{R}^{n_i \times m_i}$$

Non-Euclidean norm on $\mathcal{S}_i := \mathbb{R}^{m_i \times n_i}$

Spectral norm for $i = 1, \ldots, p-1$

New Smoothness Assumption

New Assumption: Layer-wise L0-L1 Smoothness

$$\min_{X \in \mathcal{S}} \{ f(X) := \mathbb{E}_{\xi \sim \mathcal{D}} [f_\xi(X)] \}$$

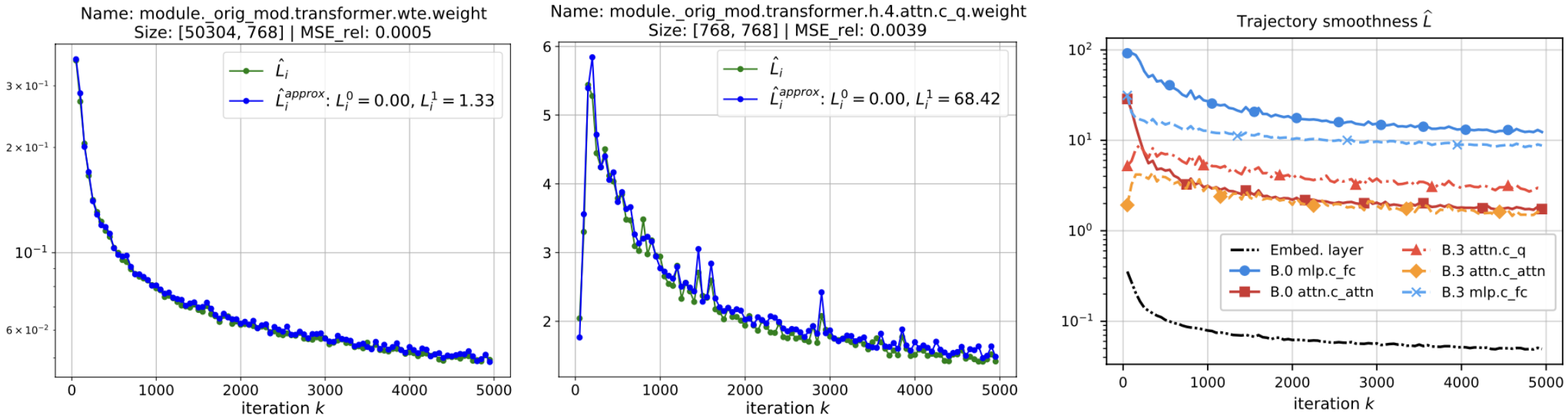
Dual norm for layer i

$$\|G_i\|_{(i)*} := \sup_{\|X_i\|_{(i)} \leq 1} \langle G_i, X_i \rangle_{(i)}$$

$$\frac{\|\nabla_i f(X) - \nabla_i f(Y)\|_{(i)*}}{\|X_i - Y_i\|_{(i)}} \leq L_i^0 + L_i^1 \|\nabla_i f(X)\|_{(i)*}, \quad \forall X, Y \in \mathcal{S}, \quad \forall i \in \{1, \ldots, p\}$$

Two smoothness constants for each layer i

Experimental Validation of Layer-wise L0-L1 Smoothness



(a) Token embedding matrix from the first/last layer.

(b) Self-attention query matrix from the 4th transformer block.

(c) Trajectory smoothness across different blocks (B.i) and layers.

Figure 1: Training `NanoGPT` on `FineWeb` validates our layer-wise (L^0, L^1)-smoothness model.

Gluon Convergence Theory

Gluon: Deterministic Variant

$$\min_{X \in \mathcal{S}} \{ f(X) := \mathbb{E}_{\xi \sim \mathcal{D}} [f_\xi(X)] \}$$

~~Stochastic gradient + momentum~~

~~$M_i^k = \beta^k M_i^{k-1} + (1-\beta^k) \nabla_i f_{\xi^k}(X^k)$~~

Trace inner product on $\mathcal{S}_i := \mathbb{R}^{m_i \times n_i}$

$\langle X_i, Y_i \rangle := \mathrm{Tr}\left(X_i^\top Y_i\right)$

Stepsize/radius for layer i

For all layers $i = 1, 2, \ldots, p$ do:

$$X_i^{k+1} = \mathrm{argmin}_{X_i \in \mathcal{S}_i} \left\{ \langle M_i^k, X_i \rangle_{(i)} \;\middle|\; \|X_i - X_i^k\|_{(i)} \leq t_i^k \right\}$$

