Training Deep Learning Models with Norm-Constrained LMOs

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

Séminaire Palaisien - May 7th, 2025

Motivation

A question...

How do we design an optimization algorithm that respects the natural geometry of neural networks?

Motivation

A question...

How do we design an optimization algorithm that respects the natural geometry of neural networks?

(in such a way that we guarantee effective learning across different model scales)

Motivation

What has been done so far?

Stochastic Gradient Descent (SGD):

Input: $x^0 \in \mathcal{X}$, step sizes $\{\gamma_k\}$,
horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \dots, n-1$ **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 uses a Euclidean geometry:

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

- This geometry is not representative of the problems we are interested in.

Two major improvements:

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, μP parameterizations, etc)

AdaGrad:

Input: $x^0 \in \mathcal{X}$, step size 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

- AdaGrad uses a Mahalanobis geometry:

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

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

1. *On-the-fly adaptation*: Methods that adapt during training (AdaGrad, RMSprop, Adam, AdamW)

SGD is not a good choice

RMSprop:

Input: $x^0 \in \mathcal{X}$, step size 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

- RMSprop also uses a Mahalanobis geometry:

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

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

- Adds the momentum parameters.

1. *On-the-fly adaptation*: Methods that adapt during training (AdaGrad, RMSprop, Adam, AdamW)

SGD is not a good choice

Adam:

Input: $x^0 \in \mathcal{X}$, step size γ, $\epsilon > 0$, momentum β_1, β_2, horizon $n \in \mathbb{N}^*$

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

 Sample ξ_k

 $g^k = \nabla f(x^k, \xi_k)$

 $m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$

 $v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$

 $\hat{m}^k = \frac{m^k}{1-\beta_1^k}$

 $\hat{v}^k = \frac{v^k}{1-\beta_2^k}$

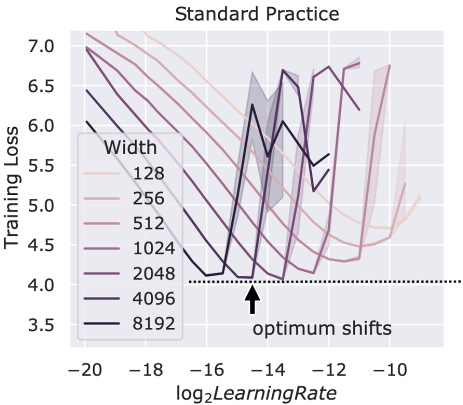
 $x^{k+1} = x^k - \frac{\gamma}{\sqrt{\hat{v}^k + \epsilon}} \odot \hat{m}^k$

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.

1. *On-the-fly adaptation*: Methods that adapt during training (AdaGrad, RMSprop, Adam, AdamW)

Shortcomings of ignoring architecture



Optimal learning rate shifts when we scale width.

- Because it's on-the-fly, Adam takes more memory when we scale our network (we have to keep track of + store the moments).

Failure of Adam to learn features as width scales

With standard parametrization (intialization + learning rate), we get stuck in the "lazy" regime if we scale width.

On Lazy Training in Differentiable Programming

Lénaïc Chizat
CNRS, Université Paris-Sud
Orsay, France
lenaic.chizat@u-psud.fr

Edouard Oyallon
CentraleSupelec, INRIA
Gif-sur-Yvette, France
edouard.oyallon@centralesupelec.fr

Francis Bach
INRIA, ENS, PSL Research University
Paris, France
francis.bach@inria.fr

Feature Learning in Infinite-Width Neural Networks

Greg Yang
Microsoft Research AI
gregyang@microsoft.com

Edward J. Hu*
Microsoft Azure AI
edwardhu@microsoft.com

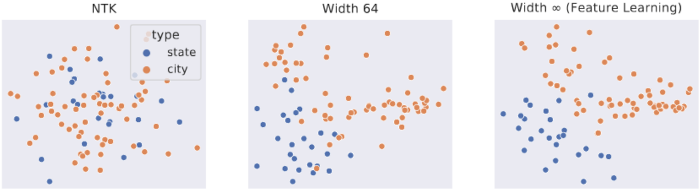


Figure 1: PCA of Word2Vec embeddings of top US cities and states, for NTK, width-64, and width-∞ feature learning networks (Definition 5.1). NTK embeddings are essentially random, while cities and states get naturally separated in embedding space as width increases in the feature learning regime.

