OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Applications of fractional calculus in learned optimization

Teodor-Alexandru Szente TEODOR.SZENTE@IMAR.RO
Institute of Mathematics of the Romanian Academy

James Harrison JAMESHARRISON@GOOGLE.COM
Google DeepMind

Mihai Zanfir MIHAI.ZANFIR@NEWTON.RO
Newton

Cristian Sminchisescu SMINCHISESCU@GOOGLE.COM
Google DeepMind, Institute of Mathematics of the Romanian Academy

Abstract

Fractional gradient descent has been studied extensively, with a focus on its ability to extend traditional gradient descent methods by incorporating fractional-order derivatives. This approach allows for more flexibility in navigating complex optimization landscapes and offers advantages in certain types of problems, particularly those involving non-linearities and chaotic dynamics. Yet, the challenge of fine-tuning the fractional order parameters remains unsolved. In this work, we demonstrate that it is possible to train a neural network to predict the order of the gradient effectively.

1. Introduction

In conventional first order optimization, the target function is typically approximated as locally linear using a Taylor expansion. It is possible to benefit from nonlinear approximations that capture the behavior of the function over a larger vicinity, offering a more accurate representation than local linear approximations. Fractional gradient descent methods were developed to take advantage of such approximations [6, 17, 20]. As shown in [9] they can greatly improve the convergence rate of the gradient descent algorithm in the convex case. These methods rely on the concept of *fractional derivatives*. The fractional derivative can be thought as an "interpolation" between two conventional derivatives. For example, the half derivative (i.e. fractional order $\alpha = 0.5$), denoted as $\frac{d^{0.5}f}{dx^{0.5}}$, represents an interpolation between the function f itself and its first derivative. However, little insight into determining the optimal fractional order for a specific problem is shown. Adaptive methods have been developed [11, 13] but they depend on additional hyper-parameters (e.g bounds limits, terminal points). Meanwhile, in the field of learned optimization, improvements have been made for fine tuning expressive optimizers [7, 8, 14]. In this paper, we illustrate how learned optimization can be employed to fine-tune the fractional order.

1.1. Fractional Calculus

The fractional derivative can be represented as a non-integer extension of the Cauchy formula for repeated integration [1, 16]

$$I^{\alpha} f(x) = \frac{1}{\Gamma(\alpha)} \int_{a}^{x} (x-t)^{\alpha-1} f(t)\, dt \tag{1}$$

$$D^{\alpha} f(x) = I^{-\alpha} f(x) \tag{2}$$

However, for negative α values, the first equation is undefined. One approach to circumvent this issue is to make use of the ceiling function. For instance, to compute $D^{1.2}$, we can first take the second derivative and then integrate with an order of 0.8 . This allows us to compute fractional derivatives based on negative α values

$$D^{\alpha} f(x) = \frac{d^{[\alpha]}}{dx^{[\alpha]}} (I^{[\alpha]-\alpha} f(x)) \tag{3}$$

By substituting I, we derive the Riemann–Liouville (RL) fractional derivative [10]

$$D^{\alpha} f(x) = \frac{1}{\Gamma([\alpha]-\alpha)} \frac{d^{[\alpha]}}{dx^{[\alpha]}} \int_{a}^{x} (x-t)^{[\alpha]-\alpha-1} f(t)\, dt \tag{4}$$

The properties of the Riemann–Liouville (RL) derivative have been extensively studied, and numerous other formulations have been proposed. One major drawback of this specific formulation is the handling of constant functions, i.e. $D^{\alpha} c = \frac{c x^{-\alpha}}{\Gamma(1-\alpha)} \neq 0$. To circumvent this, we can reverse the order in which differentiation and integration are applied in (3) [15]

$$D^{\alpha} f(x) = I^{[\alpha]-\alpha} (\frac{d^{[\alpha]} f}{dx^{[\alpha]}}(x)) \tag{5}$$

$$D^{\alpha} f(x) = \frac{1}{\Gamma([\alpha]-\alpha)} \int_{a}^{x} (x-t)^{[\alpha]-\alpha-1} \frac{d^{[\alpha]} f}{dx^{[\alpha]}}(t)\, dt \tag{6}$$