$X^{k+1} = [X_1^{k+1}, \cdots, X_i^{k+1}, \cdots, X_p^{k+1}]$

Trainable parameters belonging to layer i

$X_i \in \mathcal{S}_i := \mathbb{R}^{n_i \times m_i}$

Non-Euclidean norm on $\mathcal{S}_i := \mathbb{R}^{m_i \times n_i}$

Spectral norm for $i = 1, \ldots, p-1$

Gluon: Deterministic Variant

$$\min_{X \in \mathcal{S}} \{ f(X) := \mathbb{E}_{\xi \sim \mathcal{D}} [f_\xi(X)] \}$$

Gradient

$$M_i^k = \nabla f_i(X^k)$$

Trace inner product on $\mathcal{S}_i := \mathbb{R}^{m_i \times n_i}$

$$\langle X_i, Y_i \rangle := \mathrm{Tr}\left(X_i^\top Y_i\right)$$

Stepsize/radius for layer i

For all layers $i = 1, 2, \ldots, p$ do:

$$X_i^{k+1} = \mathrm{argmin}_{X_i \in \mathcal{S}_i} \left\{ \langle M_i^k, X_i \rangle_{(i)} \mid \|X_i - X_i^k\|_{(i)} \leq t_i^k \right\}$$

$$X^{k+1} = [X_1^{k+1}, \ldots, X_i^{k+1}, \ldots, X_p^{k+1}]$$

Trainable parameters belonging to layer i

$$X_i \in \mathcal{S}_i := \mathbb{R}^{n_i \times m_i}$$

Non-Euclidean norm on $\mathcal{S}_i := \mathbb{R}^{m_i \times n_i}$

Spectral norm for $i = 1, \ldots, p-1$

Gluon Theory 1

$$M_i^k = \nabla f_i(X^k) \qquad t_i^k = \frac{\|\nabla_i f(X^k)\|_{(i)*}}{L_i^0 + L_i^1 \|\nabla_i f(X^k)\|_{(i)*}}$$

$\Delta^0 = f(X^0) - \inf f$

$\mathrm{MAX} = \max\{L_1^1, \ldots, L_p^1\}$
(maximum of the smoothness constants $\{L_i^1\}_{i=1}^p$)

$$K \geq 2\Delta^0 \cdot \left(\frac{\sum_{i=1}^p L_i^0}{\varepsilon^2} + \frac{\mathrm{MAX}}{\varepsilon} \right)$$

Total number of iterations

$$\min_{k=0,\ldots,K-1} \sum_{i=1}^p \left\| \nabla_i f(X^k) \right\|_{(i)*} \leq \varepsilon$$

Gluon Theory 2

$$M_i^k = \nabla f_i(X^k) \qquad t_i^k = \frac{\|\nabla_i f(X^k)\|_{(i)*}}{L_i^0 + L_i^1 \|\nabla_i f(X^k)\|_{(i)*}}$$

$$\mathrm{HM} = \frac{1}{\frac{1}{p}\sum_{i=1}^{p} \frac{1}{L_i^1}}$$

(harmonic mean of the smoothness constants $\{L_i^1\}_{i=1}^p$)

$\Delta^0 = f(X^0) - \inf f$

$$K \geq 2\Delta^0 \cdot \left(\frac{\sum_{i=1}^{p} \left(\frac{\mathrm{HM}}{L_i^1}\right)^2 L_i^0}{\varepsilon^2} + \frac{\mathrm{HM}}{\varepsilon} \right)$$

Total number of iterations

$$\min_{k=0,\ldots,K-1} \sum_{i=1}^{p} \frac{\mathrm{HM}}{L_i^1} \left\|\nabla_i f(X^k)\right\|_{(i)*} \leq \varepsilon$$

Gluon Theory 3

$$\mathbb{E}_{\xi\sim\mathcal{D}}\left[\|\nabla_i f_\xi(X)-\nabla_i f(X)\|^2_{(i)*}\right]\le\sigma^2$$

$$\forall X\in\mathcal{S},\ \forall i\in\{1,\dots,p\}$$

$$\Delta^0=f(X^0)-\inf f$$

$$\min_{k=0,\dots,K-1}\sum_{i=1}^{p}\frac{1}{12L_i^1}\mathbb{E}\left[\left\|\nabla_i f(X^k)\right\|_{(i)*}\right]\lessapprox\frac{\Delta^0}{K^{1/4}}+\frac{\sum_{i=1}^{p}\left(\frac{L_i^0}{(L_i^1)^2}+\frac{\sigma}{L_i^1}\right)}{K^{1/4}}$$

Total number of iterations

Gluon Theory vs Previous Results

Table 1: Comparison of convergence guarantees for Gluon (Algorithms 1 and 2) to achieve $\min_{k=0,\dots,K-1} \sum_{i=1}^{p} \mathbb{E}[\|\nabla_i f(X^k)\|_{(i)\star}] \leq \varepsilon$, where the $\mathcal{O}(\cdot)$ notation hides logarithmic factors. Notation: K = total number of iterations, (L^0, L^1) = the result holds under layer-wise (L^0, L^1)-smoothness, t_i^k = radius/stepsize, $1-\beta^k$ = momentum.

