MoXCo:How I learned to stop exploring and love my local minima?

Esha Singh[1*] **Shoham Sabach**[2] **Yu-Xiang Wang**[3*]
[*]UC Santa Barbara [2]Technion–Israel Institute of Technology
[1]esingh@ucsb.edu [2]ssabach@technion.ac.il [3]yuxiangw@cs.ucsb.edu

Abstract

Deep Neural Networks (DNNs) are well-known for their generalization capabilities despite overparameterization. This is commonly attributed to the optimizer's ability to find "good" solutions within high-dimensional loss landscapes. However, widely employed adaptive optimizers, such as ADAM, may suffer from subpar generalization. This paper presents an innovative methodology, *MoXCo*, to address these concerns by designing adaptive optimizers that not only expedite exploration with faster convergence speeds but also ensure the avoidance of over-exploitation in specific parameter regimes, ultimately leading to convergence to good solutions.

1 Introduction

Vapnik [1999], Vapnik and Chervonenkis [1982] demonstrated that if any problem is learnable, it is also learnable by solving Empirical Risk Minimization (ERM). This influenced generations of researchers to treat the statistical problem of *generalization* and the computational problem of *optimization*, i.e., solving ERM, completely separately. However, in modern deep learning (DL) models, the boundaries between optimization and generalization have become less distinct. Our primary goal should no longer be optimizing for the sake of minimizing a given objective function as much as possible. Instead, we should think about how to design optimization algorithms that converge to good local stable regions (i.e. local minima) that generalizes better.

On a different note, the energy landscape in modern DL are remarkably intricate. There are several pitfalls that an optimizer such as SGD can get stuck into or explode from such as plateaus and cliffs, sharp local minima, flat stationary points etc. This motivated many strategies in training deep learning models, spanning architectural designs like residual connections & batch normalization, as well as optimization tricks such as ADAM, gradient clipping, and learning rate scheduling etc. We posit that many of these techniques aim at allowing the optimizer to survive these pitfalls, facilitating sufficient exploration in-order to converge to a favorable solution.

Our contributions are as follows. We propose a novel framework MoXCo - *Momentum Exploration and Commit* for making such determination. We introduce the notion of "*Inertial Adaptivity*" which is a two-step process. First, we leverage momentum to expedite any black-box first-order optimization method, facilitating exploration and attaining faster convergence compared to non-momentum-accelerated methods. Second, we engage in a strategic traversal across complex loss landscape while monitoring a *goodness score*. Once the *goodness score* satisfies a pre-defined criterion, we declare that we are in the proximity of a *good* solution and commit to converge by reducing momentum.

2 Related Work

Our framework comprises two components, elaborated in detail in sections 5.1, 5.2. The utilization of inertial force for momentum dates back to Polyak [1964]. Ochs et al. [2014] used this inertial force

Mathematics of Modern Machine Learning Workshop at NeurIPS 2023.

to support the use of a proximal gradient-type method for handling structurally induced regularization. Wang et al. [2023] uses inertial accelerated stochastic gradient methods to solve the low-rank CP decomposition problems. Similar to our second equation (2), the estimation of stochastic gradients on a slightly perturbed point, for non-smooth & non-convex problems has been extensively discussed by visionary work of Cutkosky et al. [2023].

Our second component mitigates unnecessary oscillations & divergence, while also offering insights on momentum parameter scheduling. There are various works on momentum schedules such as Wang et al. [2022], Xie et al. [2022] develop an approach for escaping saddle points & flat minima selection based on ADAM. There are very few works that recognize the need for a rapid exploration and committing phase such as us. While O'donoghue and Candes [2015] discusses adaptive restart techniques, it primarily applies to convex settings, whereas we operate in a non-convex, non-smooth context. Similarly, Zhou et al. [2020] focuses on proximal gradient parameter restarts for non-convex optimization, but our framework systematically indicates momentum hyper-parameter restarting based on local geometry. Additionally, Liu et al. [2023] employs Inertial Momentum in a federated learning setup for global convergence, whereas in our more generalized setting, inertial momentum is just one step in our two-step framework.

