Training Deep Learning Models with Norm-Constrained LMOs

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, Volkan Cevher

LPSM Séminaire de Statistique 2025

Plan

- **Motivation**
- Algorithms using LMOs
- Our algorithms: uSCG/SCG
- Applications and empirical results

Template Optimization Problem

We will focus on solving

$$\min_{x \in \mathcal{X}} f(x) := \mathbb{E}_\xi[f(x, \xi)]$$

where

- $\mathcal{X}$ is either $\mathbb{R}^d$ (unconstrained) or $\mathcal{D}$ (constrained), with

  $$\mathcal{D} := \{x : \|x\| \leq \rho\}.$$

- $\mathbb{E}_\xi[f(\cdot, \xi)]$ is Lipschitz-smooth with respect to some norm.
- We have access to a stochastic first-order oracle $\nabla f(\cdot, \xi)$ which is unbiased

  $$\mathbb{E}_\xi[\nabla f(\cdot, \xi)] = \nabla f(\cdot)$$

  and has bounded variance

  $$\mathbb{E}_\xi[\|\nabla f(\cdot, \xi) - \nabla f(\cdot)\|_2^2] \leq \sigma^2.$$

Shortcomings of SGD

Stochastic Gradient Descent (SGD):

Input: $x^0 \in \mathcal{X}$, stepsizes $\{\gamma_k\}$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $x^{k+1} = x^k - \gamma_k g^k$
Output: x^n

- SGD inherently uses Euclidean geometry:

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|_2^2$$

- This geometry is not representative of many problems (e.g., neural networks)

Two major approaches to address these limitations:

1. *On-the-fly adaptation*: Methods that adapt during training (AdaGrad, RMSprop, Adam, AdamW)
2. *A priori adaptation*: Methods designed with problem-specific geometry in mind (Bregman methods, Riemannian optimization, our approach)

On-the-fly adaptation

Today, we have many algorithms that adapt to the geometry of the problem *on-the-fly*.
AdaGrad:

Input: $x^0 \in \mathcal{X}$, stepsize γ, $\epsilon > 0$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $G_k = G_{k-1} + (g^k)^2$
  $x^{k+1} = x^k - \frac{\gamma}{\sqrt{G_k + \epsilon}} \odot g^k$
Output: x^n

We can equivalently write this as

$$x^{k+1} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|_{2, G_k}^2$$

where $\|x\|_{2,G_k}^2 = \langle x, G_k x \rangle$ is the squared Mahalanobis norm.

On-the-fly adaptation

Today, we have many algorithms that adapt to the geometry of the problem *on-the-fly*.
RMSprop:

Input: $x^0 \in \mathcal{X}$, stepsize γ, $\epsilon > 0$, momentum $\beta \in (0,1)$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $G_k = \beta G_{k-1} + (1-\beta)(g^k)^2$
  $x^{k+1} = x^k - \frac{\gamma}{\sqrt{G_k + \epsilon}} \odot g^k$
Output: x^n

We can equivalently write this as

$$x^{k+1} \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle g^k, x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|^2_{2, G_k}$$

where $\|x\|^2_{2,G_k} = \langle x, G_k x \rangle$ is the squared Mahalanobis norm.

On-the-fly adaptation

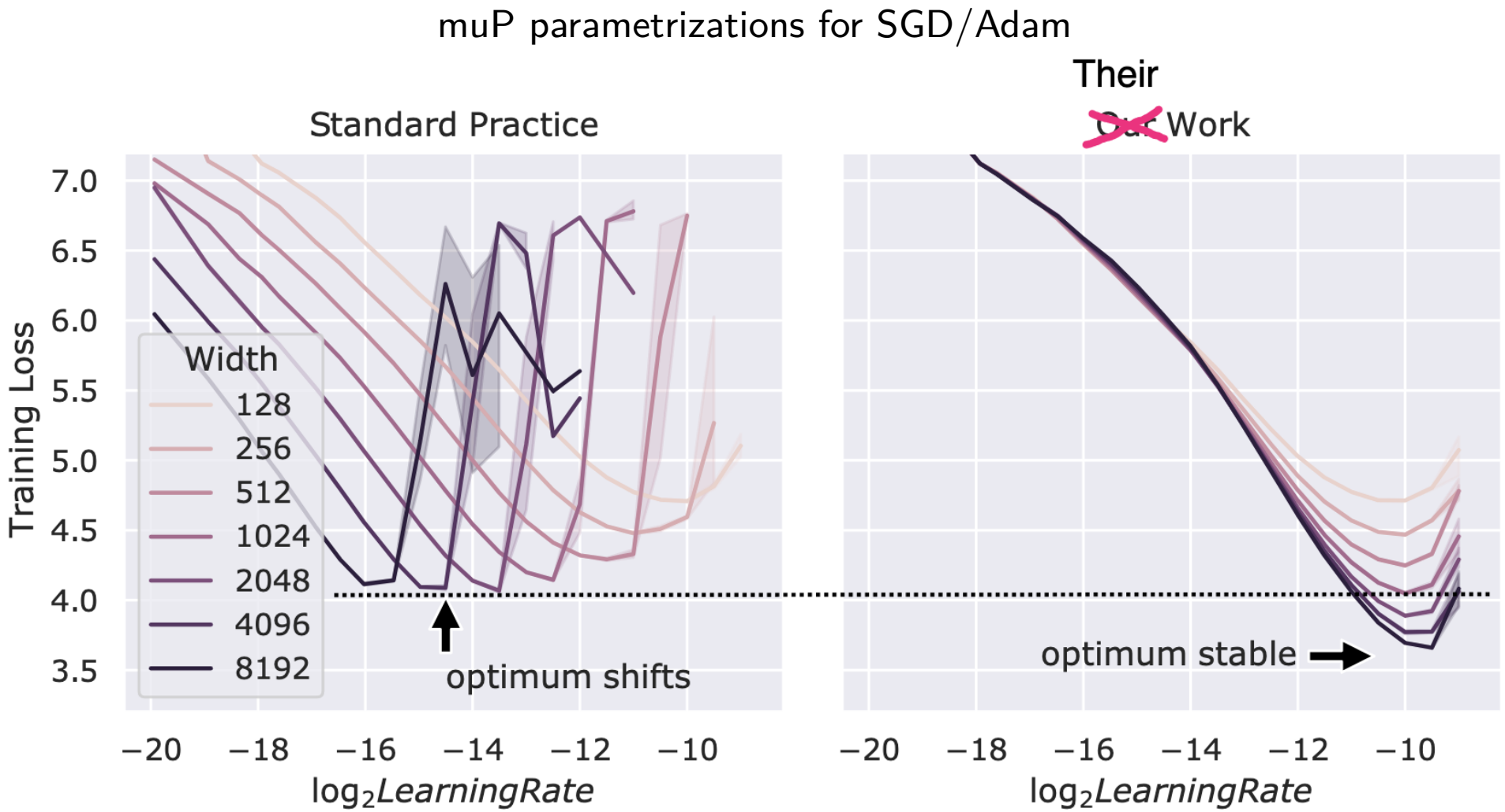
Today, we have many algorithms that adapt to the geometry of the problem *on-the-fly*.
Adam:

Input: $x^0 \in \mathcal{X}$, stepsize γ, $\epsilon > 0$, momentum β_1, β_2, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$
  $v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$
  $\hat{m}^k = \frac{m^k}{1-\beta_1^k}$
  $\hat{v}^k = \frac{v^k}{1-\beta_2^k}$
  $x^{k+1} = x^k - \frac{\gamma}{\sqrt{\hat{v}^k}+\epsilon} \odot \hat{m}^k$
Output: x^n

Simplified idea of Adam: RMSprop + 2nd moment estimation.
These methods are all still essentially Euclidean; their adaptivity comes from a Mahalanobis norm.

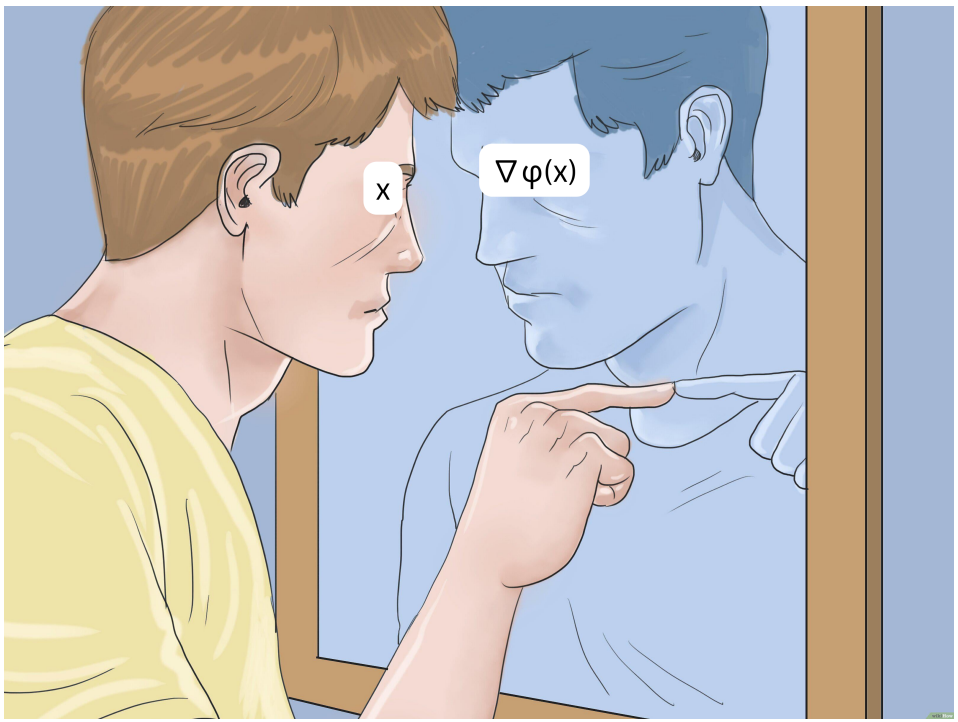
A Priori Adaptation

We can also try to design our optimizers to be adapted to the problem class.



A Priori Adaptation

We can also try to design our optimizers to be adapted to the problem class.