This is called the Caputo derivative, and while it is widely used in many applications, it may be too computationally expensive when applied to optimization tasks. A more direct approach is to generalize the finite difference form obtaining the Grünwald–Letnikov derivative [15]

$$D^{\alpha} f(x) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{k=0}^{\lfloor \frac{x}{h} \rfloor} (-1)^k \binom{\alpha}{k} f(x - kh) \tag{7}$$

1.2. Geometric interpretation

One way to conceptualize the derivative is as an approximation of a linear map near a given point of a function. Take for example $f : \mathcal{R}^2 \to \mathcal{R}^2$

$$f(x, y) = (x^2 - y^2, 3xy) \tag{8}$$

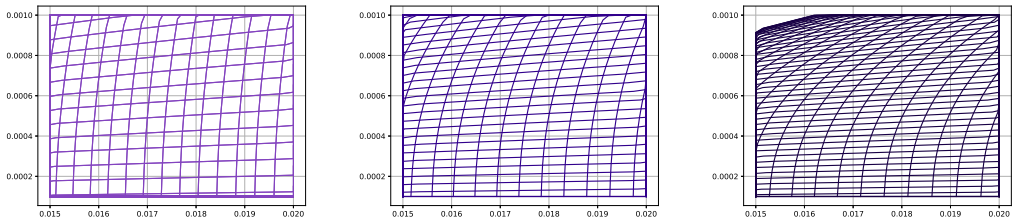
Given the Jacobian matrix defined as:

$$\mathbf{J}_f = \begin{pmatrix} 2x & -2y \\ 3y & 3x \end{pmatrix} \tag{9}$$

we define a transformation $T_{\mathbf{J}_f}(x) = \mathbf{J}_f x$. In this context, T serves as the best linear approximation to the function f at a given point x. However when defining the fractional Jacobian matrix:

$$\mathbf{J}_f^\alpha = \begin{pmatrix} x^{-\alpha}\left(\frac{2x^2}{\Gamma(3-\alpha)} - \frac{y^2}{\Gamma(1-\alpha)}\right) & y^{-\alpha}\left(\frac{x^2}{\Gamma(1-\alpha)} - \frac{2y^2}{\Gamma(3-\alpha)}\right) \\ \frac{3yx^{1-\alpha}}{\Gamma(2-\alpha)} & \frac{3xy^{1-\alpha}}{\Gamma(2-\alpha)} \end{pmatrix} \tag{10}$$

this changes. The transformation $T_{\mathbf{J}_f^\alpha}$ provides a linear approximation only when $\alpha = 1$. As the difference between α and 1 increases, the non-linear behavior in $\mathbf{J}_f^\alpha$ becomes more pronounced, deviating further from the linear approximation. While losing linear properties, the fractional Jacobian evolves to follow the global curvature of the function more closely. This enables a more faithful approximation of the objective function, potentially capturing more complex behaviors



(*a*) (*b*) (*c*)

Figure 1: Grid transform near origin for **(a)** $T_{\mathbf{J}_f}$ **(b)** $T_{\mathbf{J}_f^{1.2}}$ **(c)** $T_{\mathbf{J}_f^{1.25}}$

2. Methodology

2.1. Meta-learning on classical functions

We start by learning to optimize on a collection of classical functions[1]. We consider each function to be parameterized by a state vector $\mathbf{X}_t$. We train a neural network $\mathcal{F}_\theta$, that takes as input the current state, normalized gradients, magnitudes of the gradient and Fourier features $\gamma(\mathbf{X}_t)$ [19], and outputs the order of the fractional derivative, $\alpha_t \in [0, 1]$ and the magnitude of the update step, η_t, which are then used to compute the next state $\mathbf{X}_{t+1}$

$$F_\theta(\mathbf{X}_t, \frac{\nabla f(\mathbf{X}_t)}{|\nabla f(\mathbf{X}_t)|}, |\nabla f(\mathbf{X}_t)|, \gamma(\mathbf{X}_t)) = (\alpha_t, \eta_t) \tag{11}$$