3 Problem Setup

We are interested in optimizing any function f: $\mathbb{R}^d \longrightarrow \mathbb{R}$. We make no assumptions about that differentiablity of f and it is non-convex and possibly non-smooth. We have access to stochastic or noisy f, and the learning problem is $\min_{x \in \mathcal{X}} f(\mathbf{x}) = \mathop{\mathbb{E}}_{z \sim D}[F(\mathbf{x}, \mathbf{z})]$ where $\mathcal{X} \in \mathbb{R}^d$ and $\mathbf{z}$ is any random vector (noise) that depends upon the unknown distribution D. The objective/loss function denoted by f can be regularized with non-smooth regularizer as is the case in quantization tasks [Section 6]. We also assume that we have access to any black-box first-order optimizer. We use $||.||$ to denote l_2 norm for vectors and observe gradient of f w.r.t to $\mathbf{x}$ as $\nabla_x f_t(\mathbf{x})$. The Hessian matrix of f at input point $\mathbf{x}$ & time step t is denoted by $\nabla_x^2 f_t(\mathbf{x})$ or $H(f)(\mathbf{x})$ and its l_2 norm is $||\nabla_x^2 f_t(\mathbf{x})||_2$. λ_{max} denotes largest eigenvalue of the Hessian of the objective function f. We also assume that $\nabla_x^2 f_t(\mathbf{x}) <$ B, where B is any constant depending on network architecture.

4 Black-Box Optimization

Popular black-box optimization methods fall into categories like derivative-free (DFO) Shahriari et al. [2015], derivative-based Sarafian et al. [2020] Wright [2006], and model-based. While effective for non-differentiable functions, our generalized use-case, where we assume non-differentiable objective function and discrete parameter space render existing local search methods (derivative-based methods) impractical. We have accessible gradient information even for non-differentiable objectives. Therefore, we introduce a novel gradient-based black-box optimization approach surpassing DFO in precision & computational efficiency (& for gradient-based methods without the additional overhead of second-order derivatives Wright [2006]. Unlike conventional techniques relying solely on gradients, our method incorporates insights from local curvature properties, significantly enhancing global optimization capabilities.

Although, efforts have been made to enhance exploration efficiency Malviya et al. [2023] Liu et al. [2023], yet there persists a notable gap in addressing the need for stopping criteria. Lewkowycz et al. [2020] develops a connection between large learning rate & flatness of minima in models trained via SGD. For models incorporating momentum, training within a range of sufficiently large learning rates or range of hyper-parameter choices that are not extensively tuned for, poses a challenge due to the potential risk of divergence or unwanted oscillations. We argue that while gradient information offers insights into the stationary properties of a loss landscape, it remains insufficient for definitively deciding when to conclude the exploration phase. But second-order methods excel at characterizing such stationary points, whereas existing first-order methods fall short.

5 Proposed Framework

Below we discuss each of the two components separately.

5.1 Inertial Proximal Algorithm for Promoting Exploration

How do we promote exploration in deep learning training? One of the oldest idea is to leverage the inertial force of the "heavy-ball" algorithm [Polyak, 1964]. The method is known as "SGD with momentum" in DL optimization. It has been empirically observed that the momentum helps to accelerate the pace of SGD in low-curvature regions (e.g., plateaus) [Sutskever et al., 2013] and helps to escape saddle points and shallow local minima [Wang et al., 2021]. The vanilla SGD with momentum, however, falls short in three aspects. First, it is unclear whether it still works when the objective function is non-differentiable — which is the case for every ReLU-activated neural networks. Second, it does not support the use of a proximal gradient-type method to handle structural inducing regularization. Third, it does not take advantage of other tricks in deep learning optimizations, e.g., Adam, which are often delicate choices that enable the effective training of certain families of neural architecture. The second problem was solved in the seminal work of Ochs et al. [2014] which establishes strong convergence guarantees for proximal versions of heavy-ball algorithm. To address the first issue, we proposed adding second momentum that slightly perturbs the location to evaluate the gradient as follows.

