A Generic Approach for Escaping Saddle points

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

A central challenge to using first-order methods for optimizing nonconvex problems is the presence of saddle points. First-order methods often get stuck at saddle points, greatly deteriorating their performance. Typically, to escape from saddles one has to use second-order methods. However, most works on second-order methods rely extensively on expensive Hessian-based computations, making them impractical in large-scale settings. To tackle this challenge, we introduce a generic framework that minimizes Hessian based computations while at the same time provably converging to second-order critical points. Our framework carefully alternates between a first-order and a second-order subroutine, using the latter only close to saddle points, and yields convergence results competitive to the state-of-the-art. Empirical results suggest that our strategy also enjoys good practical performance.

1 Introduction

We study nonconvex *finite-sum* problems of the form

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ nor the individual functions $f_i : \mathbb{R}^d \to \mathbb{R}$ ($i \in [n]$) are necessarily convex. Optimization problems of this form arise naturally in machine learning e.g. empirical risk minimization.

In the large-scale settings, algorithms based on first-order information of functions f_i are typically favored as they are relatively inexpensive and scale seamlessly. An algorithm widely used in practice is stochastic gradient descent (SGD) which under suitable selection of the learning rate converges to a point x that, in expectation, satisfies the stationarity condition $\|\nabla f(x)\| \leq \epsilon$ in $O(1/\epsilon^4)$ iterations [9]. This result neither ensures convergence to second-order critical points nor the rate is fast. For general nonconvex problems, one has to settle for a more modest goal than sub-optimality, as finding the global minimizer is intractably hard. Unfortunately, SGD does not even ensure second-order critical conditions as it can get stuck at saddle points.

To overcome these issues, the cubic regularization (CR) method [23] explicitly uses Hessians to obtain faster convergence rates. In particular, Nesterov and Polyak [23] showed that CR requires $O(1/\epsilon^{3/2})$ iterations to achieve the second-order critical conditions. However, each iteration of CR is expensive as it requires computing the Hessian and solving multiple linear systems, each of which has complexity $O(d^\omega)$ (ω is the matrix multiplication constant), thus, undermining the benefit of its faster convergence. Recently, Agarwal et al. [2] designed an algorithm to solve the CR more efficiently, however, it still exhibits slower convergence in practice compared to first-order methods. Both of these approaches use Hessian based optimization in each iteration, which make them slow in practice.

A second line of work focuses on using Hessian information (or its structure) whenever the method gets stuck at stationary points that are not second-order critical. To our knowledge, the first work in this line is [8], which shows that for a class of functions that satisfy a special property called "strict-saddle" property, a noisy variant of SGD can converge to a point close to a local minimum. For this class of functions, points close to saddle points have a Hessian with a large negative eigenvalue, which proves instrumental in escaping saddle points using an isotropic noise. While such a noise-based method is appealing as it only uses first-order information, it has a very bad dependence on the dimension d, and furthermore, the result only holds when the strict-saddle property is satisfied [8].

Inspired by this line of work, we develop a general framework for finding second-order critical points. The key idea is to use first-order information for the most part of the optimization process and invoke Hessian information only when stuck at stationary points that are not second-order critical.

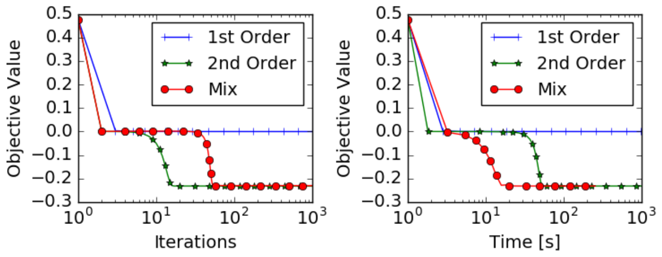

Figure 1: First order methods like GD can potentially get stuck at saddle points. Second-order methods can escape it in very few iterations (as observed in the left plot) but at the cost of expensive Hessian based iterations (see time plot to the right). The proposed framework, which is a novel mix of the two strategies, can escape saddle points *faster* in time by carefully trading off computation and iteration complexity.

2 Background & Problem Setup

We assume that each of the functions f_i in (1) is L-smooth, i.e., $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x-y\|$ for all $i \in [n]$. Furthermore, we assume that the Hessian of f in (1) is M-Lipschitz, i.e., $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq M\|x-y\|$ for all $x, y \in \mathbb{R}^d$. We also assume that the function f is bounded below, i.e., $f(x) \geq B$ for all $x \in \mathbb{R}^d$.