Afterwards we can compute the next target function state based on the predicted order and magnitude

$$\mathbf{X}_{t+1} = \mathbf{X}_t - \eta D^\alpha f(\mathbf{X}_t) \tag{12}$$

where D^α is approximated by first order truncated Taylor expansion,

$$D^\alpha f(\mathbf{X} + \Delta\mathbf{X}) = \binom{\alpha}{0} \frac{1}{\Gamma(1-\alpha)} f(\mathbf{X}) + \binom{\alpha}{1} \frac{1}{\Gamma(\alpha)} \Delta\mathbf{X} \frac{\partial f}{\partial \mathbf{X}}. \tag{13}$$

1. https://www.sfu.ca/ ssurjano/optimization.html

Then using AdamW [12] we train the neural network and optimize the objective function

$$\mathcal{L}_\theta = \log(f(\mathbf{X}_{t+1})) - \log(f(\mathbf{X}_t)) \tag{14}$$

We train this neural network in 2 regimes:

- **with supervision** at each step sample from **all** the classical functions and merge them in one single batch.
- **without supervision** at each step sample from classical functions **except** the target function.

During training, we experimented with using different functions in every batch to help our network generalize better. This indeed helped, albeit only after including Fourier features [19]. Without them it was difficult for the network to learn high frequency details in low level dimensions.

2.2. Chaotic systems

As there are more than one definitions of chaotic systems, it is easier for us to describe chaotic systems by their properties. Most importantly for our work is the sensitivity of the system to the initial conditions. Take for example the target function[2]

$$f(x) = \log(x^2 + 1 + \sin(3x)) + 1.5 \tag{15}$$

Applying classical gradient descent on f exhibits chaotic behavior (shown in fig. 2) as a small change in the initial condition (e.g. learning rate, momentum) induces a big change in the final value x over a large enough horizon. We can try to apply our proposed fractional gradient descent, but computing the fractional gradient already presents significant challenges from a numerical perspective. We presented a way to approximate it using a first order Taylor expansion in (13), but, for more complex problems, this is not accurate enough. Therefore, in the context of chaotic systems we propose to analyze the Fractional Gradient Flow (**FGF**) equation by extending (12) in continuous time

$$\frac{d^\alpha \mathbf{X}(t)}{dt} = -\nabla f(\mathbf{X}(t)) \tag{16}$$

There are several methods available to approximate fractional differential equations (FDEs) [3, 9, 18]. In [3], the following approximation is proposed:

$$\mathbf{X}_{k+1} = -\eta^\alpha \nabla f(\mathbf{X}_k) + \left(\alpha \mathbf{X}_k - \frac{\alpha(\alpha-1)}{2}\mathbf{X}_{k-1} + \ldots + \frac{\alpha(\alpha-1)\ldots(\alpha-k)}{(k+1)!}(-1)^k \mathbf{X}_0\right) \tag{17}$$

This discretization form has a convergence rate of $O(\frac{1}{t^\alpha})$ for locally Lipschitz continuous functions. We aim to use this discretization of FGF equations to reproduce the experiment defined in [8], where the parameters of a Lorenz system need to be optimized. We compare against various other gradient estimators methods and classic backpropagation through time, using the scheme presented in eq. 17. The Lorenz system is defined by the following set of three coupled nonlinear differential equations:

$$\frac{dx}{dt} = \sigma(y - x); \frac{dy}{dt} = x(\rho - z) - y; \frac{dz}{dt} = xy - \beta z \tag{18}$$

where x, y, and z are functions of time t, and σ, ρ, and β are parameters that control the behavior of the system. Our goal is to optimize for the control parameters $\mathbf{X} = [\log(\sigma), \log(\rho)]$, starting from an initial state $s_0 = (x_0, y_0, z_0) = (1.2, 1.3, 1.6)$

2. https://lukemetz.com/exploring-hyperparameter-meta-loss-landscapes-with-jax/

3. Results

3.1. Meta-learning on classical functions

We compare our method against a multitude of optimizers: a general purpose learned optimizer (VeLO), classic gradient descent and adaptive methods. For testing, we randomly sample 1000 starting points from the target function domain and report the convergence rates for a maximum of 100 steps of optimization. We consider a solution to be converged if the value of the function at that point is at a maximum distance of $\epsilon = 10^{-3}$ from the global minimum. We also report the average number of steps to reach the solution. For learning rate based methods we search over learning rates between 10^{-1} and 10^{-6} and present the run with the best results.