Plan

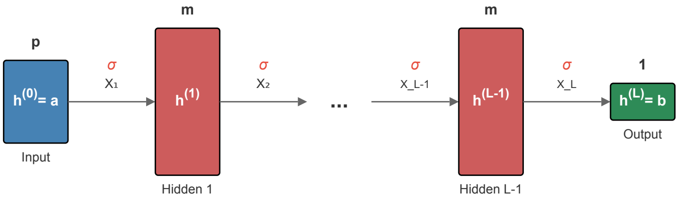
- Motivation
- **Feature Learning and μP**
- Noneuclidean Optimization
- (unconstrained) Stochastic Conditional Gradient
- A natural geometry for neural networks
- Empirical Results
- Theoretical Results

Model of a Neural Network

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

$$h^{(0)} = a \qquad h^{(l)}(h^{(l-1)}) = \sigma\left(\underbrace{\begin{bmatrix} & \mathbf{X}_l & \end{bmatrix} \begin{bmatrix} h^{(l-1)} \end{bmatrix}}_{\text{pre-activation } g^l} \right), \qquad b = h_x(a) = h^{(L)}(h^{(L-1)}(\ldots)).$$

- $x := [X_1, X_2, \ldots, X_L]$, $X_1 \in \mathbb{R}^{m \times p}$, $X_L \in \mathbb{R}^{1 \times m}$, $X_l \in \mathbb{R}^{m \times m}$ for all $l \in \{2, \ldots, L-1\}$
- m is the *width* of the network



Feature Learning

How should one update the weights during training for "good performance"?

Definition (Feature Learning)

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

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

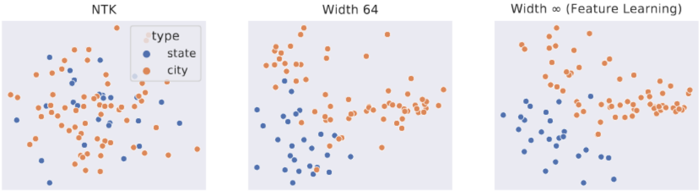
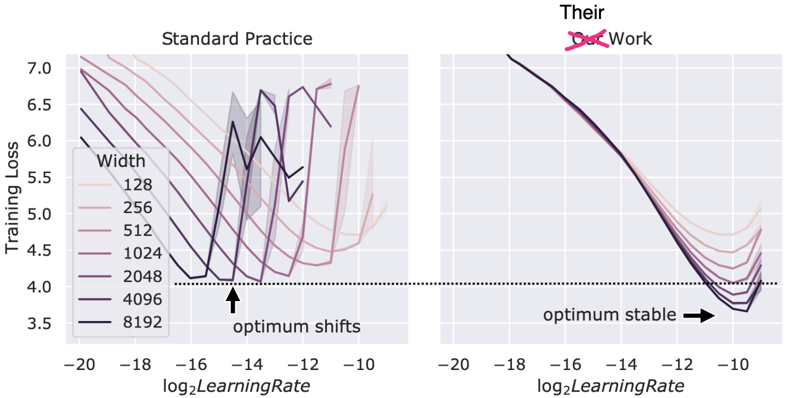


Figure 1: PCA of Word2Vec embeddings of top US cities and states, for NTK, width-64, and width-∞ feature learning networks (Definition 5.1). NTK embeddings are essentially random, while cities and states get naturally separated in embedding space as width increases in the feature learning regime.

A priori adaptation via μP

Certain initialization & layerwise step size that is scaled by dimensions to ensure

- the correct scaling behavior as the width goes to infinity (feature learning),
- that Adam/SGD has hyperparameter transfer for the global step size.



μP is architecture aware (different scaling depending on dimensions) - this is a priori adaptation.

Definition (Spectral Condition)

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

$$\begin{aligned} \|X_1\|_{\mathrm{op}} &= \Theta\left(\sqrt{\frac{m}{p}}\right) & \|\Delta X_1\|_{\mathrm{op}} &= \Theta\left(\sqrt{\frac{m}{p}}\right) \\ \|X_l\|_{\mathrm{op}} &= \Theta(1) & \|\Delta X_l\|_{\mathrm{op}} &= \Theta(1) \\ \|X_L\|_{\mathrm{op}} &= \Theta\left(\sqrt{\frac{1}{m}}\right) & \|\Delta X_L\|_{\mathrm{op}} &= \Theta\left(\sqrt{\frac{1}{m}}\right) \end{aligned}$$

then we have *feature-learning*.