In order to measure stationarity of an iterate x, similar to [9, 22, 23], we use the condition $\|\nabla f(x)\| \leq \epsilon$. In this paper, we are interested in convergence to second-order critical points. Thus, in addition to stationarity, we also require the solution to satisfy the Hessian condition $\nabla^2 f(x) \succeq -\gamma \mathbb{I}$ [23].

Definition 1. *An algorithm $\mathcal{A}$ is said to obtain a point x that is a (ϵ, γ)-second order critical point if $\mathbb{E}[\|\nabla f(x)\|] \leq \epsilon$ and $\mathbb{E}[\nabla^2 f(x)] \succeq -\gamma \mathbb{I}$, where the expectation is over any randomness in $\mathcal{A}$.*

For our algorithms, we use only cheap Incremental First-order Oracle (IFO) [1] and an Incremental Second-order Oracle (ISO), which have time complexity $O(d)$ in many practical settings.

Definition 2. *An IFO takes an index $i \in [n]$ and a point $x \in \mathbb{R}^d$, and returns the pair $(f_i(x), \nabla f_i(x))$. An ISO takes an index $i \in [n]$, point $x \in \mathbb{R}^d$ and vector $v \in \mathbb{R}^d$ and returns the vector $\nabla^2 f_i(x) v$.*

For clarity and clean comparison, the dependence of time complexity on Lipschitz constant L, M, initial point and any polylog factors in the results is hidden.

3 Generic Framework

We propose a generic framework for escaping saddle points while solving nonconvex problems of form (1). To evade saddle points, one needs to use properties of both gradients and Hessians. To this end, our framework is based on two core subroutines: GRADIENT-FOCUSED-OPTIMIZER and HESSIAN-FOCUSED-OPTIMIZER.

The idea is to use these two subroutines, each focused on different aspects of the optimization procedure. GRADIENT-FOCUSED-OPTIMIZER focuses on using gradient information for decreasing the function. On its own, the GRADIENT-FOCUSED-OPTIMIZER might not converge to a second order critical point since it can get stuck at a saddle point. Hence, we require the subroutine HESSIAN-FOCUSED-OPTIMIZER to help avoid saddle points. We design a procedure that interleave these subroutines to obtain a second-order critical point, which not only provides meaningful theoretical guarantees, but also translates into strong empirical gains.

Algorithm 1 provides pseudocode of our framework. Observe that the algorithm is still abstract, and we assume the following properties to hold for these subroutines.

- GRADIENT-FOCUSED-OPTIMIZER: Suppose $(y, z) = \text{GRADIENT-FOCUSED-OPTIMIZER}(x, n, \epsilon)$, then there exists positive function $g : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{R}^+$, such that

  G.1 $\mathbb{E}[f(y)] \leq f(x)$,

  G.2 $\mathbb{E}[\|\nabla f(y)\|^2] \leq \frac{1}{g(n,\epsilon)} \mathbb{E}[f(x) - f(z)]$.

  Here the outputs $y, z \in \mathbb{R}^d$. The expectation in the conditions above is over any randomness that is a part of the subroutine. The function g will be critical for the overall rate of Algorithm 1. Typically, GRADIENT-FOCUSED-OPTIMIZER is a first-order method, since the primary aim of this subroutine is to focus on gradient based optimization.

- HESSIAN-FOCUSED-OPTIMIZER: Suppose $(y, \tau) = \text{HESSIAN-FOCUSED-OPTIMIZER}(x, n, \epsilon, \gamma)$ where $y \in \mathbb{R}^d$ and $\tau \in \{\varnothing, \diamond\}$. If $\tau = \varnothing$, then y is a (ϵ, γ)-second order critical point with probability at least $1 - q$. Otherwise if $\tau = \diamond$, then y satisfies the following condition:

  H.1 $\mathbb{E}[f(y)] \leq f(x)$,

  H.2 $\mathbb{E}[f(y)] \leq f(x) - h(n, \epsilon, \gamma)$ when $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ for some $h : \mathbb{N} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$.

  Here the expectation is over any randomness in subroutine. The two conditions ensure that the objective function value, in expectation, never increases and decreases with a certain rate when $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$. In general, this subroutine utilizes the Hessian or its properties for minimizing the objective function. Typically, this is the most expensive part of the Algorithm 1 and hence, needs to be invoked judiciously.