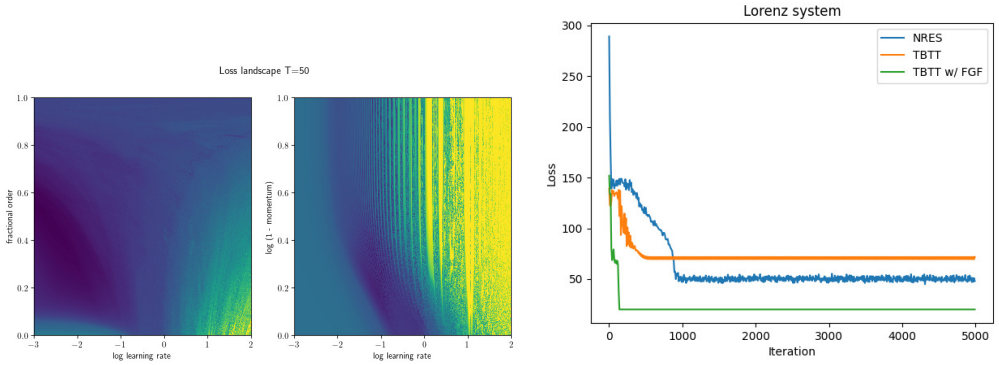
Optimizer	Convergence Rate	Truncated trajectory length
(1) GD	0.60%	994.03
(2) Adam	1.60%	985.15
(3) AdamW	1.60%	985.16
(4) RMSProp	3.50%	966.09
(5) Adafactor	1.10%	989.27
(6) Adagrad	0.90%	991.48
(7) VeLO [14]	0.30%	997.05
(8) Ours w supervision	99.20%	12.156
(9) Ours w/o supervision	71.80%	321.37

Table 1: Performance analysis on Rosenbrock 2D: our trained neural network optimizer predicts updates that closely resemble those of a second-order method. This behavior aligns with expectations for learned optimizers. (8) Our optimizer trained only on the target function with supervision (9) Our optimizer trained on other classical functions **except** the target function.

3.2. Chaotic systems

We conducted two experiments, as described in section 2.2. The first one involves a comparison of the loss landscape for the function in (15), between classical gradient descent and the FGF method. This experiment can be seen in fig. 2(a).

The second experiment regarding the optimization of the Lorenz system, compares the update step from (17) with the update step generated by classical backpropagation through time (**TBTT**) and the update step generated by the gradient estimator presented in [8], **NRES**. Changing the update step from classical gradient descent to fractional gradient descent makes **TBTT** perform the best, as it can be seen in fig. 2(b). This approach converges faster and is more stable. Although this is a toy problem, we believe that there may be potential for such techniques in real-world applications. Currently, due to the chaotic nature of TBTT, Evolutionary Search gradients are used, such as NRES. These methods, although stabilize the exploration of the loss landscape, suffer from poor convergence rates.



(*a*) (*b*)

Figure 2: a) Loss surface for the function from eq. 15 (left) classical gradient descent searching over momentum decay and learning rates; (right): our FGF method searching over fractional orders and learning rates. b) Loss convergence for the Lorenz optimization problem defined in [8] comparing NRES and TBTT, with and without the FGF discretization scheme.

4. Limitations

We investigated various forms of the fractional derivative, which show promising results in small-dimensional problems such as classical optimization tasks and traditional chaotic systems. Although their application is currently limited in meta-training, we believe that further research could make fractional calculus a viable approach. We believe that one of the problems of extending this work to higher-dimensional problems, is that a single fractional order might not accurately describe all the dimensions. In recent works, Transformers were used to overcome the problem of dimensionality in learned optimization [4], but the inference time is slow. We aim to take advantage of other advances in the field, such as faster attention mechanisms [2] or SSM [5] and make this approach more feasible in the future. In meta-learning, the inner and outer dynamics of the system differ. In this work we focused solely on the inner dynamics which sometimes behaves akin to a chaotic system. Further investigation is needed to extend this work also to the outer dynamics.