$$\mathbf{u}_t = \boldsymbol{\theta}_t + \alpha(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}) \tag{1}$$

$$\mathbf{v}_t = \boldsymbol{\theta}_t + \beta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}) \tag{2}$$

$$\boldsymbol{\theta}_{t+1} = \mathrm{Prox}_g\left(\mathbf{u}_t - \eta\hat{\nabla} f(\mathbf{v}_t)\right) \tag{3}$$

where $\eta > 0$ is the learning rate and $\alpha, \beta \in [0, 1)$ are momentum coefficients. The second line changes the location to evaluate the gradient slightly from $\boldsymbol{\theta}_t$ to $\mathbf{v}_t$. This is related to the recently proposed online-to-non-convex conversion method [Cutkosky et al., 2023, Remark 10] but without the randomized smoothing. The extra momentum on $\mathbf{v}_t$ is particularly important because the proximal operator is very likely to return the subsequent iterate $\boldsymbol{\theta}_{t+1}$ on highly-special non-smooth points, even if these non-differentiable points are inside a measure zero set. For third problem, we apply the above algorithm to any DL optimizer via an interactive black-box fashion. In particular, we can return a different update for the iterates above by replacing $\hat{\nabla} f(\mathbf{v}_t)$ with the update Δ_t sent back to us by any optimizer, i.e., Δ_t : Optimizer$(\nabla f(\mathbf{v}_t), \mathbf{v}_t, \gamma)$ where γ represents any other inputs required for black-box optimizer, apart from $\nabla f(\mathbf{v}_t)$. Also, the optimizer is allowed to have memory from the previously observed $\nabla f(\mathbf{v}_i), \boldsymbol{\theta}_i$ for $i \in [t-1]$.

5.2 "Goodness" score and when to stop exploration

Having introduced inertial momentum, a mechanism that accelerates and promotes exploration, we now shift our focus to the second component: how to determine whether the current local neighborhood is worthy of our algorithm's commitment to converge to.

This is a daunting task as we are hoping to estimate a global property — whether this neighborhood is close to a solution that solves the population level stochastic optimization problem — using only local information, i.e., gradient and function value. This is impossible in general for non-convex problems. But deep learning is not a generic non-convex problem but one with many interesting structures and additional information (e.g., ideal loss value f_{target}, boundedness). With this information, we can use them to come up with a necessary condition for the "goodness" of a local neighborhood and use that as a promising heuristic to guide our optimizer in practice.

Specifically, we came up with the following "goodness" score of a parameter

$$\mathsf{Goodness}(\theta) = \exp\left(-\tau\left[||\nabla f(\theta)||_2^2 \;+\; \left(\lambda_{\max}(\nabla^2 f(\theta)) - \frac{1}{\tilde{\eta}}\right)_+ \;+\; |f(\theta) - f_{\text{target}}|\right]\right)$$

in which τ calibrates how sensitive the score is, and the exponential transformation ensures that the score $0 \le \mathsf{Goodness} \le 1$ as the term in the square brackets is nonzero. A goodness score being closer to one means that we have found a solution that is approximately stationary, flatter than Edge of Stability, and close to an ideal target.

Theorem 1 *Let θ satisfies that (a) it is a local minimum of f; (b) it is a stable fixed point of gradient descent with learning rate $\tilde{\eta}$, (c) $|f(\theta) - f_{target}| \le \epsilon$. Then $\mathsf{Goodness}(\theta) \ge \exp(-\tau\epsilon)$.*

Table 1: Optimization Landscapes. f denotes any objective function parameterized by θ_t. We enumerate maginutude of values for objective value, it's hessian & largest eignevalue at any θ_t

Indicator	Plateau	Cliff	Sharp	Saddle point
$f(\theta_t)$	large	large/small	small	large
$\|\|\nabla f(\theta_t)\|\|_2^2$	small	large	small	small
$\lambda_{max}(\nabla^2 f(\theta_t))$	small	small	large	small

Theorem 2 (EOS adjustment) *If we consider running Inertial momentum to optimize a quadratic objective function of form* $f(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$*, then based on Cohen et al. [2021], if we consider running vanilla gradient descent on the f(x) starting from any initalization and if* $(\mathbf{q}, a)$ *be an eigenvector/eigenvalue pair of* $\mathbf{A}$*, then if* $a > \frac{2(1+\alpha)}{\eta(1+2\beta)}$*, then the sequence* $\{\mathbf{q}^T\mathbf{x}_t\}$ *will diverge.*

A desired learning algorithm should have $\epsilon \leftarrow 0$ as the number of data points n gets larger, which means the Goodness score should converge to 1.