Algorithm 1 Generic Framework

```
1:  Input - Initial point: $x^0$, total iterations $T$, error threshold parameters $\epsilon$, $\gamma$ and probability $p$
2:  for $t = 1$ to $T$ do
3:      $(y^t, z^t) = $ GRADIENT-FOCUSED-OPTIMIZER$(x^{t-1}, \epsilon)$ (refer to G.1 and G.2)
4:      Choose $u^t$ as $y^t$ with probability $p$ and $z^t$ with probability $1 - p$
5:      $(x^{t+1}, \tau^{t+1}) = $ HESSIAN-FOCUSED-OPTIMIZER$(u^t, \epsilon, \gamma)$ (refer to H.1 and H.2)
6:      if $\tau^{t+1} = \varnothing$ then
7:          Output set $\{x^{t+1}\}$
8:      end if
9:  end for
10: Output set $\{y^1, ..., y^T\}$
```

3.1 Convergence Analysis

The key aspect of these subroutines is that they, in expectation, never increase the objective function value. The functions g and h will determine the convergence rate of Algorithm 1.

Theorem 1. *Let $\Delta = f(x^0) - B$ and $\theta = \min((1-p)\epsilon^2 g(n,\epsilon), ph(n,\epsilon,\gamma))$. Also, let set Γ be the output of Algorithm 1 with* GRADIENT-FOCUSED-OPTIMIZER *satisfying **G.1** and **G.2** and* HESSIAN-FOCUSED-OPTIMIZER *satisfying **H.1** and **H.2**. Furthermore, T be such that $T > \Delta/\theta$. Suppose the multiset $S = \{i_1, ...i_k\}$ are k indices selected independently and uniformly randomly from {1, ..., $|\Gamma|$}. Then the following holds for the indices in S:*

1. *y^t, where $t \in \{i_1, ..., i_k\}$, is a (ϵ, γ)-critical point with probability at least $1 - \max(\Delta/(T\theta), q)$.*
2. *If $k = O(\log(1/\zeta)/\min(\log(\Delta/(T\theta)), \log(1/q)))$, with at least probability $1 - \zeta$, at least one iterate y^t where $t \in \{i_1, ..., i_k\}$ is a (ϵ, γ)-critical point.*

The proof of the result is presented in Appendix B. The key point regarding the above result is that the overall convergence rate depends on the magnitude of both functions g and h. Theorem 1 shows that the slowest amongst the subroutines GRADIENT-FOCUSED-OPTIMIZER and HESSIAN-FOCUSED-OPTIMIZER governs the overall rate of Algorithm 1. Thus, it is important to ensure that both these procedures have good convergence. Also, note that the optimal setting for p based on the result above satisfies $1/p = 1/\epsilon^2 g(n,\epsilon) + 1/h(n,\epsilon,\gamma)$.

3.2 An Example Instantiation

We now present a specific instantiation of our framework and derive the time complexity required to reach a second order critical point. For this example we use SVRG as the GRADIENT-FOCUSED-OPTIMIZER and HESSIANDESCENT as the HESSIAN-FOCUSED-OPTIMIZER.

- SVRG [12, 26] is a stochastic algorithm recently shown to be very effective for reducing variance in finite-sum problems. Strong convergence rates have been proved for SVRG in the context of convex and nonconvex optimization [12, 26]. The following result shows that SVRG meets the requirements of a GRADIENT-FOCUSED-OPTIMIZER.

  Lemma 1. SVRG *with $\eta_t = \eta = 1/4Ln^{2/3}$, $m = n$ and $T_g = T_\epsilon$, which depends on ϵ, is a* GRADIENT-FOCUSED-OPTIMIZER *with $g(n,\epsilon) = T_\epsilon/40Ln^{2/3}$.*

  The algorithm for SVRG and the proof of the result is presented in Appendix C.

- HESSIANDESCENT is a direct approach using the eigenvector corresponding to the smallest eigenvalue of the hessian to make a descent step. More specifically, when the smallest eigenvalue of the Hessian is negative and reasonably large in magnitude, i.e. $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ then the Hessian information can be used to ensure descent in the objective function value. Note the subroutine is designed in a fashion such that the objective function value never increases. The following result shows that HESSIANDESCENT meets the requirements of a HESSIAN-FOCUSED-OPTIMIZER.

  Lemma 2. HESSIANDESCENT *is a* HESSIAN-FOCUSED-OPTIMIZER *with $h(n,\epsilon,\gamma) = \frac{\rho}{24M^2}\gamma^3$.*

  The algorithm for HESSIANDESCENT and proof of the result is presented in Appendix D.