Acknowledgments: This work was supported in part by project CNCS-UEFISCDI (PN-III-P4-PCE-2021-1959). The authors thank Andrei Zanfir and Mykhaylo Andriluka for their insightful feedback throughout various stages of this project.

References

[1] Augustin-Louis Cauchy. *Résumé des leçons données à l'école royale polytechnique sur le calcul infinitésimal*, volume 1. Imprimerie royale, 1823. Reprint: Completes II(4), Gauthier-Villars, Paris.

[2] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[3] Furkan Nur Deniz, Baris Baykant Alagoz, Nusret Tan, and Murat Koseoglu. Revisiting four approximation methods for fractional order transfer function implementations: Stability preservation, time and frequency response matching analyses. *Annual Reviews in Control*, 49:239–257, 2020. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2020.03.003.

[4] Erik Gärtner, Luke Metz, Mykhaylo Andriluka, C Daniel Freeman, and Cristian Sminchisescu. Transformer-based learned optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11970–11979, 2023.

[5] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.

[6] Pham Viet Hai and Joel A. Rosenfeld. The gradient descent method from the perspective of fractional calculus. *Mathematical Methods in the Applied Sciences*, 44(7):5520–5547, 2021. doi: https://doi.org/10.1002/mma.7127.

[7] James Harrison, Luke Metz, and Jascha Sohl-Dickstein. A closer look at learned optimization: Stability, robustness, and inductive biases. *Advances in Neural Information Processing Systems*, 35:3758–3773, 2022.

[8] Oscar Li, James Harrison, Jascha Sohl-Dickstein, Virginia Smith, and Luke Metz. Variance-reduced gradient estimation via noise-reuse in online evolution strategies. *Advances in Neural Information Processing Systems*, 36:45489–45501, 2023.

[9] Shu Liang, Le Yi Wang, and George Yin. Fractional differential equation approach for convex optimization with convergence rate analysis. *Optim. Lett.*, 14(1):145–155, 2020. doi: 10.1007/S11590-019-01437-6.

[10] J Liouville. Note sur une formule pour les différentielles à indices quelconques, à l'occasion d'un mémoire de m. tortolini. *Journal de mathématiques pures et appliquées*, 20, 1855.

[11] Jiaxu Liu, Song Chen, Shengze Cai, and Chao Xu. The novel adaptive fractional order gradient decent algorithms design via robust control. *ArXiv*, abs/2303.04328, 2023.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[13] Weipu Lou, Wei Gao, Xianwei Han, and Yimin Zhang. Variable order fractional gradient descent method and its application in neural networks optimization. In *2022 34th Chinese Control and Decision Conference (CCDC)*, pages 109–114, 2022. doi: 10.1109/CCDC55256.2022.10033456.

[14] Luke Metz, James Harrison, C. Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, and Jascha Sohl-Dickstein. VeLO: Training Versatile Learned Optimizers by Scaling Up. *arXiv e-prints*, art. arXiv:2211.09760, November 2022. doi: 10.48550/arXiv.2211.09760.

[15] K. Oldham and J. Spanier. *The Fractional Calculus Theory and Applications of Differentiation and Integration to Arbitrary Order*. ISSN. Elsevier Science, 1974. ISBN 9780080956206.

[16] Bernhard Riemann. *Versuch einer allgemeinen Auffassung der Integration und Differentiation*. Gesammelte Werke, Lipzig, 1876. ed. publ. posthumously.

[17] Yeonjong Shin, Jérôme Darbon, and George Em Karniadakis. A caputo fractional derivative-based algorithm for optimization. *arXiv preprint arXiv:2104.02259*, 2021.

[18] Zeng Liao Shu Liang, Cheng Peng and Yong Wang. State space approximation for general fractional order dynamic systems. *International Journal of Systems Science*, 45(10):2203–2212, 2014. doi: 10.1080/00207721.2013.766773.

[19] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[20] Zhiguang Zhu, Ang Li, and Yong Wang. Study on two-stage fractional order gradient descend method. In *2021 40th Chinese Control Conference (CCC)*, pages 7960–7964, 2021. doi: 10.23919/CCC52363.2021.9549324.