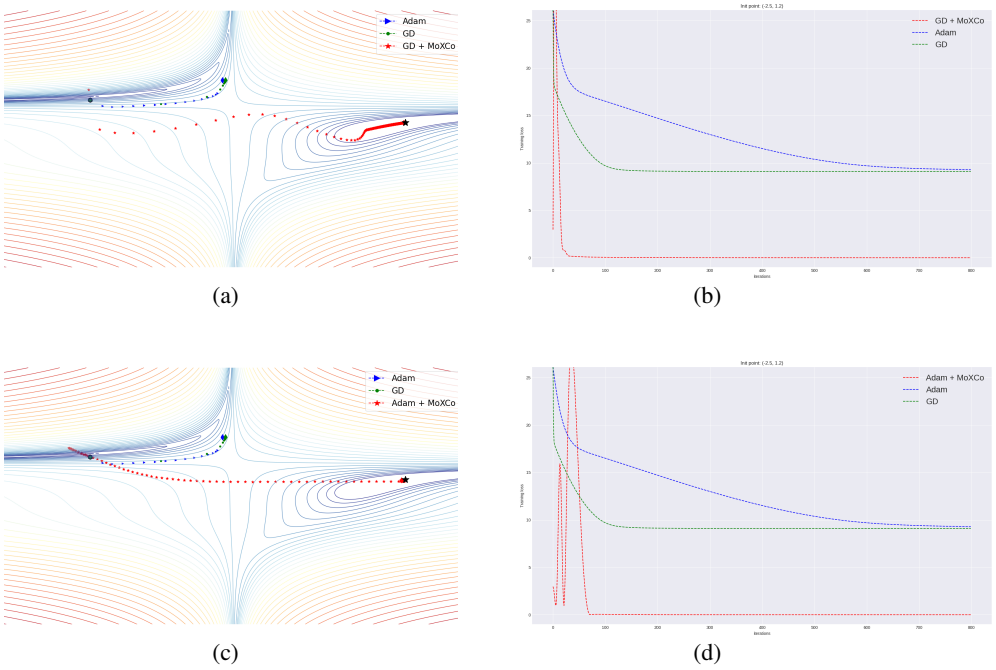
Our algorithm essentially works by monitoring this score and decide to reduce α, β when the score is above a threshold. The first-term in the square brackets is the standard measurement of stationarity. The second-term measures the θ's sharpness and penalizes any solution that is bigger than the designated "Edge of Stability" level of sharpness specified using an adjusted effective learning rate $\tilde{\eta}$ (Theorem 2, proof A). The third term measures the absolute difference between the current objective function to a problem-dependent target. For example, f_{target} should be 0 for classification tasks, and σ^2 for regression tasks. Table 1 shows why thresholding this score function is a good necessary condition for finding a "good solution".

6 Experiments

To illustrate our framework's behavior, we provide detailed results with toy examples. In Figure 1, we show optimization trajectory on Beale for our framework (red curve) for 800 steps. In Fig 1 (a) we plot the optimization trajectory wrt to MoXCo with GD update, Adam and GD. In Fig 1(b) we plot corresponding training loss curves for all optimizers. We can see that our method MoXCo not converges faster but also reaches an optimal minima for beale curve as compared to other optimizers which get stuck very quickly in nearest local minima, hence depicting aggressive exploration and optimal commit-converge strategy in our proposal. Similar behavior is reflected when MoXCo is used with Adam iterate. In all Fig 1, we use consistent and highest stable learning rate of $\eta = 0.005$ with $\alpha, \beta = 0.9, 0.9$ for (a) and $\alpha, \beta = 0.95, 0.5$ for (b). Our framework (red curve) effectively balances exploration & convergence despite significant momentum and a relatively large η. Even with a small $\eta = 0.001$ we maintain aggressive exploration & commit-to-converge to optima (additional details in Appendix A.2). Figure 2(b) presents the escape count analysis. As we can see our methods have higher count of points reaching global minima than other methods.

Another one of our target is to perform a focused comparison of the implicit biases exhibited by various optimizers towards flat regions within the loss landscape. To empirically investigate this, we conducted an escape count analysis on a 1D test function (Figure 2). This test function features two minima: one shallow and the other sharp. Figure 2(b) demonstrates that ADAM + Inertia(MoXCo) and GD + Inertia(MoXCo) (both denoting MoXCo method) surpass Adam & GD in escape count. Note, that we plot the results for only 1000 runs so we plot only the points which reached convergence in 1000 steps among 50 different initalizations. This experiment was conducted with 50 different initialization chosen at random. As we see in Figure 1(c) Adam + Inertial & GD + Inertial have a higher count of points reaching global as compared to ADAM & GD.