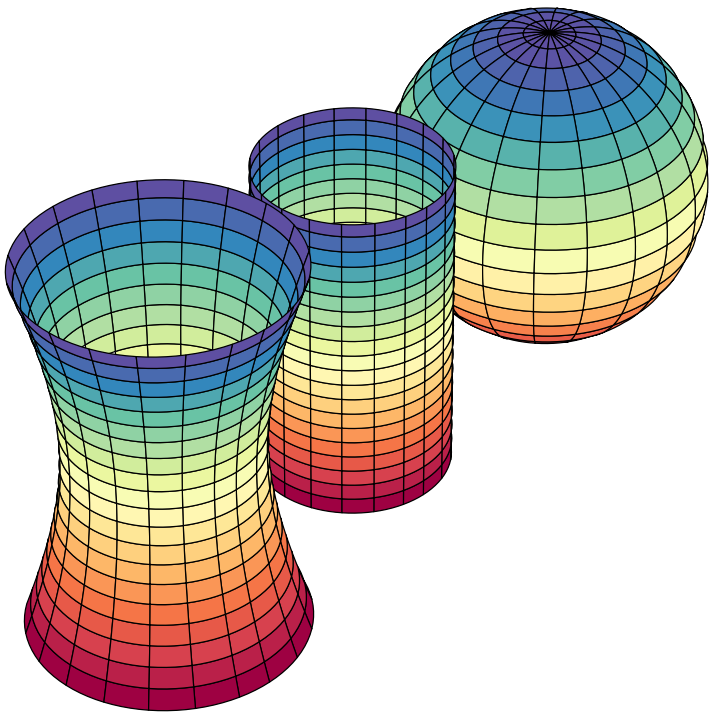


Mirror descent, NoLips, Bregman methods

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathcal{X}} \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{\gamma} D_\phi(x, x^k)$$
$$= (\nabla \phi)^{-1}(\nabla \phi(x^k) - \gamma \nabla f(x^k)).$$

A Priori Adaptation

We can also try to design our optimizers to be adapted to the problem class.



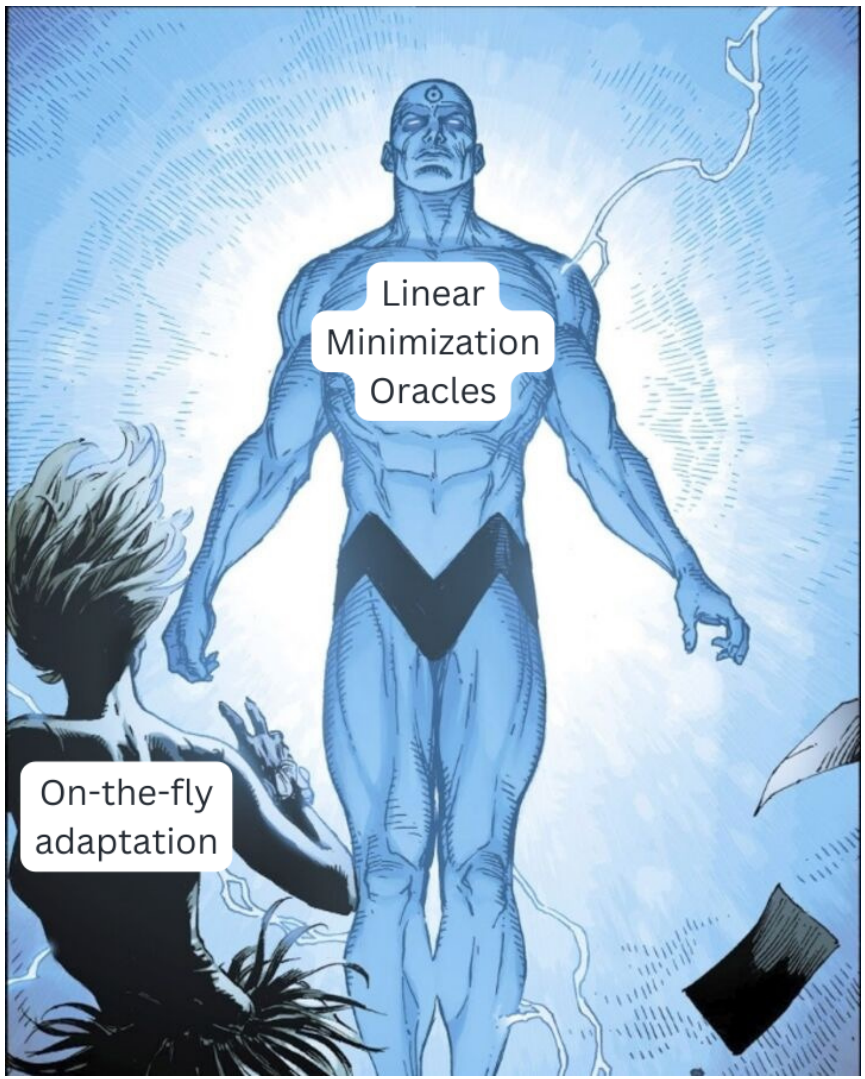
Riemannian Optimization

$$x^{k+1} = \exp_{x^k}(-\gamma \nabla_{\mathcal{M}} f(x^k))$$

where $\mathcal{M}$ is a Riemannian manifold with metric $d(\cdot, \cdot)$.

A Priori Adaptation

We can also try to design our optimizers to be adapted to the problem class.



Noneuclidean Linearizations

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2\gamma_k} \|x - x^k\|^2$$

or

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \nabla f(x^k), x - x^k \rangle + \iota_{\mathcal{D}}(x - x^k)$$

This is what we will focus on today!

Plan

- Motivation
- **Algorithms using LMOs**
- Our algorithms: uSCG/SCG
- Applications and empirical results

Steepest Descent via Dual Norms

In steepest descent, we seek the direction of maximum decrease in our norm. Since f is Lipschitz-smooth, we can use a quadratic approximation of f around x:

$$f(x + \Delta x) \approx f(x) + \langle \nabla f(x), \Delta x \rangle + \frac{L}{2} \|\Delta x\|^2$$

Steepest Descent via Dual Norms

In steepest descent, we seek the direction of maximum decrease in our norm. Since f is Lipschitz-smooth, we can use a quadratic approximation of f around x:

$$f(x + \Delta x) \approx f(x) + \langle \nabla f(x), \Delta x \rangle + \frac{L}{2} \|\Delta x\|^2$$

Minimizing this with respect to Δx gives a closed-form solution using the dual norm of $\nabla f(x)$,

$$\Delta x = -\frac{1}{L} \sharp(\nabla f(x)) := \frac{1}{L} \|\nabla f(x)\|_* \, \mathrm{lmo}(\nabla f(x))$$

where $\sharp(\cdot)$ is the *sharp operator* and lmo is the *linear minimization oracle*:

$$\mathrm{lmo}(\nabla f(x)) \in \operatorname*{argmin}_{s \in \mathcal{D}} \langle \nabla f(x), s \rangle$$

and $\mathcal{D}$ is the unit-ball for the norm $\|\cdot\|$.

Steepest Descent via Dual Norms

In steepest descent, we seek the direction of maximum decrease in our norm. Since f is Lipschitz-smooth, we can use a quadratic approximation of f around x:

$$f(x + \Delta x) \approx f(x) + \langle \nabla f(x), \Delta x \rangle + \frac{L}{2} \|\Delta x\|^2$$

Minimizing this with respect to Δx gives a closed-form solution using the dual norm of $\nabla f(x)$,

$$\Delta x = -\frac{1}{L} \sharp(\nabla f(x)) := \frac{1}{L} \|\nabla f(x)\|_* \, \mathrm{lmo}(\nabla f(x))$$

where $\sharp(\cdot)$ is the *sharp operator* and lmo is the *linear minimization oracle*:

$$\mathrm{lmo}(\nabla f(x)) \in \operatorname*{argmin}_{s \in \mathcal{D}} \langle \nabla f(x), s \rangle$$

and $\mathcal{D}$ is the unit-ball for the norm $\|\cdot\|$.

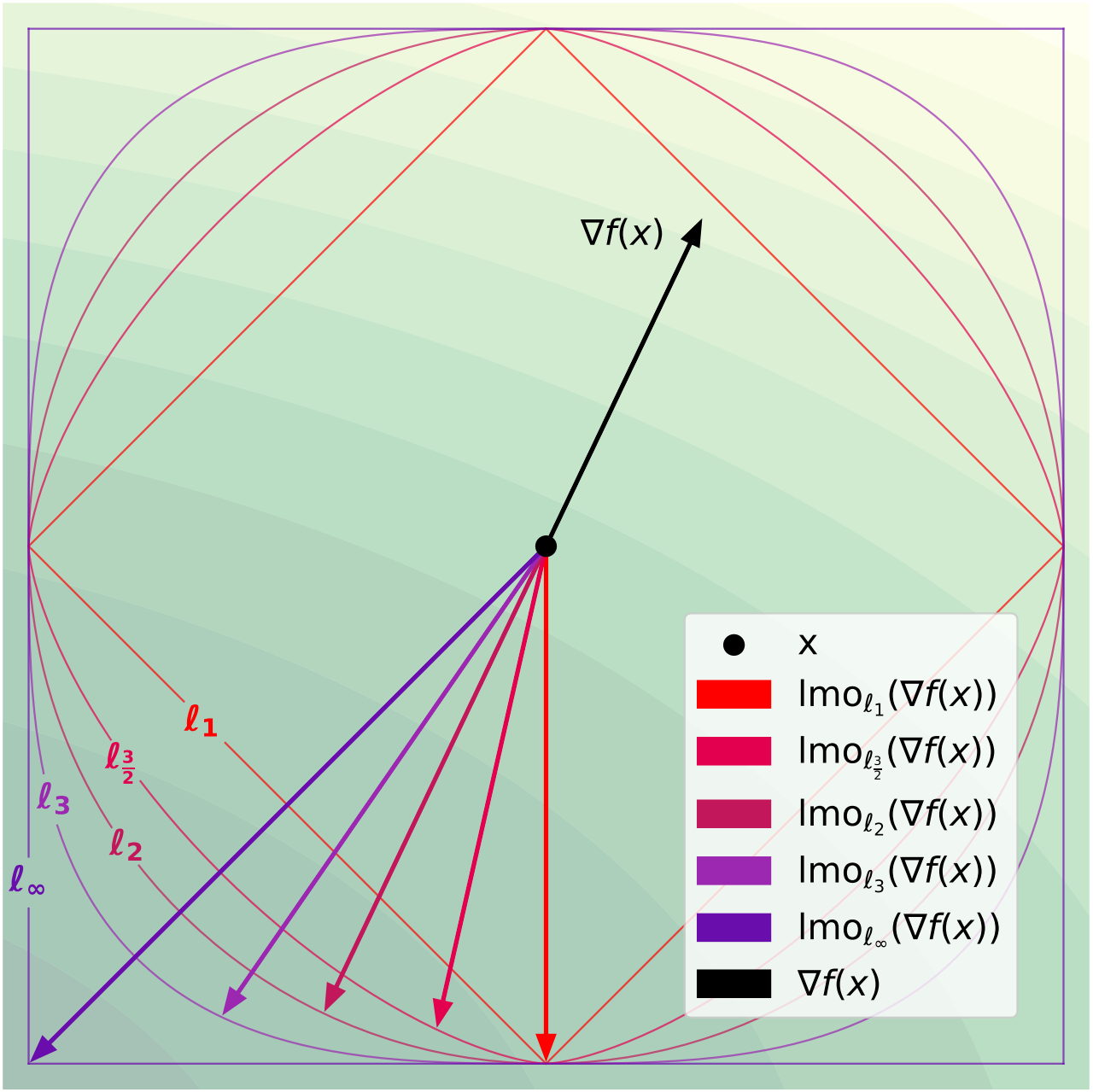
Key insight: Choosing a norm changes the geometry of the optimization and can be tailored to the structure of the problem!