Result	**Stochastic?**	(L^0, L^1)	**Rate**	**Stepsizes** t_i^k	$1-\beta^k$
[16, Theorem 1]	✗	✗	$\mathcal{O}\left(\frac{1}{K^{1/2}}\right)$	const $\propto \frac{1}{K^{1/2}}$ (b)	—
[16, Theorem 2]	✓	✗	$\mathcal{O}\left(\frac{1}{K^{1/4}}\right)$	const $\propto \frac{1}{K^{3/4}}$ (b)	const $\propto \frac{1}{K^{1/2}}$
[18, Theorem 2.1] (a)	✓	✗	$\mathcal{O}\left(\frac{1}{K^{1/4}}\right)$	const $\propto \frac{1}{K^{3/4}}$ (b)	const $\propto \frac{1}{K^{1/2}}$
[24, Lemma 5.4]	✓	✗	$\mathcal{O}\left(\frac{1}{K^{1/4}}\right)$	const $\propto \frac{1}{K^{3/4}}$ (b)	$\propto \frac{1}{k^{1/2}}$
NEW: Theorem 1	✗	✓	$\mathcal{O}\left(\frac{1}{K^{1/2}}\right)$	Adaptive	—
NEW: Theorem 2	✓	✓	$\mathcal{O}\left(\frac{1}{K^{1/4}}\right)$	$\propto \frac{1}{k^{3/4}}$	$\propto \frac{1}{k^{1/2}}$

(a) Applies only to the Muon/Scion update in (14) with $p = 1$.

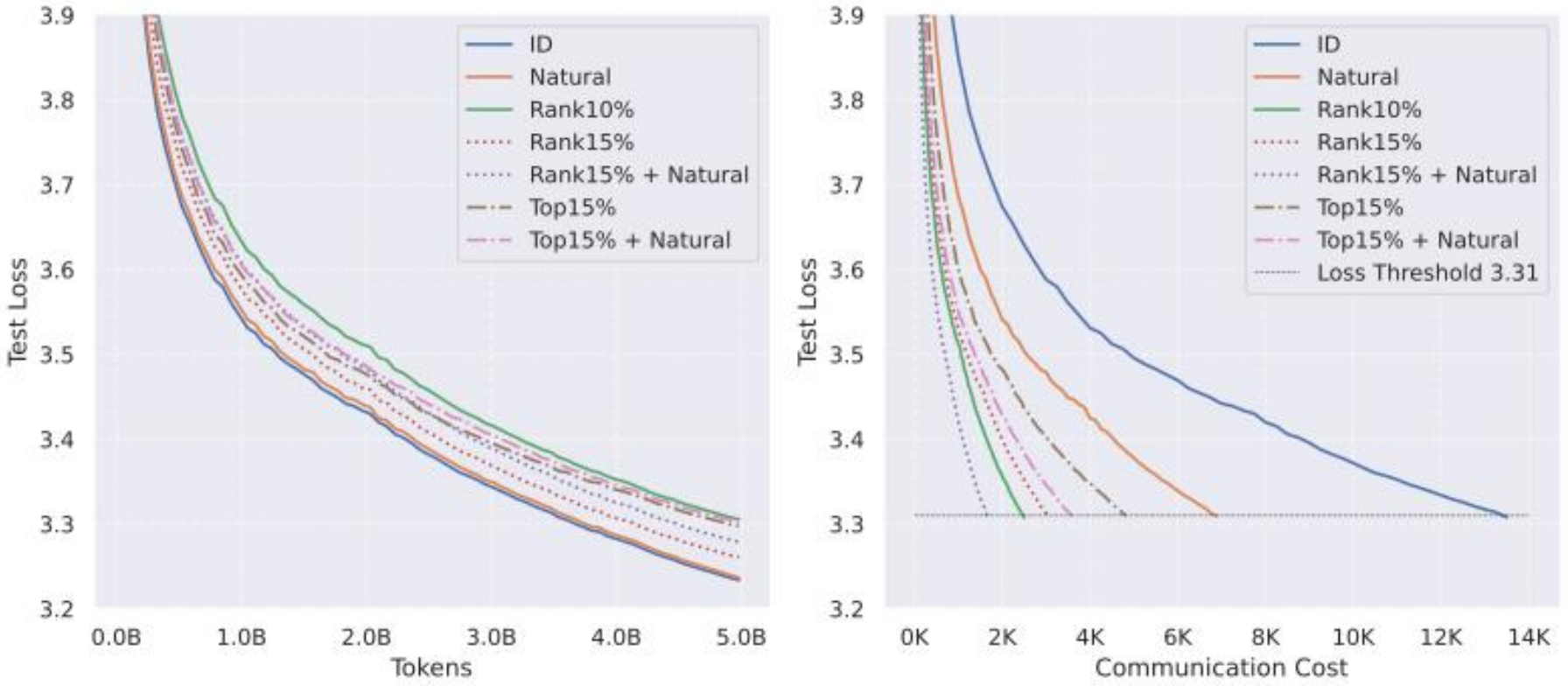
(b) These stepsizes are impractically tiny since they have an inverse dependence on the total number of iterations K.