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

$\Longrightarrow$ we need to control scaled operator norms layer-by-layer in the network to ensure feature learning as we scale width.

An architecture-aware norm for neural networks

Our strategy:

1. Cook up a noneuclidean norm based on the layerwise scaled operator norms.
2. Incorporate this noneuclidean norm into our optimization algorithm to have a priori adaptation.

Plan

- Motivation
- Feature Learning and μP
- **Noneuclidean Optimization**
- (unconstrained) Stochastic Conditional Gradient
- A natural geometry for neural networks
- Empirical Results
- Theoretical Results

The update of SGD can be written

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

Steepest Descent via Dual Norms

The update of SGD can be written

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

What if we change the norm?

Steepest Descent via Dual Norms

The update of Steepest Descent can be written

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

What if we change the norm?

Steepest Descent via Dual Norms

The update of Steepest Descent can be written

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

What if we change the norm?

This update has a closed-form solution using the dual norm $\|\cdot\|_*$,

$$x^{k+1} = x^k + \gamma_k \|g^k\|_* \operatorname{lmo}(g^k)$$

where lmo is the *linear minimization oracle*:

$$\operatorname{lmo}(g^k) \in \operatorname*{argmin}_{s \in \mathcal{D}} \langle g^k, s \rangle = -\partial \|g^k\|_*$$

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

Steepest Descent via Dual Norms

The update of Steepest Descent can be written

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

What if we change the norm?

This update has a closed-form solution using the dual norm $\|\cdot\|_*$,

$$x^{k+1} = x^k + \gamma_k \|g^k\|_* \operatorname{lmo}(g^k)$$

where lmo is the *linear minimization oracle*:

$$\operatorname{lmo}(g^k) \in \underset{s \in \mathcal{D}}{\operatorname{argmin}} \langle g^k, s \rangle = -\partial \|g^k\|_*$$

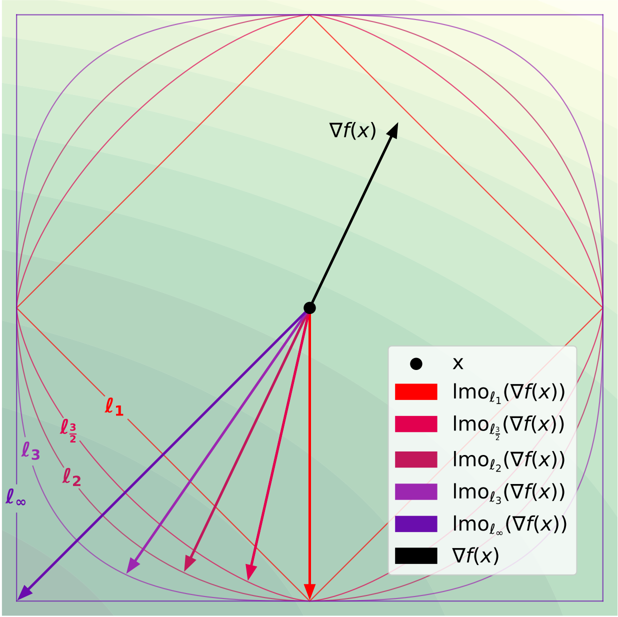
and $\mathcal{D}$ is the unit-ball for the norm $\|\cdot\|$.

Key insight: If we can compute the linear minimization oracle, we can do optimization with respect to a noneuclidean norm.

Given a norm $\|\cdot\|$, the associated *linear minimization oracle* (lmo) gives back a direction least aligned with its input,

$$\text{lmo}(g) \in \underset{\{s:\ \|s\| \leq 1\}}{\text{argmin}} \langle g, s \rangle.$$

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



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	**Steepest Descent ($-\|g\|_* \,\mathrm{lmo}(g)$)**
$\|\cdot\|_* = \|\cdot\|_2$	$-\|g\|_2 \left(-\frac{g}{\|g\|_2}\right) = g$

Steepest Descent in ℓ^2-norm recovers gradient descent/SGD.