Linear Minimization Oracles

Given a norm $\|\cdot\|$, we assume access to the associated *linear minimization oracle* (lmo),

$$\mathrm{lmo}(g) \in \operatorname*{argmin}_{\{s:\ \|s\| \le 1\}} \langle g, s \rangle.$$

- lmo is *scale-invariant*; $\mathrm{lmo}(ag) = \mathrm{lmo}(g)$ for all $a > 0$.
- The lmo for the scaled ball is the scaled lmo for the unit ball.



Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls

If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$, then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Ball	**Linear Minimization Oracle (lmo)**
ℓ_2 Ball	$\mathrm{lmo}(g) = -\frac{g}{\|g\|_2}$
Dual Norm	**Sharp Operator**
$\|\cdot\|_* = \|\cdot\|_2$	$\sharp(g) = -\|g\|_2 \left(-\frac{g}{\|g\|_2}\right) = g$

Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls

If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Ball	**Linear Minimization Oracle (lmo)**
ℓ_1 Ball	$\mathrm{lmo}(g) = -\mathrm{sign}(g_i)e_i$ where $i \in \underset{1\leq j\leq n}{\mathrm{argmax}} \lvert g_j\rvert$
Dual Norm	**Sharp Operator**
$\|\cdot\|_* = \|\cdot\|_\infty$	$\sharp(g) = -\|g\|_\infty(-\mathrm{sign}(g_i)e_i) = (\max_i g_i)\,\mathrm{sign}(g_i)e_i$

Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls

If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Ball	**Linear Minimization Oracle (lmo)**
ℓ_∞ Ball	$\mathrm{lmo}(g) = -\mathrm{sign}(g)$
Dual Norm	**Sharp Operator**
$\|\cdot\|_* = \|\cdot\|_1$	$\sharp(g) = -\|g\|_1 \left(-\mathrm{sign}(g)\right) = \left(\sum_i g_i\right) \mathrm{sign}(g)$

Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls

If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Ball	**Linear Minimization Oracle (lmo)**
Nuclear Norm Ball ℓ^1 norm on singular values	$\mathrm{lmo}(g) = -uv^T$ where (u, v) are leading singular vectors
Dual Norm	**Sharp Operator**
$\|\cdot\|_* = \|\cdot\|_{\mathrm{op}}$	$\sharp(g) = -\|g\|_{\mathrm{op}}\left(-uv^T\right) = \sigma_1(g)\left(uv^T\right)$

Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls

If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Ball	**Linear Minimization Oracle (lmo)**
$\ell_2 \to \ell_2$ Operator Norm Ball ℓ^∞ norm on singular values	$\mathrm{lmo}(g) = -UV^T$ where $g = U\Sigma V^T$ (reduced SVD)
Dual Norm	**Sharp Operator**
$\|\cdot\|_* = \|\cdot\|_{\mathrm{Nuc}}$	$\sharp(g) = -\|g\|_{\mathrm{Nuc}}\left(-UV^T\right) = \left(\sum_i \sigma_i(g)\right)\left(UV^T\right)$

Examples of Linear Minimization Oracles

Linear Minimization Oracles (lmo) for Norm Balls

If $\mathcal{D}$ is the unit-ball associated to a norm $\|\cdot\|$,
then $\mathrm{lmo}_{\mathcal{D}}(g) = -\partial\|g\|_*$ where $\|\cdot\|_*$ is the *dual norm*.

Ball	**Linear Minimization Oracle (lmo)**
$\ell_\infty \to \ell_\infty$ Operator Norm Ball	$\mathrm{lmo}(g) =$ Row-wise ℓ_1 lmo: $= -\mathrm{sign}(g_{i,j_i})e_{i,j_i}$ where $j_i \in \mathrm{argmax}_j \lvert g_{ij}\rvert$
Dual Norm	**Sharp Operator**
Row-wise sum norm	$\sharp(g) = -\|g\|_{\text{row-sum}}\,\mathrm{lmo}(g)$
$\|g\|_{\text{row-sum}} = \max_i \sum_j \lvert g_{ij}\rvert$	$= \left(-\max_i \sum_j \lvert g_{ij}\rvert\right) \begin{pmatrix} \begin{bmatrix} \mathrm{lmo}_{\ell^1}(g_{1\cdot}) \\ \vdots \\ \mathrm{lmo}_{\ell^1}(g_{n\cdot}) \end{bmatrix} \end{pmatrix}$

- Extends to many sets beyond norm balls (polyhedral sets, atomic sets)

Generalized Matching Pursuit

The generalized matching pursuit problem is given by

$$\min_{x \in \text{span}(\mathcal{A})} f(x)$$

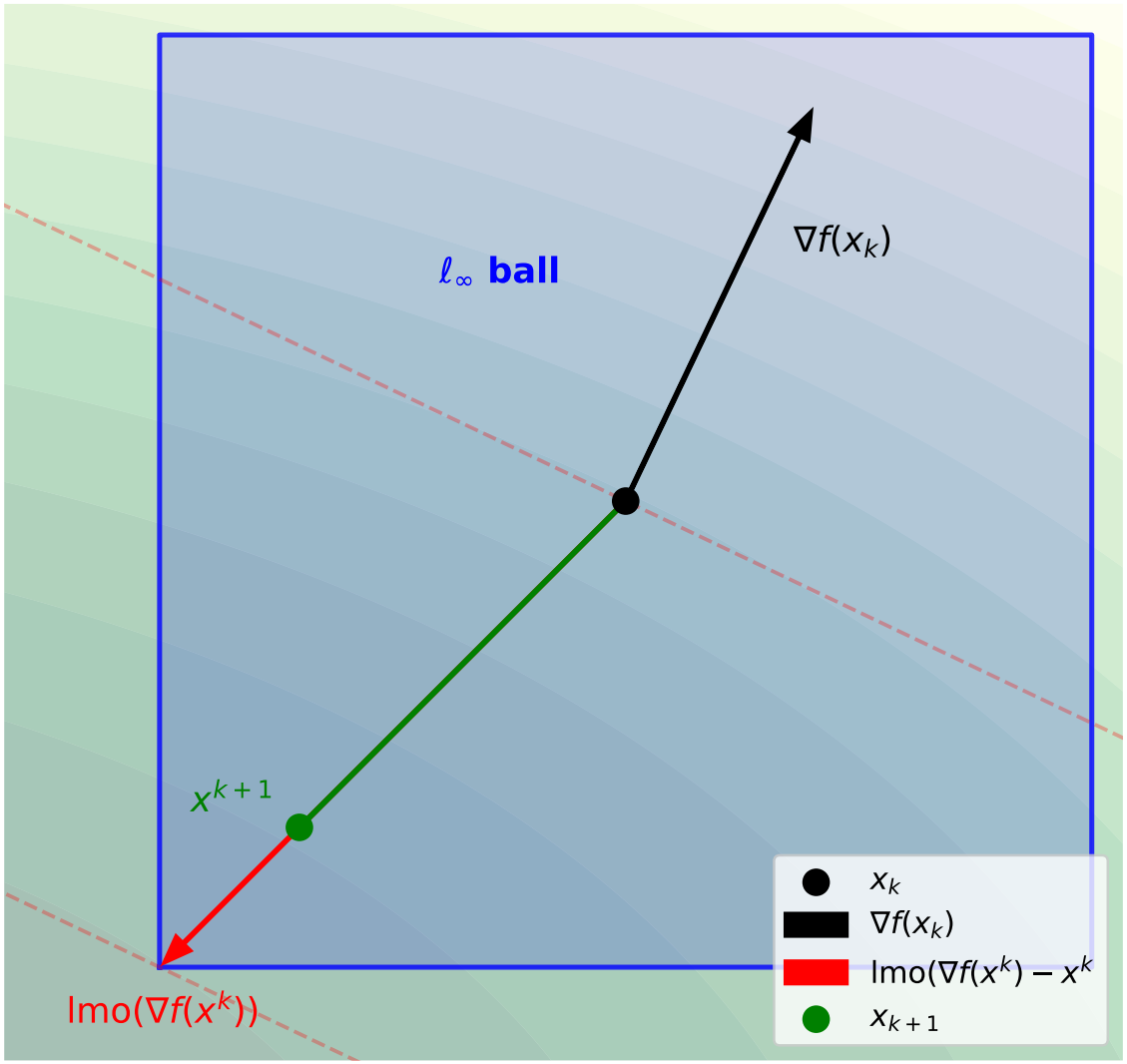
where $\mathcal{A}$ is a set of *atoms*.