Part 4
Muon: Compression & Error Feedback

Kaja Gruntkowska, Yassine Maziane, Zheng Qu, P.R.
Drop-Muon: Update less, converge faster
arXiv:2510.02239, 2025

Kaja Gruntkowska, Alexander Gaponov, Zhirayr Tovmasyan, P.R.
Error feedback for Muon and friends
ICLR 2026

Up to 7x Communication Savings on NanoGPT



NanoGPT-124M on FineWeb10B. Left: Test loss vs. # of tokens processed. Right: Test loss vs. # of bytes sent to the server from each worker normalized by model size

Compressor	Relative Cost
ID	1.0000
Natural	0.5000
Rank20%	0.2687
Rank15%	0.2019
Rank15% + Natural	0.1010
Rank10%	0.1335
Rank10% + Natural	0.0667
Rank5%	0.0667
Top20%	0.3625
Top15%	0.2718
Top15% + Natural	0.1969
Top10%	0.1812
Top10% + Natural	0.1312
Top5%	0.0906

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Algorithm 2 EF21-Muon

Parameters: radii $t^k > 0$; momentum parameter $\beta \in (0, 1]$; initial iterate $X^0 \in \mathcal{S}$ (stored on the server); initial iterate shift $W^0 = X^0$ (stored on the server and the workers); initial gradient estimators G_j^0 (stored on the workers); $G^0 = \frac{1}{n}\sum_{j=1}^n G_j^0$ (stored on the server); initial momentum M_j^0 (stored on the workers); worker compressors $\mathcal{C}_j^k$; server compressors $\mathcal{C}^k$

for $k = 0, 1, \ldots, K-1$ **do**

  $X^{k+1} = \mathrm{LMO}_{\mathcal{B}(X^k, t^k)}\left(G^k\right)$ — Take LMO-type step

  $S^k = \mathcal{C}^k(X^{k+1} - W^k)$ — Compress shifted model on the server

  $W^{k+1} = W^k + S^k$ — Update model shift

  Broadcast S^k to all workers

  for $j = 1, \ldots, n$ **in parallel do**

    $W^{k+1} = W^k + S^k$ — Update model shift

    $M_j^{k+1} = (1-\beta)M_j^k + \beta \nabla f_j(W^{k+1}; \xi_j^{k+1})$ — Compute momentum

    $R_j^{k+1} = \mathcal{C}_j^k(M_j^{k+1} - G_j^k)$ — Compress shifted gradient

    $G_j^{k+1} = G_j^k + R_j^{k+1}$

    Broadcast R_j^{k+1} to the server

  end for

  $G^{k+1} = \frac{1}{n}\sum_{j=1}^n G_j^{k+1} = G^k + \frac{1}{n}\sum_{j=1}^n R_j^{k+1}$ — Compute gradient estimator

end for

Assumptions

Assumptions

A1 *There exist* $f^\star \in \mathbb{R}$ *such that* $f(X) \geq f^\star$ *for all* $X \in \mathcal{S}$.

A2 *For all* $j \in [n]$, *there exist* $f_j^\star \in \mathbb{R}$ *such that* $f_j(X) \geq f_j^\star$ *for all* $X \in \mathcal{S}$.

Assumptions

A1 *There exist* $f^\star \in \mathbb{R}$ *such that* $f(X) \geq f^\star$ *for all* $X \in \mathcal{S}$.

A2 *For all* $j \in [n]$, *there exist* $f_j^\star \in \mathbb{R}$ *such that* $f_j(X) \geq f_j^\star$ *for all* $X \in \mathcal{S}$.