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	**Steepest Descent ($-\|g\|_* \, \mathrm{lmo}(g)$)**
$\|\cdot\|_* = \|\cdot\|_1$	$-\|g\|_1 \, (-\mathrm{sign}(g)) = \left(\sum_i g_i\right) \mathrm{sign}(g)$

Steepest Descent in ℓ^∞-norm recovers sign descent.

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	$\mathrm{lmo}(g) = -UV^T$ where $g = U\Sigma V^T$ (reduced SVD)
Dual Norm	**Steepest Descent ($-\|g\|_*$ lmo(g))**
$\|\cdot\|_* = \|\cdot\|_{\mathrm{Nuc}}$	$-\|g\|_{\mathrm{Nuc}}\left(-UV^T\right) = \left(\sum_i \sigma_i(g)\right)\left(UV^T\right)$

Steepest Descent in $\|\cdot\|_{\mathrm{op}}$ recovers spectral descent/Muon.

LMOs for product sets

In the case where $x = [X_1, \dots, X_L]$ and we want to assign a norm $\|\cdot\|_{\{l\}}$ to each X_l for $l \in [L]$, we can take the *max*-norm,

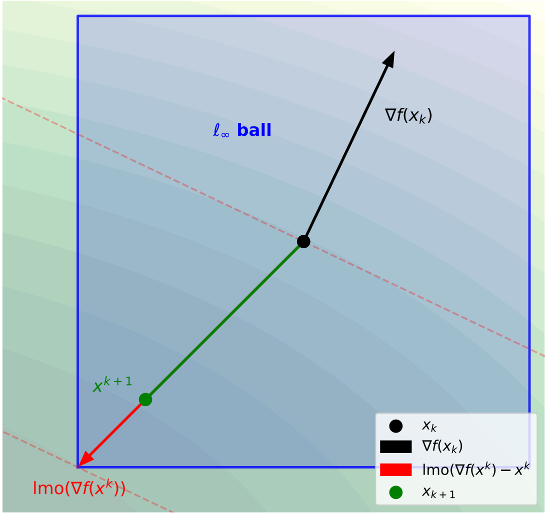
$$\|x\| := \max \left\{ \|X_1\|_{\{1\}}, \dots, \|X_L\|_{\{L\}} \right\}$$

so that the lmo with respect to this norm is *separable* across the X_l:

$$\mathrm{lmo}(g) = \mathrm{lmo}([g_1, \dots, g_L]) = [\underset{\{1\}}{\mathrm{lmo}}(g_1), \dots, \underset{\{L\}}{\mathrm{lmo}}(g_L)]$$

with each $\mathrm{lmo}_{\{l\}}$ corresponding to the lmo over the ball induced by the norm $\|\cdot\|_{\{l\}}$.

Conditional Gradient Algorithm

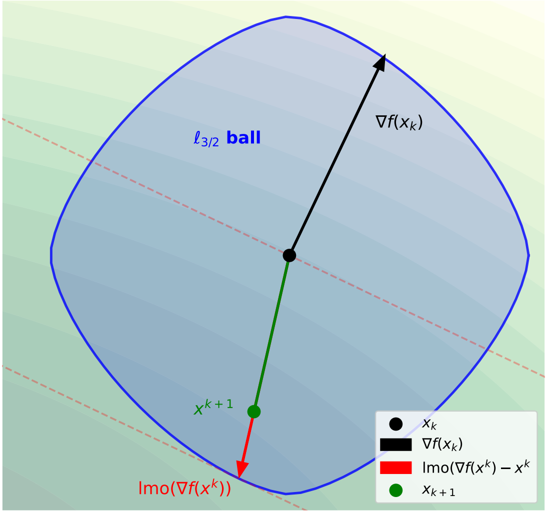


The conditional gradient algorithm (also known as the Frank-Wolfe algorithm) solves constrained optimization problems:

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

Conditional Gradient (CG):

Input: $x_0 \in \mathcal{D}$, step sizes $\{\gamma_k\}$ where $\gamma_k \in [0, 1]$, horizon $n \in \mathbb{N}^*$
for $k = 0, 1, \ldots, n-1$ **do**
  $s^k = \mathrm{lmo}(\nabla f(x^k))$
  $v^k = s^k - x^k$
  $x^{k+1} = x_k + \gamma_k v^k$
Output: x^n