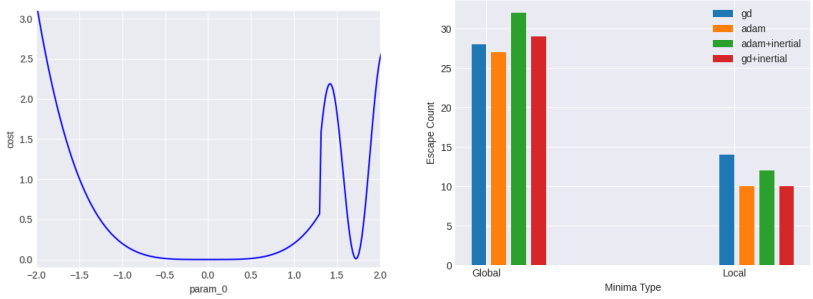
In future work, we will validate our framework on binary quantization, as it provides non-convex loss landscapes due to its non-smooth regularizer.



(a) (b)

(c) (d)

Figure 1: (a)(b) Toy example using Beale test function. The colored diamond markers indicate end points for all the optimizers. The black star indicates global minima at (3, 0.5) for the Beale function. The black circular marker indicates the same initialization point for both plots. While both Adam & GD get stuck in local minima, our method (ADAM+MoXCo or GD+MoXCo, denoting our method - contains both inertial momentum & parameter restart with goodness score) is able to explore further, commit & converge in order to reach a flatter optima.



(a) (b)

Figure 2: (a) Escape count analysis for all optimizes when run for 50 different initalizations. For each optimizer, we tracked how many times a point successfully navigated away from sharp minima to locate a flatter minima. The x-axis denotes two bins, distinguishing between the two types of minima in our example 1D test function A. As we can see our methods higher count of points reaching global minima than other methods.

References

Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

Ashok Cutkosky, Harsh Mehta, and Francesco Orabona. Optimal stochastic non-smooth non-convex optimization through online-to-non-convex conversion. *ICML-2023*, 2023.

Saber Elaydi. An introduction to difference equation. 01 2005.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism, 2020.

Yixing Liu, Yan Sun, Zhengtao Ding, Li Shen, Bo Liu, and Dacheng Tao. Enhance local consistency in federated learning: A multi-step inertial momentum approach, 2023.

Pranshu Malviya, Gonçalo Mordido, Aristide Baratin, Reza Babanezhad Harikandeh, Jerry Huang, Simon Lacoste-Julien, Razvan Pascanu, and Sarath Chandar. Promoting exploration in memory-augmented adam using critical momenta. *arXiv preprint arXiv:2307.09638*, 2023.

Peter Ochs, Yunjin Chen, Thomas Brox, and Thomas Pock. ipiano: Inertial proximal algorithm for non-convex optimization, 2014.

Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15:715–732, 2015.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Elad Sarafian, Mor Sinay, Yoram Louzoun, Noa Agmon, and Sarit Kraus. Explicit gradient learning for black-box optimization. In *ICML*, pages 8480–8490, 2020.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

Vladimir N Vapnik and A Ya Chervonenkis. Necessary and sufficient conditions for the uniform convergence of means to their expectations. *Theory of Probability & Its Applications*, 26(3): 532–553, 1982.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Bao Wang, Tan Nguyen, Tao Sun, Andrea L Bertozzi, Richard G Baraniuk, and Stanley J Osher. Scheduled restart momentum for accelerated stochastic gradient descent. *SIAM Journal on Imaging Sciences*, 15(2):738–761, 2022.

Jun-Kun Wang, Chi-Heng Lin, and Jacob Abernethy. Escaping saddle points faster with stochastic momentum. *arXiv preprint arXiv:2106.02985*, 2021.

Qingsong Wang, Zehui Liu, Chunfeng Cui, and Deren Han. Inertial accelerated sgd algorithms for solving large-scale lower-rank tensor cp decomposition problems. *Journal of Computational and Applied Mathematics*, 423:114948, 2023.