A3(a) $\|\nabla f(X) - \nabla f(Y)\|_\star \leq L\|X - Y\|$

A3(b) $\|\nabla f(X) - \nabla f(Y)\|_\star \leq \left(L^0 + L^1\|\nabla f(X)\|_\star\right)\|X - Y\|$

Assumptions

A1 *There exist* $f^\star \in \mathbb{R}$ *such that* $f(X) \geq f^\star$ *for all* $X \in \mathcal{S}$.

A2 *For all* $j \in [n]$, *there exist* $f_j^\star \in \mathbb{R}$ *such that* $f_j(X) \geq f_j^\star$ *for all* $X \in \mathcal{S}$.

A3(a) $\|\nabla f(X) - \nabla f(Y)\|_\star \leq L\|X - Y\|$

A3(b) $\|\nabla f(X) - \nabla f(Y)\|_\star \leq \left(L^0 + L^1\|\nabla f(X)\|_\star\right)\|X - Y\|$

A4 $\mathbb{E}_{\xi_j \sim \mathcal{D}_j}\left[\nabla f_j(X;\xi_j)\right] = \nabla f_j(X)$ $\mathbb{E}_{\xi_j \sim \mathcal{D}_j}\left[\|\nabla f_j(X;\xi_j) - \nabla f_j(X)\|_2^2\right] \leq \sigma^2$

Convergence Theory

Let A1, A3(a) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $X^0 = W^0, G_j^0 = M_j^0 = \nabla f_j(X^0; \xi_j^0)$ and run with

$\mathcal{C}^k \in \mathbb{B}(\alpha_P), \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $\gamma = \left(2\sqrt{\zeta} + 2L\right)^{-1}$, where $\zeta = \frac{\bar{\rho}^2}{\underline{\rho}^2}\left(\frac{12}{\beta^2}L^2 + \frac{24(\beta+2)}{\alpha_P^2}L^2 + \frac{36(\beta^2+4)}{\alpha_D^2}\tilde{L}^2 + \frac{144\beta^2(2\beta+5)}{\alpha_P^2\alpha_D^2}\tilde{L}^2\right)$ and

$\beta = \min\left\{1, \left(\frac{\delta^0 L n}{\underline{\rho}^2\sigma^2 K}\right)^{1/2}, \left(\frac{\delta^0 L \alpha_D}{\underline{\rho}^2\sigma^2 K}\right)^{2/3}, \left(\frac{\delta^0 L \alpha_D^2}{\underline{\rho}^2\sigma^2 K}\right)^{1/4}\right\}$. Then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(X^k)\|_\star^2\right] = \mathcal{O}\left(\frac{\delta^0\bar{\rho}^2\tilde{L}^0}{\underline{\rho}^2\alpha_P\alpha_D K} + \left(\frac{\delta^0\bar{\rho}^4\sigma^2 L}{\underline{\rho}^2 n K}\right)^{1/2} + \left(\frac{\delta^0\bar{\rho}^3\sigma L}{\underline{\rho}^2\sqrt{\alpha_D}K}\right)^{2/3} + \left(\frac{\delta^0\underline{\rho}^{8/3}\sigma^{2/3}L}{\bar{\rho}^2\alpha_D^{2/3}K}\right)^{3/4}\right)$$

Convergence Theory

Let A1, A3(a) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $X^0 = W^0, G_j^0 = M_j^0 = \nabla f_j(X^0; \xi_j^0)$ and run with

$\mathcal{C}^k \in \mathbb{B}(\alpha_P), \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $\gamma = \left(2\sqrt{\zeta} + 2L\right)^{-1}$, where $\zeta = \frac{\bar{\rho}^2}{\underline{\rho}^2}\left(\frac{12}{\beta^2}L^2 + \frac{24(\beta+2)}{\alpha_P^2}L^2 + \frac{36(\beta^2+4)}{\alpha_D^2}\tilde{L}^2 + \frac{144\beta^2(2\beta+5)}{\alpha_P^2\alpha_D^2}\tilde{L}^2\right)$ and

$\beta = \min\left\{1, \left(\frac{\delta^0 L n}{\underline{\rho}^2\sigma^2 K}\right)^{1/2}, \left(\frac{\delta^0 L \alpha_D}{\underline{\rho}^2\sigma^2 K}\right)^{2/3}, \left(\frac{\delta^0 L \alpha_D^2}{\underline{\rho}^2\sigma^2 K}\right)^{1/4}\right\}$. Then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(X^k)\|_\star^2\right] = \mathcal{O}\left(\frac{\delta^0\bar{\rho}^2\tilde{L}^0}{\underline{\rho}^2\alpha_P\alpha_D K} + \left(\frac{\delta^0\bar{\rho}^4\sigma^2 L}{\underline{\rho}^2 n K}\right)^{1/2} + \left(\frac{\delta^0\bar{\rho}^3\sigma L}{\underline{\rho}^2\sqrt{\alpha_D}K}\right)^{2/3} + \left(\frac{\delta^0\underline{\rho}^{8/3}\sigma^{2/3}L}{\bar{\rho}^2\alpha_D^{2/3}K}\right)^{3/4}\right)$$