Generalized Matching Pursuit (GMP):

Input: $x_0 \in \text{span}(\mathcal{A})$, stepsizes $\{\gamma_k\}$
for $k = 0, 1, \ldots, n-1$ **do**
  $s_k = \text{lmo}(\nabla f(x_k))$
  $x_{k+1} = x_k + \gamma_k s_k$
Output: x^n

- GMP uses the LMO to greedily select atoms from $\mathcal{A}$
- Special cases: Coordinate descent (when $\mathcal{A}$ = standard basis)

Conditional Gradient Algorithm



The conditional gradient algorithm (also known as Frank-Wolfe algorithm) solves constrained optimization problems:

$$\min_{x \in \mathcal{D}} f(x)$$

Conditional Gradient (CG):

Input: $x_0 \in \mathcal{D}$, stepsizes $\{\gamma_k\}$ where $\gamma_k \in [0, 1]$, horizon $n \in \mathbb{N}^*$

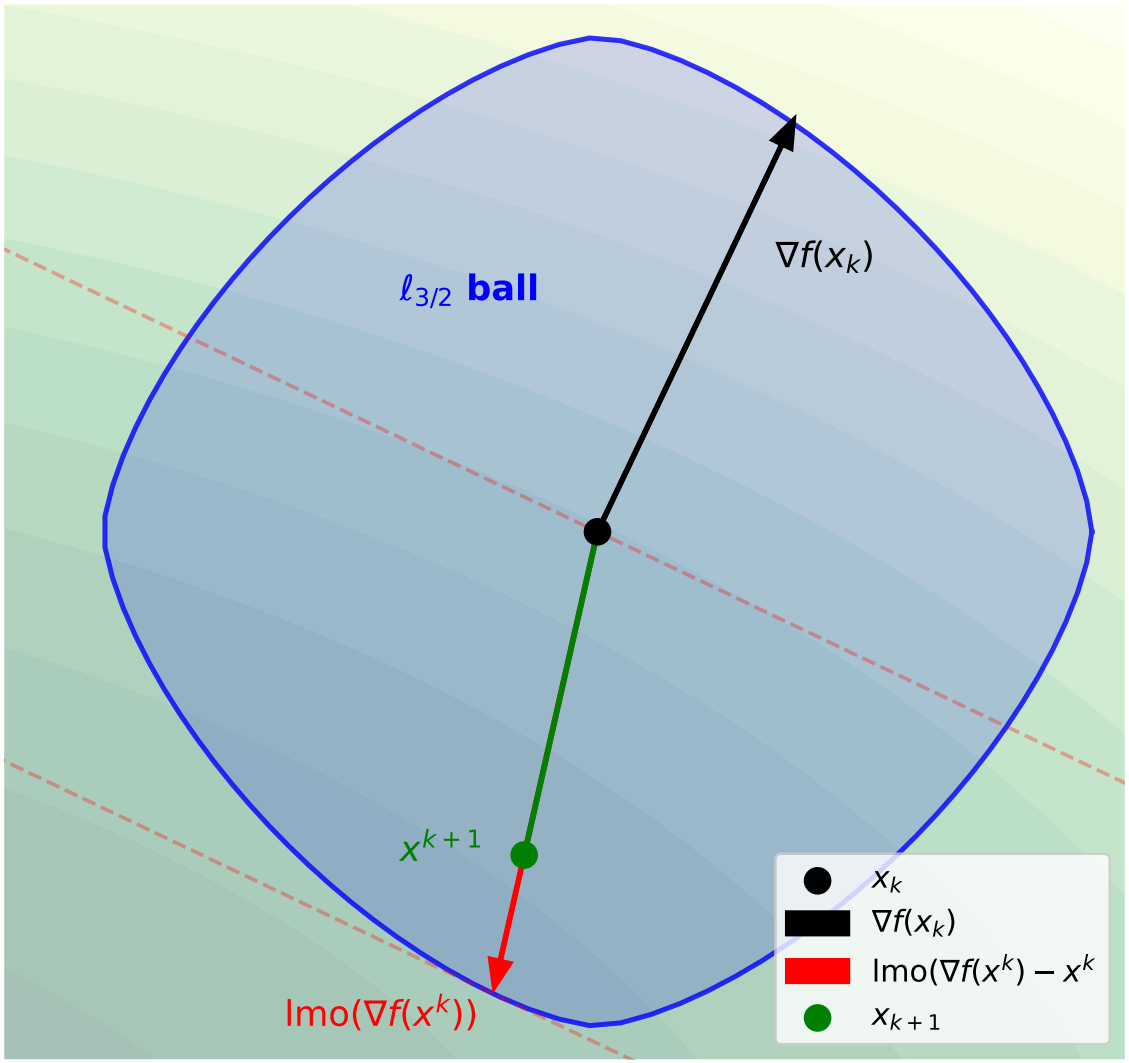
for $k = 0, 1, \ldots, n-1$ **do**

$s^k = \text{lmo}(\nabla f(x^k))$

$x^{k+1} = (1 - \gamma_k)x^k + \gamma_k s^k$

Output: x^n

Conditional Gradient Algorithm



The conditional gradient algorithm (also known as Frank-Wolfe algorithm) solves constrained optimization problems:

$$\min_{x \in \mathcal{D}} f(x)$$

Conditional Gradient (CG):

Input: $x_0 \in \mathcal{D}$, stepsizes $\{\gamma_k\}$ where $\gamma_k \in [0, 1]$, horizon $n \in \mathbb{N}^*$

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

$s^k = \mathrm{lmo}(\nabla f(x^k))$

$x^{k+1} = (1 - \gamma_k)x^k + \gamma_k s^k$

Output: x^n

Plan

- Motivation
- Algorithms using LMOs
- **Our algorithms: uSCG/SCG**
- Applications and empirical results

A Stochastic Conditional Gradient that uses Momentum

(Unconstrained) Stochastic Conditional Gradient (uSCG/SCG):

Input: $x^0 \in \mathcal{D}$, stepsizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $n \in \mathbb{N}$
Initialize $d^0 = 0$
for $k = 0, 1, 2, \ldots n-1$ **do**
  Sample ξ_k
  $g^k = \nabla f(x^k, \xi_k)$
  $d^k = (1-\alpha_k)d^{k-1} + \alpha_k g^k$
  $s^k = \mathrm{lmo}(d^k)$
  $v^k = \begin{cases} s^k & \text{uSCG} \\ s^k - x^k & \text{SCG} \end{cases}$
  $x^{k+1} = x^k + \gamma_k v^k$
Output: $\bar{x}^n$ selected uniformly at random among all iterates.

- Momentum reduces variance in stochastic setting.
- The difference between uSCG and SCG is *not* just Tikhonov regularization; there is a geometric interpretation.
- Unifies steepest descent, generalized matching pursuit, and conditional gradient algorithms into one framework.

Weight Decay and SCG

In Deep Learning, it has long been understood that Weight Decay should not simply be seen as Tikhonov regularization (Hutter et al.).

$$\text{GD with weight decay (decoupled):} \quad x^{k+1} = (1-\lambda)x^k - \gamma \nabla f(x^k)$$

$$\text{GD on Tikhonov problem (coupled):} \quad x^{k+1} = x^k - \gamma(\nabla f(x^k) + \lambda x^k)$$

However, these two formulations really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better").

Weight Decay and SCG

In Deep Learning, it has long been understood that Weight Decay should not simply be seen as Tikhonov regularization (Hutter et al.).

$$\text{GD with weight decay (decoupled):} \quad x^{k+1} = (1-\lambda)x^k - \gamma \nabla f(x^k)$$
$$\text{GD on Tikhonov problem (coupled):} \quad x^{k+1} = x^k - \gamma(\nabla f(x^k) + \lambda x^k)$$

However, these two formulations really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better").
On the other hand, in a noneuclidean setting, this point is *crucial* because the lmo is nonlinear.

$$\begin{aligned}\text{uSCG with weight decay} \to \text{SCG:} \quad x^{k+1} &= (1-\lambda)x^k - \gamma \operatorname{lmo}(\nabla f(x^k)) \\ &= (1-\lambda)x^k - \lambda \operatorname*{lmo}_{\gamma/\lambda}(\nabla f(x^k))\end{aligned}$$
$$\text{uSCG on Tikhonov problem:} \quad x^{k+1} = x^k - \gamma \operatorname{lmo}(\nabla f(x^k) + \lambda x^k)$$

The "correct" interpretation of Weight Decay in this context is that it transforms your unconstrained optimizer into a constrained optimizer, with implicit radii that are dictated by the chosen combination of stepsize γ and Weight Decay λ!

Norm Control

Let ρ be the radius of the set $\mathcal{D}$ that is used to define lmo. Both uSCG and SCG provide control over the norm of the output $\bar{x}^n$:

- SCG Guarantees $\|\bar{x}^n\| \leq \rho$
- uSCG Guarantees $\|\bar{x}^n\| \leq \rho \sum_{k=0}^{n-1} \gamma_k$

We can use uSCG to solve constrained problems by initializing at 0 and taking γ_k so that $\sum_{k=0}^{n-1} \gamma_k = 1$, e.g., $\gamma_k = \frac{1}{k+1}$.

Relationship to other Algorithms

Algorithm	α	**Norm**	lmo **Formula**
Normalized SGD	1	Euclidean $\|\cdot\|_2$	$-\frac{d}{\|d\|_2}$
Normalized SGD with momentum	$[0,1]$	Euclidean $\|\cdot\|_2$	$-\frac{d}{\|d\|_2}$
SignSGD	1	Max-norm $\|\cdot\|_\infty$	$-\mathrm{sign}(d)$
Signum	$[0,1]$	Max-norm $\|\cdot\|_\infty$	$-\mathrm{sign}(d)$
Muon*	$[0,1]$	$\ell^2 \to \ell^2$ operator-norm $\|\cdot\|_{\mathrm{op}}$	$-UV^T$

SVD decomposition notation: $d = U\mathrm{diag}(\sigma)V^T$
Our framework generalizes these algorithms through norm selection and momentum parameter.

Convergence Results for fixed α

Let ρ be the radius of the set $\mathcal{D}$ used in the lmo.

Theorem (Convergence rate for uSCG with constant α)

Let $n \in \mathbb{N}^*$ and let $\bar{x}^n$ be the output of uSCG with $\alpha \in (0,1)$ and constant stepsize $\gamma = \frac{1}{\sqrt{n}}$. Then,

$$\mathbb{E}[\|\nabla f(\bar{x}^n)\|_*] \leq O\left(\frac{L\rho}{\sqrt{n}} + \sigma\right)$$

Theorem (Convergence rate for SCG with constant α)

Let $n \in \mathbb{N}^*$ and let $\bar{x}^n$ be the output of SCG with $\alpha \in (0,1)$ and constant stepsize $\gamma = \frac{1}{\sqrt{n}}$. Then, for all $u \in \mathcal{D}$,

$$\mathbb{E}[\langle \nabla f(\bar{x}^n), \bar{x}^n - u\rangle] \leq O\left(\frac{L\rho^2}{\sqrt{n}} + \sigma\right)$$

$\Longrightarrow$ convergence to a noise-dominated region given by σ.