Stephen J Wright. *Numerical optimization*. 2006.

Zeke Xie, Xinrui Wang, Huishuai Zhang, Issei Sato, and Masashi Sugiyama. Adaptive inertia: Disentangling the effects of adaptive learning rate and momentum, 2022.

Yi Zhou, Zhe Wang, Kaiyi Ji, Yingbin Liang, and Vahid Tarokh. Proximal gradient algorithm with momentum and flexible parameter restart for nonconvex optimization. *arXiv preprint arXiv:2002.11582*, 2020.

A Appendix

A.1 Stability properties of gradient descent with Inertial Momentum proof

We follow similar derivation as in Cohen et al. [2021] appendix. We are interested in optimizing against an quadratic objective function of the form:

$$f(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

Re-arranging our Inertial equations:

$$\mathbf{v}_t = \mathbf{x}_t + \alpha(\mathbf{x}_t - \mathbf{x}_{t-1}) = (\alpha+1)\mathbf{x}_t - \alpha\mathbf{x}_{t-1} \tag{4}$$
$$\mathbf{y}_t = x_t + \beta(\mathbf{x}_t - \mathbf{x}_{t-1}) \tag{5}$$
$$\mathbf{x}_{t+1} = \mathbf{v}_t - \eta\nabla f(\mathbf{y}_t) \tag{6}$$

$$\nabla_{y_t} f(\mathbf{y}_t) = \mathbf{A}((1+\beta)\mathbf{x}_t - \beta\mathbf{x}_{t-1}) + \mathbf{b}$$
$$\nabla_{y_t} f(\mathbf{y}_t) = (1+\beta)\mathbf{A}\mathbf{x}_t - \beta\mathbf{A}\mathbf{x}_{t-1} + \mathbf{b}$$

Now GD with inertial momentum update step -

$$\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{v}_t - \eta\nabla L(\mathbf{y}_t) \\
\mathbf{x}_{t+1} &= (\alpha+1)\mathbf{x}_t - \alpha\mathbf{x}_{t-1} - \eta[(1+\beta)\mathbf{A}\mathbf{x}_t - \beta\mathbf{A}\mathbf{x}_{t-1} + \mathbf{b}] \\
&= [1+\alpha-\eta(1+\beta)\mathbf{A}]\mathbf{x}_t - (\alpha - \eta\beta\mathbf{A})\mathbf{x}_{t-1} + \mathbf{b} \\
&= (1+\alpha)\Big[\mathbf{I} - \frac{\eta(1+\beta)}{1+\alpha}\mathbf{A}\Big]\mathbf{x}_t - \alpha\Big[\mathbf{I} - \frac{\eta\beta\mathbf{A}}{\alpha}\Big]\mathbf{x}_{t-1} - \eta\mathbf{b}
\end{aligned}$$

The quantity $\mathbf{q}^T\mathbf{x}_t$ evolves under gradient descent as:- (also note $(\mathbf{q}^T\mathbf{A} = a\mathbf{q})$)

$$\mathbf{q}^T\mathbf{x}_{t+1} = (1+\alpha)\Big[1 - \frac{\eta(1+\beta)}{1+\alpha}a\Big]\mathbf{q}^T\mathbf{x}_t - \alpha\Big[1 - \frac{\eta\beta a}{\alpha}\Big]\mathbf{q}^T\mathbf{x}_{t-1} - \eta\mathbf{q}^T\mathbf{b}$$

Defining $\tilde{x}_t = q^T x_t + \frac{q^T b}{a}$, similar to Cohen et al. [2021] and note that $q^T x_t$ diverges iff $\tilde{x}_t$ diverges. Thus, rearranging above equation.

$$\tilde{\mathbf{x}}_{t+1} = (1+\alpha)\Big[1 - \frac{\eta(1+\beta)}{1+\alpha}a\Big]\mathbf{q}^T\tilde{\mathbf{x}}_t - \alpha\Big[1 - \frac{\eta\beta a}{\alpha}\Big]\mathbf{q}^T\tilde{\mathbf{x}}_{t-1}$$

Above equation, is a linear homogenous second-order difference equation. By Theorem 2.37 in Elaydi [2005] (replacing coefficients from above equations in Theorem 2.37). We get If

$$a > \frac{1}{\eta}\Big(\frac{2+2\alpha}{1+2\beta}\Big)$$

then this recurrence diverges. Hence, using the above results we can get appropriate adjusted Edge of Stability (EOS) bound for inertial momentum.