$f(X^0) - f^\star$

Convergence Theory

Let A1, A3(a) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $X^0 = W^0, G_j^0 = M_j^0 = \nabla f_j(X^0; \xi_j^0)$ and run with

$\mathcal{C}^k \in \mathbb{B}(\alpha_P), \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $\gamma = \left(2\sqrt{\zeta} + 2L\right)^{-1}$, where $\zeta = \frac{\bar{\rho}^2}{\underline{\rho}^2}\left(\frac{12}{\beta^2}L^2 + \frac{24(\beta+2)}{\alpha_P^2}L^2 + \frac{36(\beta^2+4)}{\alpha_D^2}\tilde{L}^2 + \frac{144\beta^2(2\beta+5)}{\alpha_P^2\alpha_D^2}\tilde{L}^2\right)$ and

$\beta = \min\left\{1, \left(\frac{\delta^0 L n}{\underline{\rho}^2\sigma^2 K}\right)^{1/2}, \left(\frac{\delta^0 L \alpha_D}{\underline{\rho}^2\sigma^2 K}\right)^{2/3}, \left(\frac{\delta^0 L \alpha_D^2}{\underline{\rho}^2\sigma^2 K}\right)^{1/4}\right\}$. Then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(X^k)\|_\star^2\right] = \mathcal{O}\left(\frac{\delta^0\bar{\rho}^2\tilde{L}^0}{\underline{\rho}^2\alpha_P\alpha_D K} + \left(\frac{\delta^0\bar{\rho}^4\sigma^2 L}{\underline{\rho}^2 n K}\right)^{1/2} + \left(\frac{\delta^0\bar{\rho}^3\sigma L}{\underline{\rho}^2\sqrt{\alpha_D}K}\right)^{2/3} + \left(\frac{\delta^0\underline{\rho}^{8/3}\sigma^{2/3}L}{\bar{\rho}^2\alpha_D^{2/3}K}\right)^{3/4}\right)$$

Convergence Theory

Let A1, A3(a) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $X^0 = W^0, G_j^0 = M_j^0 = \nabla f_j(X^0; \xi_j^0)$ and run with

$\mathcal{C}^k \in \mathbb{B}(\alpha_P), \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $\gamma = \left(2\sqrt{\zeta} + 2L\right)^{-1}$, where $\zeta = \frac{\bar{\rho}^2}{\underline{\rho}^2}\left(\frac{12}{\beta^2}L^2 + \frac{24(\beta+2)}{\alpha_P^2}L^2 + \frac{36(\beta^2+4)}{\alpha_D^2}\tilde{L}^2 + \frac{144\beta^2(2\beta+5)}{\alpha_P^2\alpha_D^2}\tilde{L}^2\right)$ and

$\beta = \min\left\{1, \left(\frac{\delta^0 L n}{\underline{\rho}^2\sigma^2 K}\right)^{1/2}, \left(\frac{\delta^0 L \alpha_D}{\underline{\rho}^2\sigma^2 K}\right)^{2/3}, \left(\frac{\delta^0 L \alpha_D^2}{\underline{\rho}^2\sigma^2 K}\right)^{1/4}\right\}$. Then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(X^k)\|_\star^2\right] = \mathcal{O}\left(\frac{\delta^0\bar{\rho}^2\tilde{L}^0}{\underline{\rho}^2\alpha_P\alpha_D K} + \left(\frac{\delta^0\bar{\rho}^4\sigma^2 L}{\underline{\rho}^2 n K}\right)^{1/2} + \left(\frac{\delta^0\bar{\rho}^3\sigma L}{\underline{\rho}^2\sqrt{\alpha_D}K}\right)^{2/3} + \left(\frac{\delta^0\underline{\rho}^{8/3}\sigma^{2/3}L}{\bar{\rho}^2\alpha_D^{2/3}K}\right)^{3/4}\right)$$

Convergence Theory

Let A1, A3(a) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $X^0 = W^0, G_j^0 = M_j^0 = \nabla f_j(X^0; \xi_j^0)$ and run with