Convergence Results for vanishing α_k

Let ρ be the radius of the set $\mathcal{D}$ used in the lmo.

Theorem (Convergence rate for uSCG with vanishing α_k)

Let $n \in \mathbb{N}^*$ and let $\bar{x}^n$ be the output of uSCG with $\alpha_k = 1/\sqrt{k}$ and constant stepsize $\gamma = \frac{3}{4n^{3/4}}$. Then,

$$\mathbb{E}[\|\nabla f(\bar{x}^n)\|_*] \leq O\left(\frac{1}{n^{1/4}} + \frac{L\rho}{n^{3/4}}\right)$$

Theorem (Convergence rate for SCG with vanishing α_k)

Let $n \in \mathbb{N}^*$ and let $\bar{x}^n$ be the output of SCG with $\alpha_k = 1/\sqrt{k}$ and constant stepsize $\gamma = \frac{3}{4n^{3/4}}$. Then, for all $u \in \mathcal{D}$,

$$\mathbb{E}[\langle \nabla f(\bar{x}^n), \bar{x}^n - u\rangle] \leq O\left(\frac{1}{n^{1/4}} + \frac{L\rho^2}{n^{3/4}}\right)$$

$\Longrightarrow$ convergence to a first-order critical point for either the unconstrained (uSCG) or the constrained (SCG) problem.

Plan

- Motivation
- Algorithms using LMOs
- Our algorithms: uSCG/SCG
- **Applications and empirical results**

Model of a Neural Network

We consider an L-layer fully-connected neural network with input $a \in \mathbb{R}^p$ and output $b \in \mathbb{R}$:

$$h^{(0)} = a$$

$$h^{(l)}(h^{(l-1)}) = \sigma\left(\underbrace{\begin{bmatrix} & \mathbf{W}_l & \end{bmatrix}\begin{bmatrix} h^{(l-1)} \end{bmatrix}}_{\text{pre-activation } g^l}\right),$$

$$b = h_x(a) = h^{(L)}(h^{(L-1)}(\ldots)), \quad x := [W_1, W_2, \ldots, W_L]$$

- $W_1 \in \mathbb{R}^{m \times p}$, $W_L \in \mathbb{R}^{1 \times m}$, $W_l \in \mathbb{R}^{m \times m}$ for all $l \in \{2, \ldots, L-1\}$
- m is the *width* of the network

Feature Learning

We will now discuss how to choose a norm for the specific problem of training NNs. How should one update the weights during training for "good performance"?

Definition (Feature Learning [Yang, Simon, Benstein 2023])

Let $\Delta h^{(l)}$ denote the feature change after one iteration of training, for the l^{th} layer. We are in the feature learning regime if the following properties hold:

1. $\|h^{(l)}\|_{\text{RMS}} = \Theta(1), \quad \forall l \in [L]$ (stable forward pass),
2. $\|\Delta h^{(l)}\|_{\text{RMS}} = \Theta(1), \quad \forall l \in [L]$ (bounded feature update),

where the RMS norm is defined as $\|\cdot\|_{\text{RMS}} := \frac{1}{\sqrt{m}}\|\cdot\|_2$

Spectral Conditions for Feature Learning

Definition (Spectral Condition)

Given an L-layer NN, consider applying a gradient update ΔW_l to the weight matrix W_l. If the spectral norms of the weights and the weight updates satisfy the following $\forall 2 \leq l \leq L-1$,

$$\begin{aligned} \|W_1\|_{\text{op}} &= \Theta\left(\sqrt{\tfrac{m}{p}}\right) & \|\Delta W_1\|_{\text{op}} &= \Theta\left(\sqrt{\tfrac{m}{p}}\right) \\ \|W_l\|_{\text{op}} &= \Theta(1) & \|\Delta W_l\|_{\text{op}} &= \Theta(1) \\ \|W_L\|_{\text{op}} &= \Theta\left(\sqrt{\tfrac{1}{m}}\right) & \|\Delta W_L\|_{\text{op}} &= \Theta\left(\sqrt{\tfrac{1}{m}}\right) \end{aligned}$$

then we have *feature-learning*.

- This spectral condition ensures that $\|h^{(l)}\|_{\text{RMS}} = \Theta(1)$ and $\|\Delta h^{(l)}\|_{\text{RMS}} = \Theta(1)$.
- This can be extended to rectangular matrices by requiring the norm of both objects to scale like $\Theta\left(\sqrt{\frac{n_{\text{in}}}{n_{\text{out}}}}\right)$

Picking a Norm and Initializations

If we can specify a norm $\|\cdot\|_{\alpha_l}$ for the input space and a norm $\|\cdot\|_{\beta_l}$ for the output spaces of each layer of our network, then this induces an operator norm for each layer.
We can specify a norm for the whole set of parameters by taking

$$\|x\| = \max_{l \in [L]} \{\|W_l\|_{\alpha_l \to \beta_l}\}$$

Spectral Feature learning suggests taking the RMS norm on the input and output spaces of intermediary layers.
$\to$ leads to a scaled ℓ^2 operator norm on weight matrices:

$$\|X\|_{\text{RMS}\to\text{RMS}} = \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}}\|X\|_{\text{op}}.$$

The lmo associated to the ball for this norm is given by $-\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}UV^T$.

The first and final layers require more thought!

Norms for different layers

The operator norm chosen for the initial layer differs from the intermediary layers, depending on the task (NLP, images, etc).
For language tasks, the input z is usually a 1-hot encoded vector so

$$\|z\|_\infty = \|z\|_2 = \|z\|_1 = 1$$

identically. This in turn means

$$\|W_1\|_{\infty\to\text{RMS}} = \|W_1\|_{2\to\text{RMS}} = \|W_1\|_{1\to\text{RMS}}$$

on this restricted domain.

Table 4. Example lmo choices for 1-hot encoded inputs.

Parameter	W_1 (1-hot encoded input)		
Norm	$2 \to \text{RMS}$	$1 \to \text{RMS}$	$1 \to \infty$
LMO	$\sqrt{d_{\text{out}}}UV^\top$	$\text{col}_i(W_1) \mapsto \sqrt{d_{\text{out}}}\frac{\text{col}_i(W_1)}{\|\text{col}_i(W_1)\|_2}$	$\text{sign}(W_L)$
Init.	Semi-orthogonal	Column-wise normalized Gaussian	Random sign

(Note there is a sign error for lmo in this table)

Norms for different layers

We have no restriction to bound the output in RMS norm; instead we consider bounding the maximal entry using L_∞.
Also note that $\|A\|_{\mathrm{RMS}\to\infty} \le \frac{1}{d_{\text{in}}}\|A\|_{1\to\infty}$ which gives us a scaled sign lmo for the last layer.

Table 2. Example operator norms and the associated lmos of a matrix $A \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$. The reduced SVD is given as $A = U\,\mathrm{diag}(\sigma)V^\top$, sign acts elementwise, $\mathrm{col}_i(A) := A_{i,\cdot}$ and $\mathrm{row}_j(A) := A_{\cdot,j}$. Note that this table is not exhaustive.

	$1 \to \mathrm{RMS}$ (ColNorm)	$1 \to \infty$ (Sign)	$\mathrm{RMS} \to \mathrm{RMS}$ (Spectral)	$\mathrm{RMS} \to \infty$ (RowNorm)
Norm	$\max_i \frac{1}{\sqrt{d_{\text{out}}}}\|\mathrm{col}_i(A)\|_2$	$\max_{i,j}\|A_{i,j}\|$	$\sqrt{d_{\text{in}}/d_{\text{out}}}\|A\|_{\mathcal{S}_\infty}$	$\max_j \sqrt{d_{\text{in}}}\|\mathrm{row}_j(A)\|_2$
LMO	$\mathrm{col}_i(A) \mapsto \sqrt{d_{\text{out}}}\frac{\mathrm{col}_i(A)}{\|\mathrm{col}_i(A)\|_2}$	$A \mapsto \mathrm{sign}(A)$	$A \mapsto \sqrt{d_{\text{out}}/d_{\text{in}}}UV^\top$	$\mathrm{row}_j(A) \mapsto \frac{1}{\sqrt{d_{\text{in}}}}\frac{\mathrm{row}_j(A)}{\|\mathrm{row}_j(A)\|_2}$

Table 3. The choice of lmo can be different between layers and can depend on the assumptions on the input. For simplicity we overload notation and write the reduced SVD as $W_\ell = U\,\mathrm{diag}(\sigma)V^\top \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ for all $\ell \in [L]$.

Parameter	W_1 (image domain)	$\{W_\ell\}_{\ell\in[2,\dots,L-1]}$	W_L			b_ℓ
Norm	$\mathrm{RMS}\to\mathrm{RMS}$	$\mathrm{RMS}\to\mathrm{RMS}$	$\mathrm{RMS}\to\mathrm{RMS}$	$\mathrm{RMS}\to\infty$	$1\to\infty$	RMS
LMO	$\max(1, \sqrt{d_{\text{out}}/d_{\text{in}}})UV^\top$	$\sqrt{d_{\text{out}}/d_{\text{in}}}UV^\top$	$\sqrt{d_{\text{out}}/d_{\text{in}}}UV^\top$	$\mathrm{row}_j(W_L) \mapsto \frac{1}{\sqrt{d_{\text{in}}}}\frac{\mathrm{row}_j(W_L)}{\|\mathrm{row}_j(W_L)\|_2}$	$\frac{1}{d_{\text{in}}}\mathrm{sign}(W_L)$	$\frac{b_\ell}{\|b_\ell\|_{\mathrm{RMS}}}$
Init.	Semi-orthogonal	Semi-orthogonal	Semi-orthogonal	Row-wise normalized Gaussian	Random sign	0

(Note there is a sign error for lmo in this table)

uSCION and SCION

Table 2. Example operator norms and the associated lmos of a matrix $A \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. The reduced SVD is given as $A = U \operatorname{diag}(\sigma) V^\top$, sign acts elementwise, $\operatorname{col}_i(A) := A_{i,\cdot}$ and $\operatorname{row}_j(A) := A_{\cdot,j}$. Note that this table is not exhaustive.