A.2 Further Toy Experiments

Figure (3) illustrates the behavior of our method (ADAM+MoXCo) when initialized in unfavorable energy regions. The red circle indicates the starting point and the plot demonstrates that Adaptive Inertia adeptly explores and identifies shallow regions for convergence. We used Goodness score of 0.85, $\eta = 0.001, \alpha, \beta = 0.99, 0.95$ and restarting $\alpha, \beta = 0.9, 0.8$

Additional note about Figure 1 (a) - it compares optimization trajectory between GD + MoXCo with GD & Adam and (b) compares between Adam + MoXCo with Adam & GD. Both figures show trajectories with same initialization (black circle) and all optimizers run for total of 800 steps. We see that in both plots only adaptive inertia variants are able to explore the loss surfer and converge to global minima whereas other optimizers get stuck in unwanted local stationary points. We used Goodness score of 0.85, temperature of 0.15 and restarting values of $\alpha, \beta = 0.1, 0.1$ for both figures.

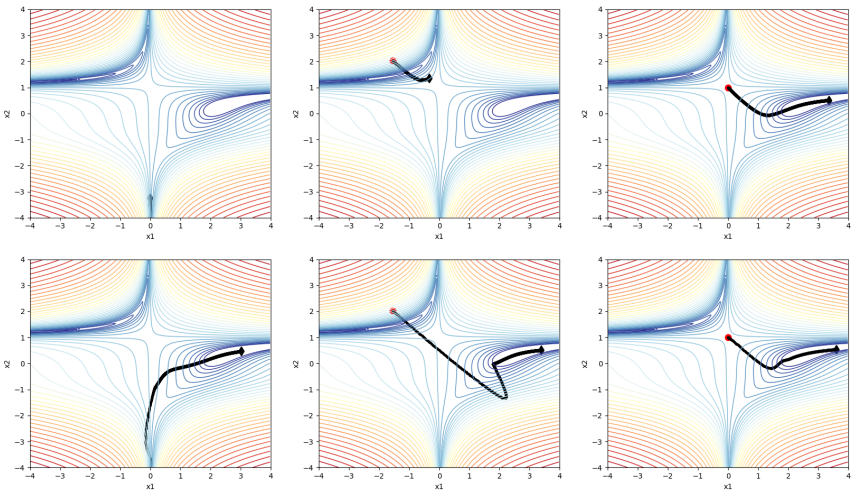


Figure 3: The top row displays results obtained with ADAM for three different initializations. Meanwhile, the bottom row showcases outcomes for ADAM + Inertial (which contains both inertial momentum & parameter restart with goodness score) with the same initializations as the top row. ADAM tends to become trapped in local stationary points when initiated in unfavorable energy zones. In contrast, the adaptive inertia can effectively navigate away from sharper regions, ultimately converging towards a flatter optimum.

A.3 Extended work & Discussion

In this paper, we present a framework for enhancing exploration and a commit-to-converge strategy (5.2). While this work is still in progress, below we outline the extended research scope that has emerged from this study.

Goodness score every iteration In section 5, we compute our goodness score per epoch. An alternative approach would be to perform this calculation every iteration. This would allow the heuristic function become more responsive and give us ability to estimate λ_{max} frequently. However, this is not feasible with our current method of largest eigenvalue estimation. A tractable alternative could be to implement the PowerSGD (Vogels et al. [2019]) method for efficiently estimating the low-rank approximation of the gradient via generalized power iteration at every iteration. Since the gradients accessible to us are noisy stochastic estimates, the subsequent matrix-vector products are inaccurate. Hence, we should use a generalized power iteration method that is robust in the presence of noise.

Alternative to SAM Foret et al. [2020] Given that one of our key metrics is the λ_{max} of the validation data, we monitor it on an epoch-by-epoch basis. We are currently exploring an alternative approach to sharpness-aware training by incorporating regularization based on the eigenvector associated with the largest eigenvalue of to our quantized objective. By doing this we would be penalizing any sharp curvature changes thereby, promoting flatter minima.