$\mathcal{C}^k \in \mathbb{B}(\alpha_P), \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $\gamma = \left(2\sqrt{\zeta} + 2L\right)^{-1}$, where $\zeta = \frac{\bar{\rho}^2}{\underline{\rho}^2}\left(\frac{12}{\beta^2}L^2 + \frac{24(\beta+2)}{\alpha_P^2}L^2 + \frac{36(\beta^2+4)}{\alpha_D^2}\tilde{L}^2 + \frac{144\beta^2(2\beta+5)}{\alpha_P^2\alpha_D^2}\tilde{L}^2\right)$ and

$\beta = \min\left\{1, \left(\frac{\delta^0 L n}{\underline{\rho}^2\sigma^2 K}\right)^{1/2}, \left(\frac{\delta^0 L \alpha_D}{\underline{\rho}^2\sigma^2 K}\right)^{2/3}, \left(\frac{\delta^0 L \alpha_D^2}{\underline{\rho}^2\sigma^2 K}\right)^{1/4}\right\}$. Then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(X^k)\|_\star^2\right] = \mathcal{O}\left(\frac{\delta^0\bar{\rho}^2\tilde{L}^0}{\underline{\rho}^2\alpha_P\alpha_D K} + \left(\frac{\delta^0\bar{\rho}^4\sigma^2 L}{\underline{\rho}^2 n K}\right)^{1/2} + \left(\frac{\delta^0\bar{\rho}^3\sigma L}{\underline{\rho}^2\sqrt{\alpha_D}K}\right)^{2/3} + \left(\frac{\delta^0\underline{\rho}^{8/3}\sigma^{2/3}L}{\bar{\rho}^2\alpha_D^{2/3}K}\right)^{3/4}\right)$$

Convergence Theory

Let A1, A3(a) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $X^0 = W^0, G_j^0 = M_j^0 = \nabla f_j(X^0; \xi_j^0)$ and run with

$\mathcal{C}^k \in \mathbb{B}(\alpha_P), \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $\gamma = \left(2\sqrt{\zeta} + 2L\right)^{-1}$, where $\zeta = \frac{\bar{\rho}^2}{\underline{\rho}^2}\left(\frac{12}{\beta^2}L^2 + \frac{24(\beta+2)}{\alpha_P^2}L^2 + \frac{36(\beta^2+4)}{\alpha_D^2}\tilde{L}^2 + \frac{144\beta^2(2\beta+5)}{\alpha_P^2\alpha_D^2}\tilde{L}^2\right)$ and

$\beta = \min\left\{1, \left(\frac{\delta^0 L n}{\underline{\rho}^2\sigma^2 K}\right)^{1/2}, \left(\frac{\delta^0 L \alpha_D}{\underline{\rho}^2\sigma^2 K}\right)^{2/3}, \left(\frac{\delta^0 L \alpha_D^2}{\underline{\rho}^2\sigma^2 K}\right)^{1/4}\right\}$. Then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(X^k)\|_\star^2\right] = \mathcal{O}\left(\frac{\delta^0\bar{\rho}^2\tilde{L}^0}{\underline{\rho}^2\alpha_P\alpha_D K} + \left(\frac{\delta^0\bar{\rho}^4\sigma^2 L}{\underline{\rho}^2 n K}\right)^{1/2} + \left(\frac{\delta^0\bar{\rho}^3\sigma L}{\underline{\rho}^2\sqrt{\alpha_D}K}\right)^{2/3} + \left(\frac{\delta^0\underline{\rho}^{8/3}\sigma^{2/3}L}{\bar{\rho}^2\alpha_D^{2/3}K}\right)^{3/4}\right)$$

Convergence Theory

Let A1, A2, A3(b) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $M_j^0 = \nabla f_j(X^0;\xi_j^0), G_j^0 = \mathcal{C}_j^0(\nabla f_j(X^0;\xi_j^0))$ and run with $\mathcal{C}^k \equiv \mathcal{I}, \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $t = \dfrac{\eta}{(K+1)^{3/4}}$, where $\eta^2 \leq \min\left\{\dfrac{(K+1)^{1/2}}{6(L^1)^2}, \dfrac{(1-\sqrt{1-\alpha_D})\underline{\rho}}{24(K+1)^{1/2}\sqrt{1-\alpha_D}\bar{\rho}(L^1_{\max})^2}, \dfrac{\underline{\rho}}{24\bar{\rho}(L^1_{\max})^2}, 1\right\}$ and $\beta = (K+1)^{-1/2}$. Then