	$1 \to \text{RMS}$ (ColNorm)	$1 \to \infty$ (Sign)	$\text{RMS} \to \text{RMS}$ (Spectral)	$\text{RMS} \to \infty$ (RowNorm)
Norm	$\max_i \frac{1}{\sqrt{d_{\text{out}}}} \lVert \operatorname{col}_i(A) \rVert_2$	$\max_{i,j} \lvert A_{i,j} \rvert$	$\sqrt{d_{\text{in}}/d_{\text{out}}} \lVert A \rVert_{\mathcal{S}_\infty}$	$\max_j \sqrt{d_{\text{in}}} \lVert \operatorname{row}_j(A) \rVert_2$
LMO	$\operatorname{col}_i(A) \mapsto \sqrt{d_{\text{out}}} \frac{\operatorname{col}_i(A)}{\lVert \operatorname{col}_i(A) \rVert_2}$	$A \mapsto \operatorname{sign}(A)$	$A \mapsto \sqrt{d_{\text{out}}/d_{\text{in}}} UV^\top$	$\operatorname{row}_j(A) \mapsto \frac{1}{\sqrt{d_{\text{in}}}} \frac{\operatorname{row}_j(A)}{\lVert \operatorname{row}_j(A) \rVert_2}$

(Note there is a sign error for lmo in this table)

We refer to the instantiation of uSCG and SCG using operator norms as Unconstrained Scion and Scion respectively, which stands for

Stochastic **C**onditional grad**I**ent with **O**perator **N**orms

We recommend the following norms (First layer $\to$ Intermediary layers $\to$ Last layer):

- image domains: Spectral $\to$ Spectral $\to$ Sign
- 1-hot input: ColNorm $\to$ Spectral $\to$ Sign
- weight sharing: Sign $\to$ Spectral $\to$ Sign

Empirical Results

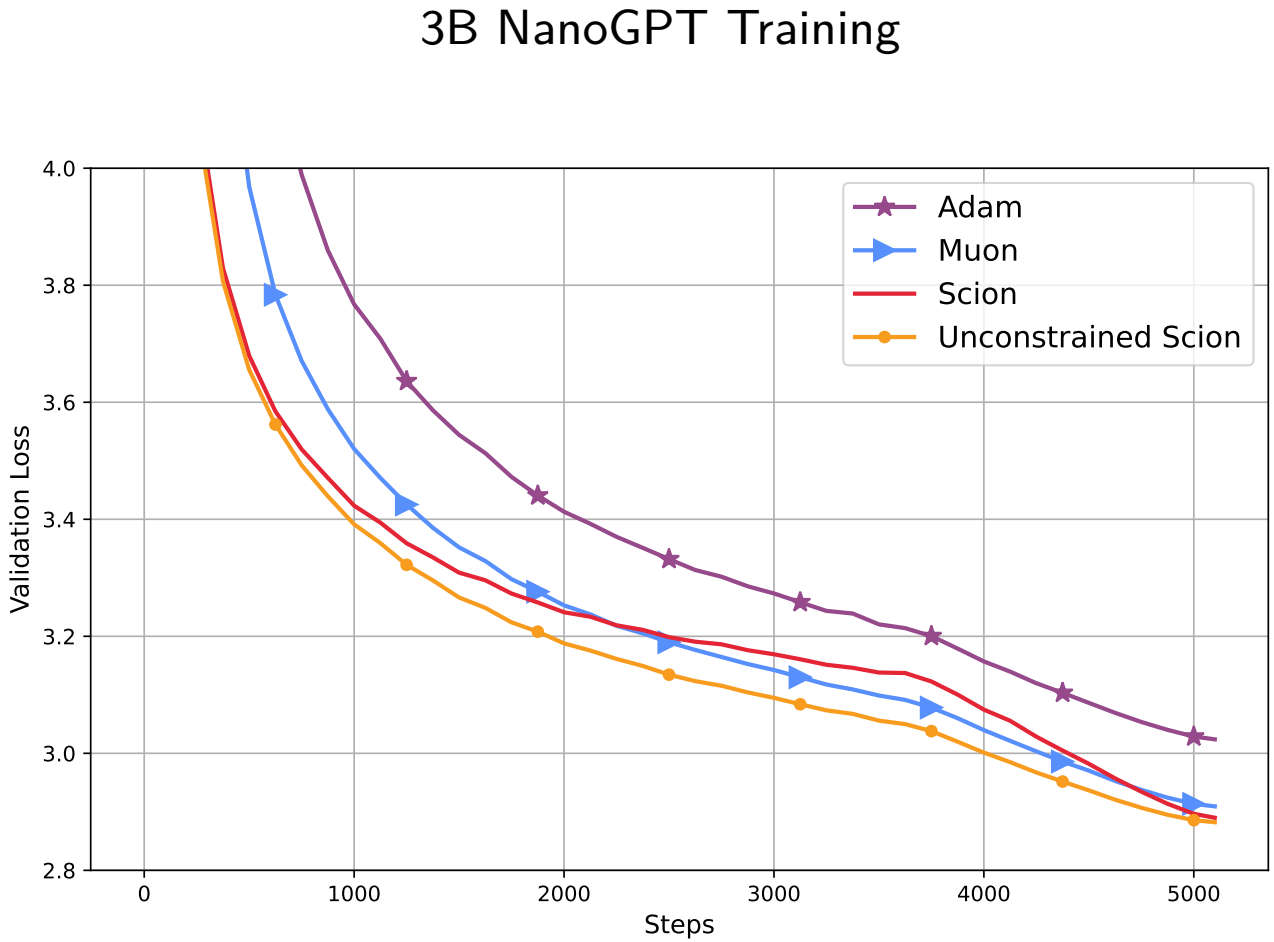


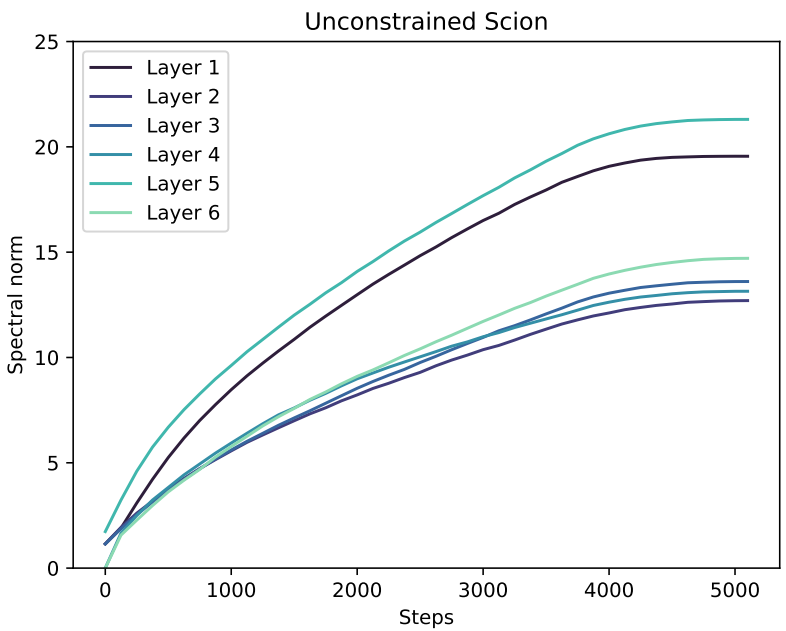
Table 5. Validation loss on a 3B parameter GPT model.

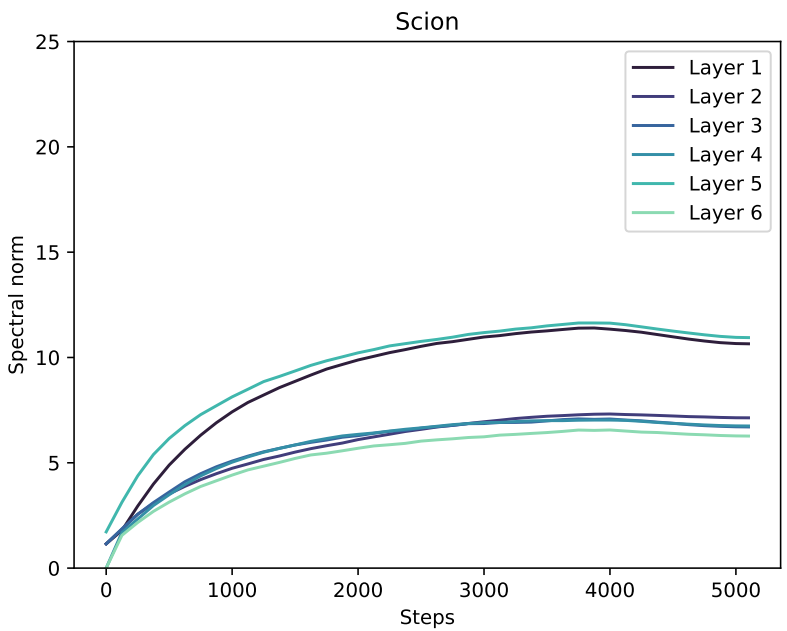
Adam	Muon	UNCONSTRAINED SCION	SCION
3.024	2.909	**2.882**	2.890

Illustration of norm control: GPT Training

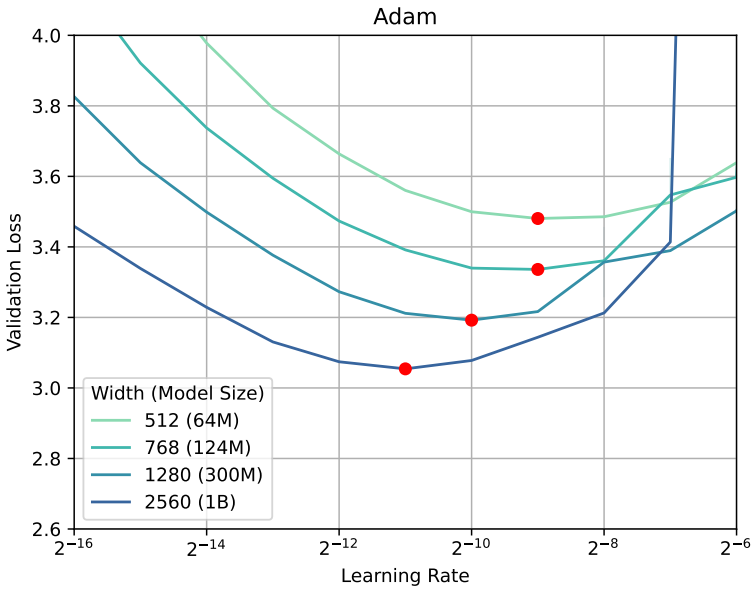
Let ρ be the radius of the set $\mathcal{D}$ that is used to define lmo. Both uSCG and SCG provide control over the norm of the output $\bar{x}^n$:

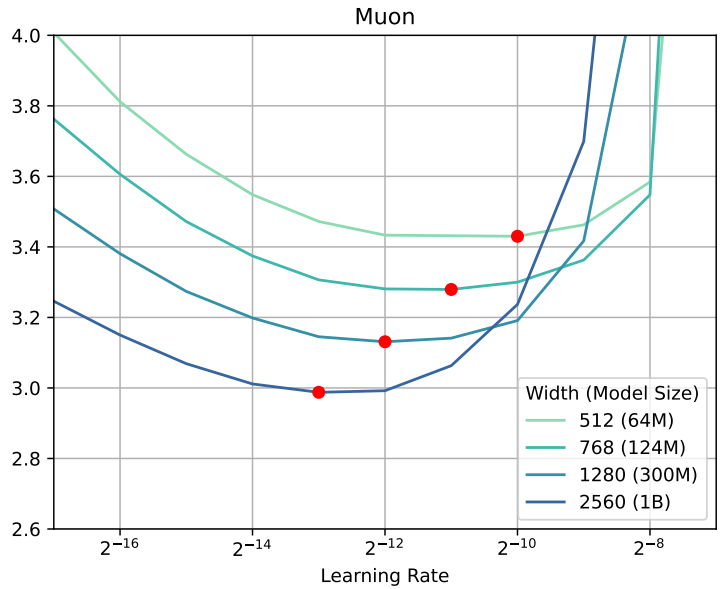
- SCG Guarantees $\|\bar{x}^n\| \leq \rho$
- uSCG Guarantees $\|\bar{x}^n\| \leq \rho \sum_{k=0}^{n-1} \gamma_k$