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

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

Conditional Gradient (CG):

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

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

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

$\quad v^k = s^k - x^k$

$\quad x^{k+1} = x_k + \gamma_k v^k$

Output: x^n

Plan

- Motivation
- Feature Learning and μP
- Noneuclidean Optimization
- **(unconstrained) Stochastic Conditional Gradient**
- A natural geometry for neural networks
- Empirical Results
- Theoretical Results

Instead of Steepest Descent

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

which scales the update by $\|\nabla f(x^k)\|_*$, we can directly use

$$x^{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \langle g^k, x - x^k \rangle + \iota_{\gamma_k \mathcal{D}}(x - x^k)$$

to get

$$x^{k+1} = x^k + \gamma_k \mathrm{lmo}_{\mathcal{D}}(\nabla f(x^k)).$$

Related to Frank-Wolfe/Conditional Gradient and Generalized Matching Pursuit algorithms.

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

Input: $x^0 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, 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 (for the analysis).

- Momentum reduces variance in stochastic setting.
- uSCG solves the problem $\min_{x \in \mathbb{R}^d} f(x)$ while SCG solves the problem $\min_{x \in \mathcal{D}} f(x)$ where $\mathcal{D}$ is the unit ball of the norm.
- The direction s^k has fixed norm.
- SCG is "just" uSCG with weight decay

Weight Decay and SCG

We know 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 really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better").

Weight Decay and SCG

We know 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 really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better").
In a noneuclidean setting, this point is *critical* because the lmo is nonlinear.

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

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

Weight Decay and SCG

We know 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 really are equivalent up to a rescaling/renaming of constants (but decoupled is known to work "better").
In a noneuclidean setting, this point is *critical* because the lmo is nonlinear.

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

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 step size γ and Weight Decay λ!

Plan

- Motivation
- Feature Learning and μP
- Noneuclidean Optimization
- (unconstrained) Stochastic Conditional Gradient
- **A natural geometry for neural networks**
- Empirical Results
- Theoretical Results

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.

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

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

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

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

- Spectral Feature learning suggests taking the RMS norm on the input and output spaces of intermediary layers.
  $\to$ leads to a scaled $\ell^2 \to \ell^2$ operator norm $\|\cdot\|_{\mathrm{op}}$ on weight matrices:

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

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

- 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

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

- Spectral Feature learning suggests taking the RMS norm on the input and output spaces of intermediary layers.
  $\to$ leads to a scaled $\ell^2 \to \ell^2$ operator norm $\|\cdot\|_{\mathrm{op}}$ on weight matrices:

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

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

The first and final layers require more thought!

Norms for input 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

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

on this restricted domain.

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

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

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

For image domains, we use the RMS norm which gives the scaled operator norm for the initial layer.

Norms for output layers

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

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)	$\{W_\ell\}_{\ell\in[2,\ldots,L-1]}$	W_L			b_ℓ
Norm	RMS $\to$ RMS	RMS $\to$ RMS	RMS $\to$ RMS	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)

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
SCION

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

Plan

- Motivation
- Feature Learning and μP
- Noneuclidean Optimization
- (unconstrained) Stochastic Conditional Gradient
- A natural geometry for neural networks
- **Empirical Results**
- Theoretical Results

3B NanoGPT Training

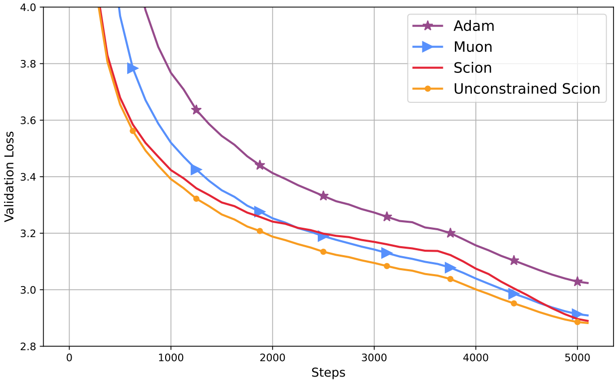
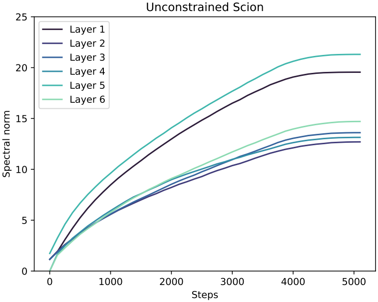