$$\min_{k=0,\ldots,K} \mathbb{E}\left[\|\nabla f(X^k)\|_\star\right] \leq \frac{3\left(f(X^0)-f^\star\right)}{\eta(K+1)^{1/4}} + \frac{\eta L^0}{(K+1)^{3/4}} + \frac{16\sqrt{1-\alpha_D}\bar{\rho}\sigma}{(1-\sqrt{1-\alpha_D})(K+1)^{1/2}} + \frac{8\bar{\rho}\sigma}{\sqrt{n}(K+1)^{1/4}}$$
$$+\frac{\eta\bar{\rho}}{\underline{\rho}}\left(\frac{8}{(K+1)^{1/4}} + \frac{8\sqrt{1-\alpha_D}}{(1-\sqrt{1-\alpha_D})(K+1)^{3/4}}\right)\left(\frac{1}{n}\sum_{j=1}^n (L_j^1)^2\left(f^\star - f_j^\star\right) + \bar{L}^0\right)$$

Convergence Theory

Let A1, A2, A3(b) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $M_j^0 = \nabla f_j(X^0;\xi_j^0), G_j^0 = \mathcal{C}_j^0(\nabla f_j(X^0;\xi_j^0))$ and run with $\mathcal{C}^k \equiv \mathcal{I}, \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $t = \dfrac{\eta}{(K+1)^{3/4}}$, where $\eta^2 \leq \min\left\{\dfrac{(K+1)^{1/2}}{6(L^1)^2}, \dfrac{(1-\sqrt{1-\alpha_D})\underline{\rho}}{24(K+1)^{1/2}\sqrt{1-\alpha_D}\bar{\rho}(L^1_{\max})^2}, \dfrac{\underline{\rho}}{24\bar{\rho}(L^1_{\max})^2}, 1\right\}$ and $\beta = (K+1)^{-1/2}$. Then

$$\min_{k=0,\ldots,K} \mathbb{E}\left[\|\nabla f(X^k)\|_\star\right] \leq \frac{3\left(f(X^0)-f^\star\right)}{\eta(K+1)^{1/4}} + \frac{\eta L^0}{(K+1)^{3/4}} + \frac{16\sqrt{1-\alpha_D}\bar{\rho}\sigma}{(1-\sqrt{1-\alpha_D})(K+1)^{1/2}} + \frac{8\bar{\rho}\sigma}{\sqrt{n}(K+1)^{1/4}}$$
$$+\frac{\eta\bar{\rho}}{\underline{\rho}}\left(\frac{8}{(K+1)^{1/4}} + \frac{8\sqrt{1-\alpha_D}}{(1-\sqrt{1-\alpha_D})(K+1)^{3/4}}\right)\left(\frac{1}{n}\sum_{j=1}^n (L_j^1)^2\left(f^\star - f_j^\star\right) + \bar{L}^0\right)$$

Convergence Theory

Let A1, A2, A3(b) and A4 hold. Let $\{X^k\}_{k=0}^{K-1}$ be the iterates of EF21-Muon initialized with $M_j^0 = \nabla f_j(X^0;\xi_j^0), G_j^0 = \mathcal{C}_j^0(\nabla f_j(X^0;\xi_j^0))$ and run with $\mathcal{C}^k \equiv \mathcal{I}, \mathcal{C}_j^k \in \mathbb{B}_2(\alpha_D)$, $t = \dfrac{\eta}{(K+1)^{3/4}}$, where $\eta^2 \leq \min\left\{\dfrac{(K+1)^{1/2}}{6(L^1)^2}, \dfrac{(1-\sqrt{1-\alpha_D})\underline{\rho}}{24(K+1)^{1/2}\sqrt{1-\alpha_D}\bar{\rho}(L^1_{\max})^2}, \dfrac{\underline{\rho}}{24\bar{\rho}(L^1_{\max})^2}, 1\right\}$ and $\beta = (K+1)^{-1/2}$. Then

$$\min_{k=0,\ldots,K} \mathbb{E}\left[\|\nabla f(X^k)\|_\star\right] \leq \frac{3\left(f(X^0)-f^\star\right)}{\eta(K+1)^{1/4}} + \frac{\eta L^0}{(K+1)^{3/4}} + \frac{16\sqrt{1-\alpha_D}\bar{\rho}\sigma}{(1-\sqrt{1-\alpha_D})(K+1)^{1/2}} + \frac{8\bar{\rho}\sigma}{\sqrt{n}(K+1)^{1/4}}$$
$$+\frac{\eta\bar{\rho}}{\underline{\rho}}\left(\frac{8}{(K+1)^{1/4}} + \frac{8\sqrt{1-\alpha_D}}{(1-\sqrt{1-\alpha_D})(K+1)^{3/4}}\right)\left(\frac{1}{n}\sum_{j=1}^n (L_j^1)^2\left(f^\star - f_j^\star\right) + \bar{L}^0\right)$$

The End