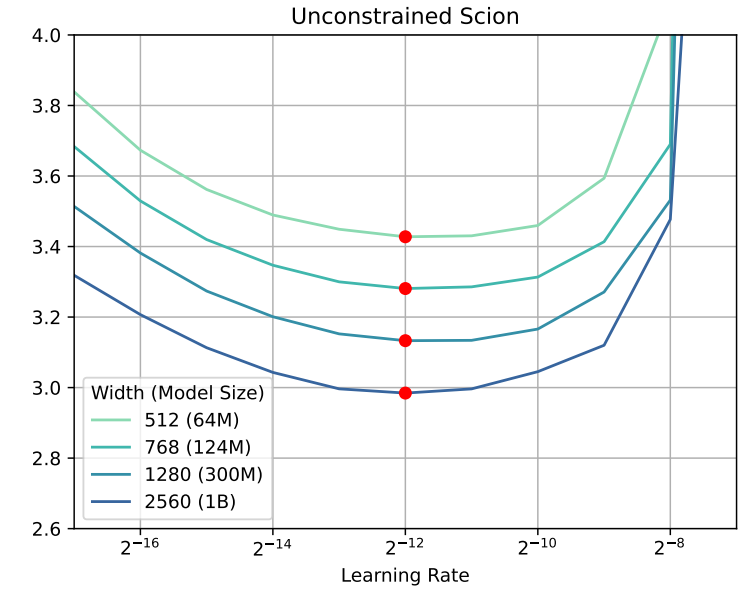


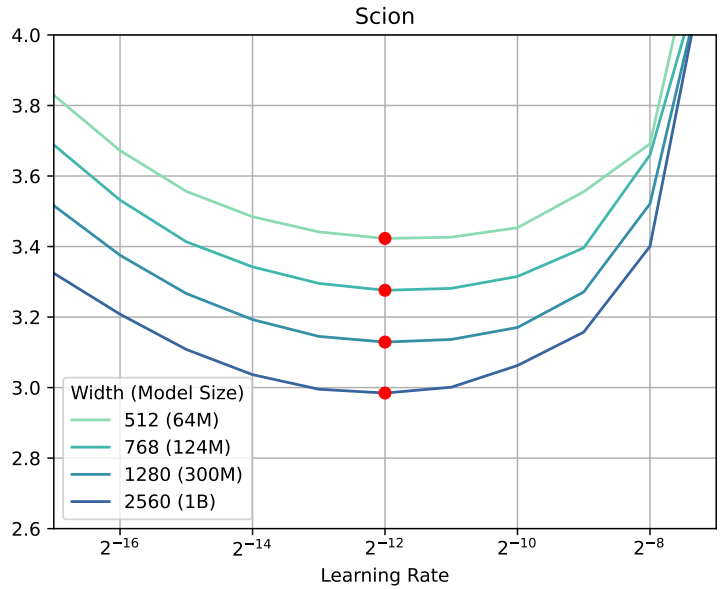


Hyperparameter Transfer: GPT Training



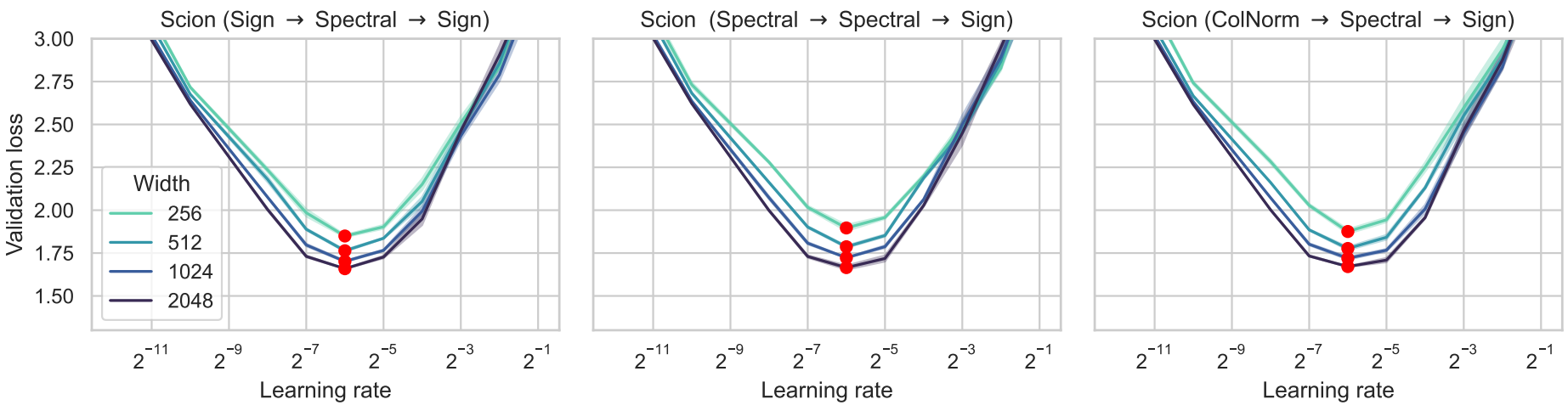




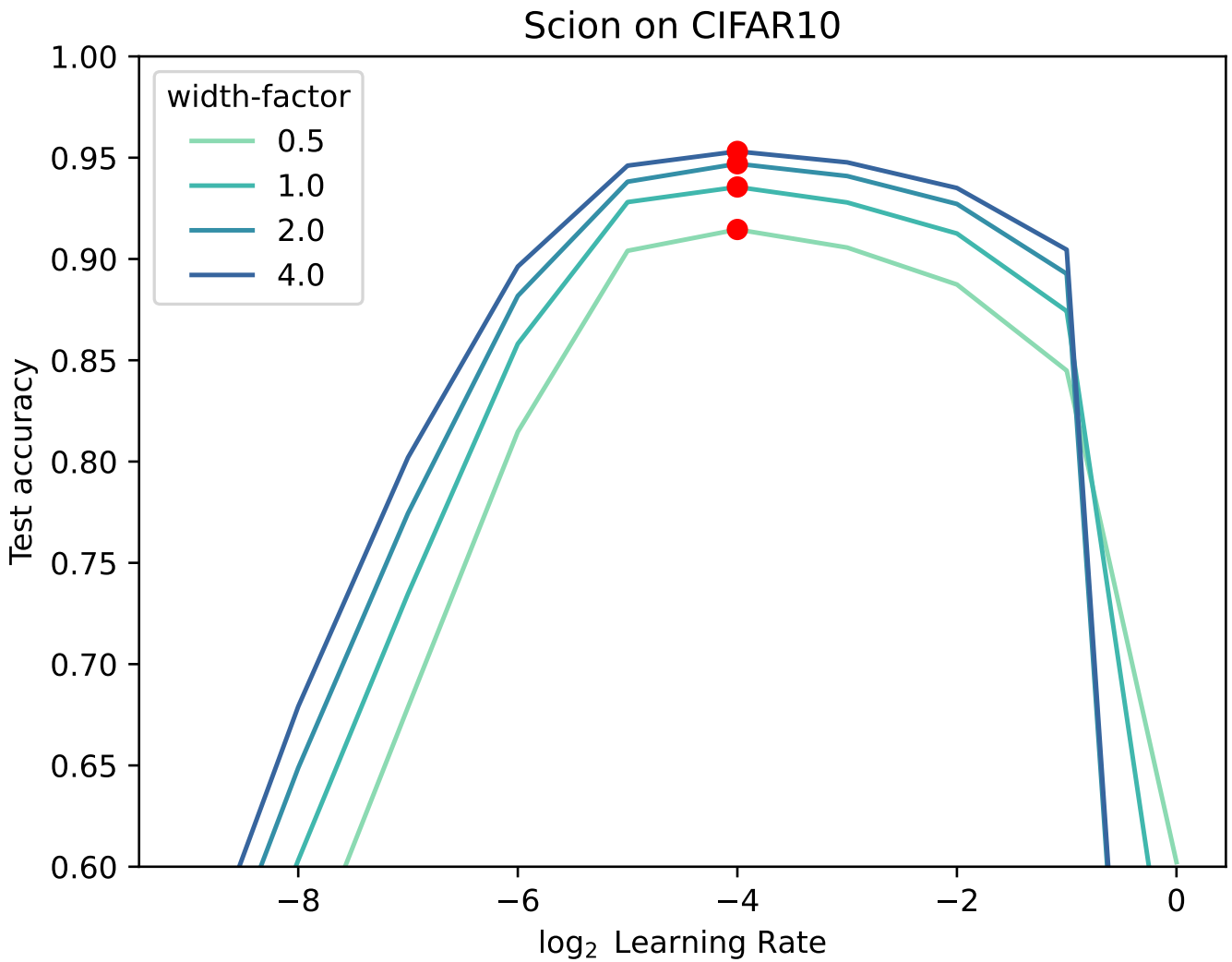


Different Norm Choices on First/Last Layer

Shallow MLP Trained on Cifar10



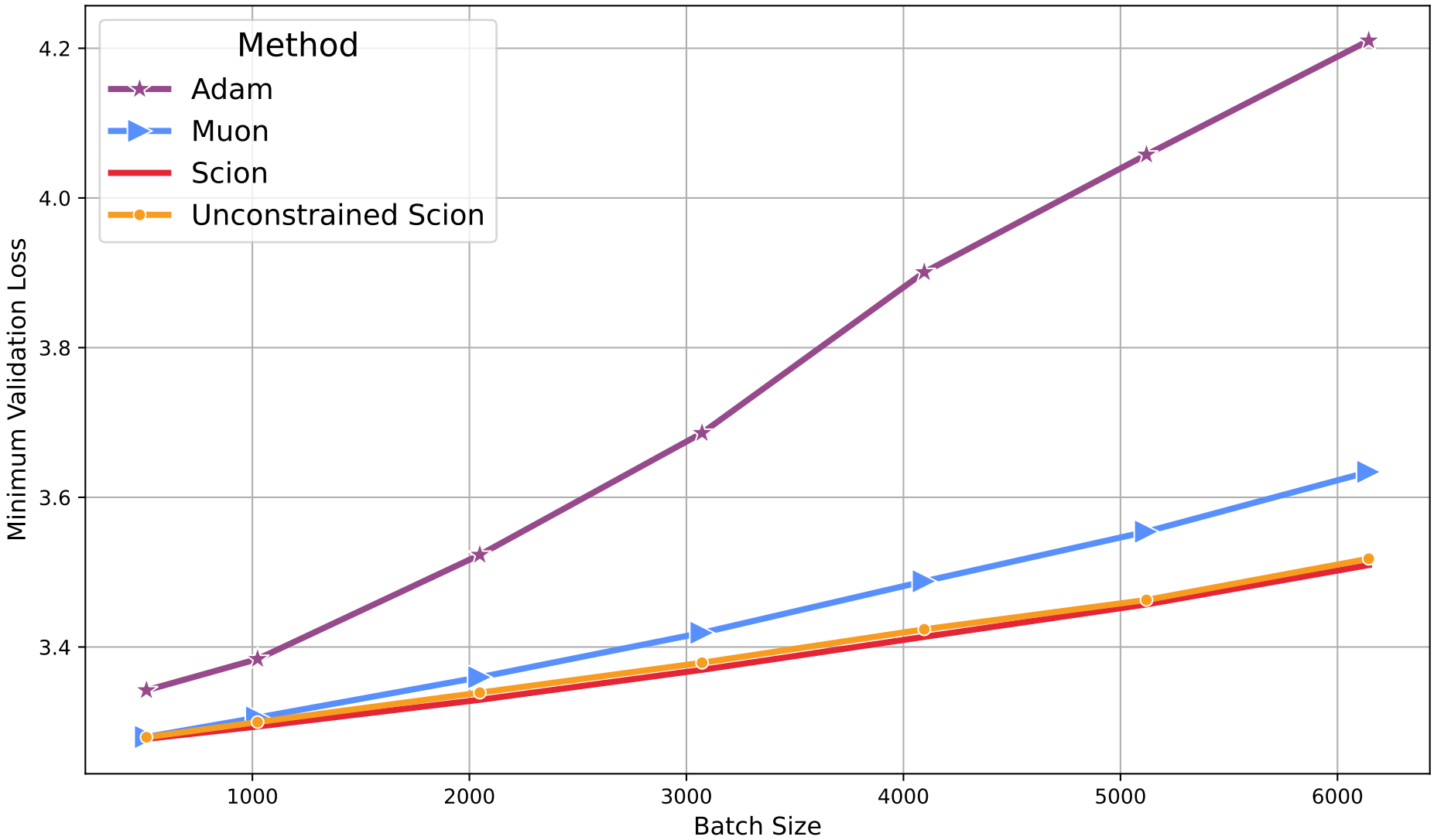
Hyperparameter Transfer



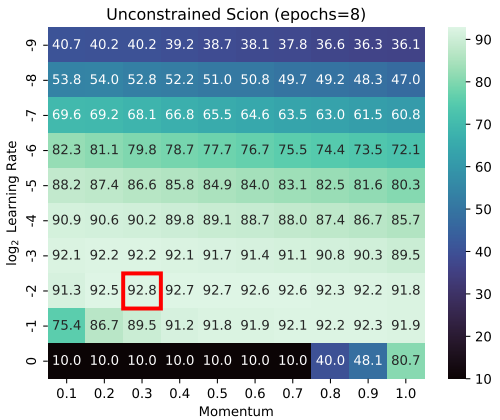
Optimal stepsize transfer across width in a convolutional NN trained to classify with CIFAR10.

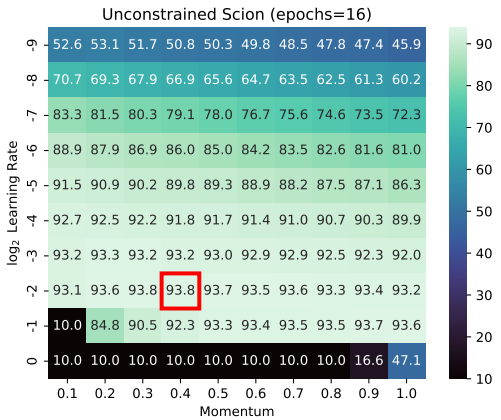
Effect of batch size

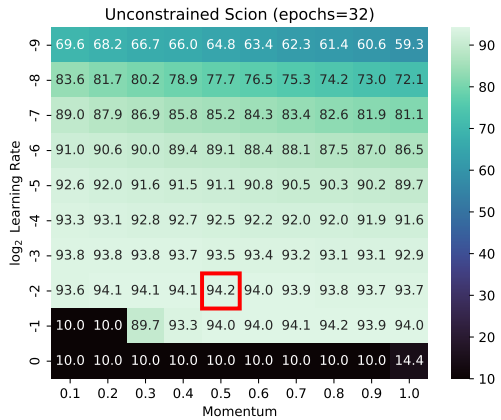
Batchsize sensitivity on NanoGPT (124M). SCION is less sensitive to batch increases (for a fixed token budget)

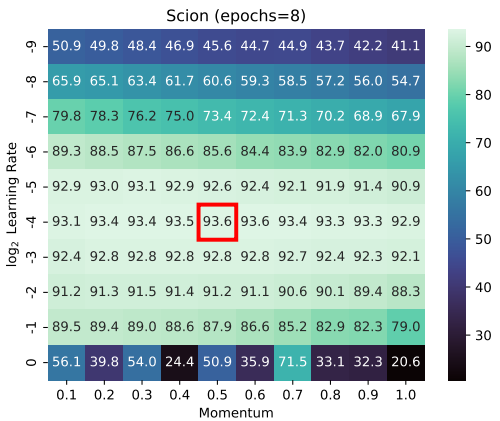


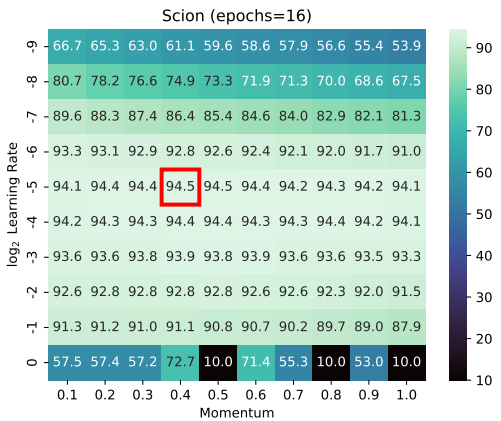
Tuning the momentum

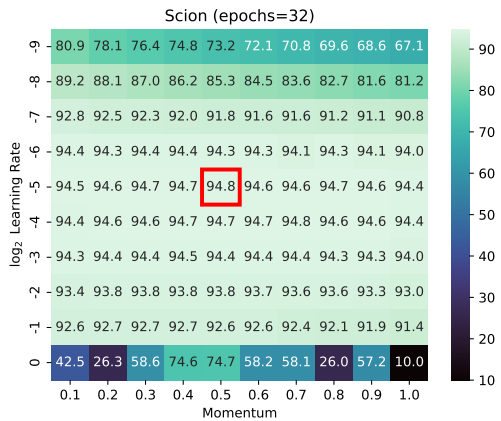












Related Work

Lion-K: Lizhang Chen, Bo Liu, Kaizhao Liang, Qiang Liu (Oct. 2023)
Muon blogpost: Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein (Dec. 2024)
Kimi Moonshot AI: many (Feb. 2025)
PSGD: Omead Pooladzandi and Xi-Lin Li (Feb. 2024)

ALMOND for dessert

Averaged **LMO** direction**N**al **D**escent (ALMOND):

Input: $x^0 \in \mathcal{D}$, stepsizes $\{\gamma_k\}$, momentum $\{\alpha_k\}$, horizon $n \in \mathbb{N}$
Initialize $d^0 = 0$
for $k = 0, 1, 2, \ldots n-1$ **do**
$g^k = \nabla f(x^k, \xi_k)$
$d^k = (1-\alpha_k)d^{k-1} + \alpha_k \mathrm{lmo}(g^k)$
$x^{k+1} = x^k + \gamma_k d^k$
Output: $\bar{x}^n$ selected uniformly at random among all iterates.

Not competitive empirically. Theoretically, can only show convergence to a noise dominated region.

Paper

arXiv:2502.07529

Computer Science > Machine Learning

[Submitted on 11 Feb 2025]

Training Deep Learning Models with Norm-Constrained LMOs

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, Volkan Cevher

In this work, we study optimization methods that leverage the linear minimization oracle (LMO) over a norm-ball. We propose a new stochastic family of algorithms that uses the LMO to adapt to the geometry of the problem and, perhaps surprisingly, show that they can be applied to unconstrained problems. The resulting update rule unifies several existing optimization methods under a single framework. Furthermore, we propose an explicit choice of norm for deep architectures, which, as a side benefit, leads to the transferability of hyperparameters across model sizes. Experimentally, we demonstrate significant speedups on nanoGPT training without any reliance on Adam. The proposed method is memory-efficient, requiring only one set of model weights and one set of gradients, which can be stored in half-precision.

Subjects: **Machine Learning (cs.LG)**; Optimization and Control (math.OC)
Cite as: arXiv:2502.07529 **[cs.LG]**
(or arXiv:2502.07529v1 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2502.07529