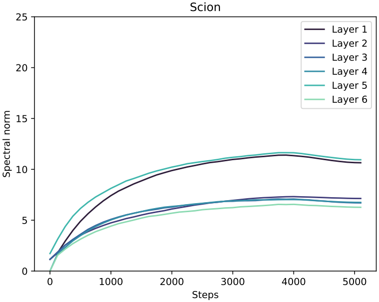
Table 5. Validation loss on a 3B parameter GPT model.

Adam	Muon	Unconstrained Scion	Scion
3.024	2.909	**2.882**	2.890

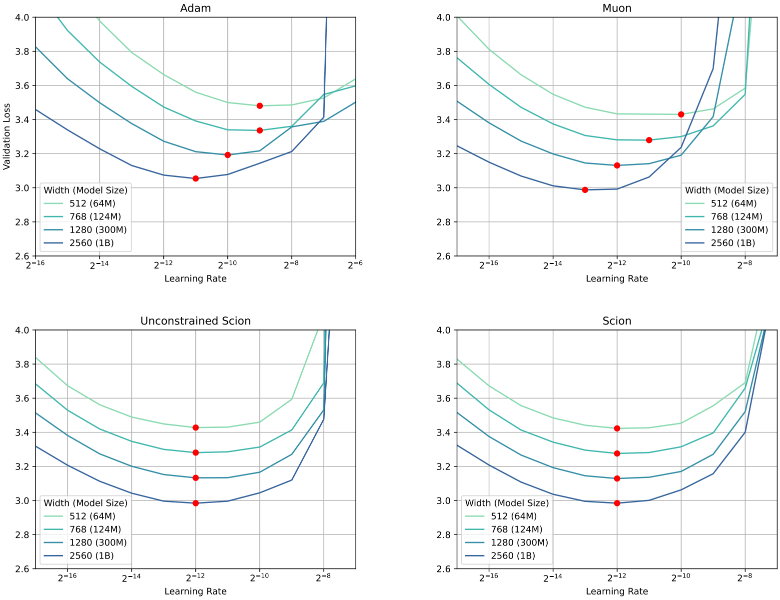
Let ρ be the radius of the set $\mathcal{D}$ that is used to define lmo. Both uSCG and SCG provide control over the norm of the output $\bar{x}^n$:

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

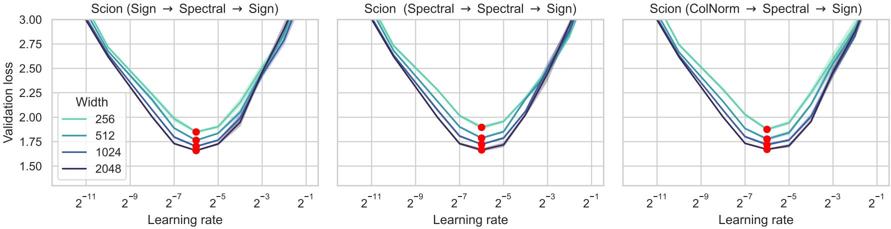


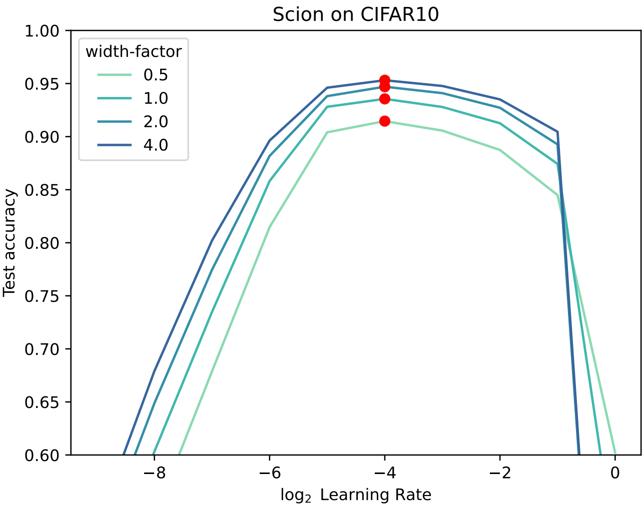


Hyperparameter Transfer: GPT Training



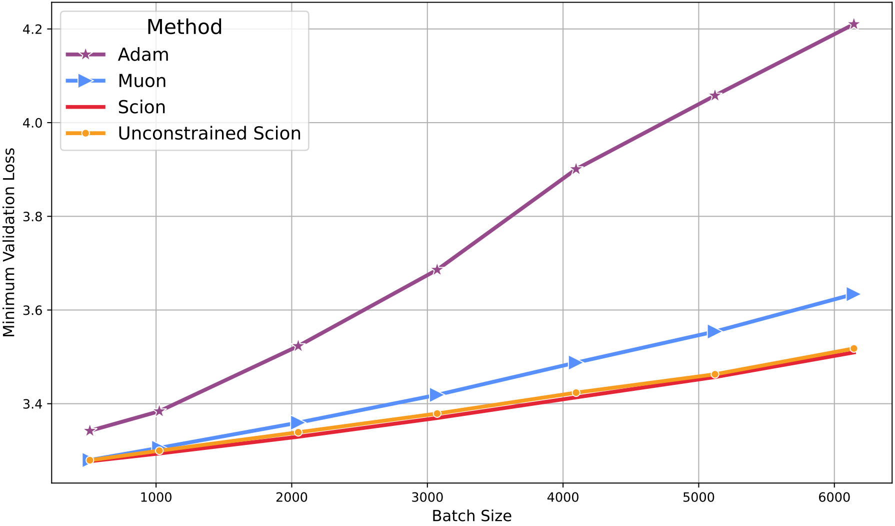
Shallow (3 layers) GPT on Shakespeare dataset.



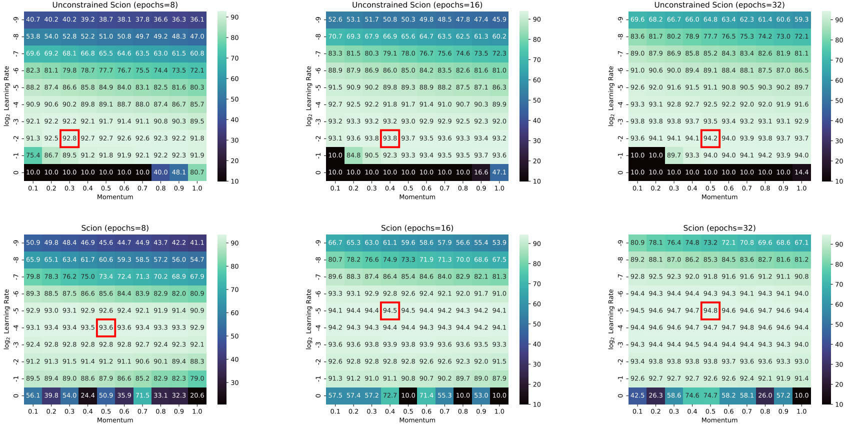


Optimal step size transfer across width in a convolutional NN trained to classify with CIFAR10.

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



Tuning the momentum on CIFAR10



Plan

- Motivation
- Feature Learning and μP
- Noneuclidean Optimization
- (unconstrained) Stochastic Conditional Gradient
- A natural geometry for neural networks
- Empirical Results
- **Theoretical Results**

To analyze the algorithm, we consider the class of problems

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

where

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

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

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

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

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 step size $\gamma = \frac{1}{\sqrt{n}}$. 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 step size $\gamma = \frac{1}{\sqrt{n}}$. Then, for all $u \in \mathcal{D}$,

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

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

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 step size $\gamma = \frac{3}{4n^{3/4}}$. 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 step size $\gamma = \frac{3}{4n^{3/4}}$. Then, for all $u \in \mathcal{D}$,

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

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

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 $\|\cdot\|_{\mathrm{op}}$	$-UV^T$, $d = U\Sigma V^T$

Our framework generalizes these algorithms through norm selection and momentum parameter.

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)

Paper

arXiv:2502.07529, also at ICML 2025 (Spotlight)

arXiv > cs > arXiv:2502.07529

Search...

Help | A

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

Currently working on extensions as well :)

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

Input: $x^0 \in \mathcal{D}$, step sizes $\{\gamma_k\}$, 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.