Now we can show the following key result:

Theorem 2. *Suppose* SVRG *with $m = n$, $\eta_t = \eta = 1/4Ln^{2/3}$ for all $t \in \{1, ..., m\}$ and $T_g = 40Ln^{2/3}/\epsilon^{1/2}$ is used as* GRADIENT-FOCUSED-OPTIMIZER *and* HESSIANDESCENT *is used as* HESSIAN-FOCUSED-OPTIMIZER *with $q = 0$, then Algorithm 1 finds a $(\epsilon, \sqrt{\epsilon})$-second order critical point in $T = O(\Delta/\min(p, 1-p)\epsilon^{3/2})$ with probability at least 0.9.*

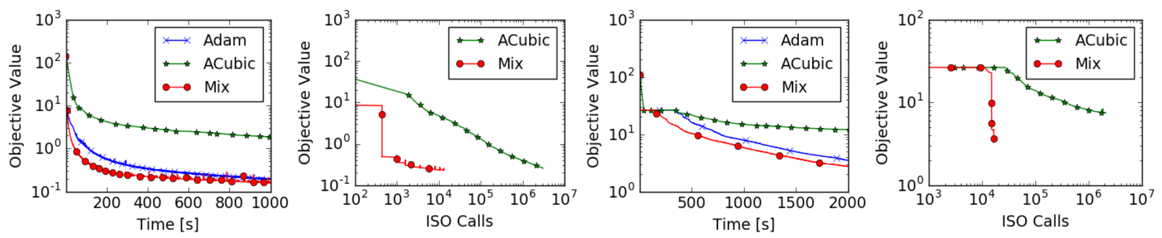

Figure 2: Comparison of various methods on CURVES and MNIST Deep Autoencoder. Our mix approach converges faster than the baseline methods and uses relatively few ISO calls, which are practically relatively expensive to IFO calls, in comparison to APPROXCUBICDESCENT.

The result directly follows from using Lemma 1 and 2 in Theorem 1. Combining this with the time complexity of SVRG which is $O(nd + T_g d) = O(nd + n^{2/3}d/\epsilon^{1/2})$ and HESSIANDESCENT which is $(O(nd + n^{3/4}d/\epsilon^{1/4}))$, we get the following result.

Corollary 1. *The overall running time of Algorithm 1 to find a* $(\epsilon, \sqrt{\epsilon})$*-second order critical point, with parameter settings used in Theorem 2, is* $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4} + n^{2/3}d/\epsilon^2)$.

Note that the dependence on ϵ is much better in comparison to that of Noisy SGD used in [8]. Furthermore, our results are competitive with [2, 4] in their respective settings, but with a much simpler algorithm and analysis. We also note that our algorithm is faster than [11], which has a time complexity of $O(nd/\epsilon^2)$.

3.3 Practical Considerations

To further achieve good empirical performance, we had to slightly modify these procedures. For HESSIAN-FOCUSED-OPTIMIZER, we found stochastic, adaptive and inexact approaches for solving HESSIANDESCENT and CUBICDESCENT work well in practice. Due to lack of space, the exact description of these modifications is deferred to Appendix F. Furthermore, in the context of deep learning, empirical evidence suggests that first-order methods like ADAM [13] exhibit behavior that is in congruence with properties **G.1** and **G.2**. While theoretical analysis for a setting where ADAM is used as GRADIENT-FOCUSED-OPTIMIZER is still unresolved, we nevertheless demonstrate its performance through empirical results.

4 Experiments

To investigate the practical performance of the framework, we applied it to two deep autoencoder optimization problems from [10] called “CURVES” and “MNIST”. Due to their high difficulty, performance on these problems has become a standard benchmark for neural network optimization methods, e.g. [20, 21, 32, 33]. The “CURVES” autoencoder consists of an encoder with layers of size (28x28)-400-200-100- 50-25-6 and a symmetric decoder totaling in 0.85M parameters. The six units in the code layer were linear and all the other units were logistic. The network was trained on 20,000 images and tested on 10,000 new images. The data set contains images of curves that were generated from three randomly chosen points in two dimensions. The “MNIST” autoencoder consists of an encoder with layers of size (28x28)-1000-500-250-30 and a symmetric decoder, totaling in 2.8M parameters. The thirty units in the code layer were linear and all the other units were logistic. The network was trained on 60,000 images and tested on 10,000 new images. The data set contains images of handwritten digits 0-9. The pixel intensities were normalized to lie between 0 and 1.[1]

As an instantiation of our framework, we use a mix of ADAM, which is popular in deep learning community, and an APPROXCUBICDESCENT for the practical reasons mentioned in Section 3.3. This method with ADAM and APPROXCUBICDESCENT. The parameters of these algorithms were chosen to produce the best generalization on a held out test set. The regularization parameter M was chosen as the smallest value such that the function value does not fluctuate in the first 10 epochs. We use the initialization suggested in [20] and a mini-batch size of 1000 for all the algorithms. We report objective function value against wall clock time and ISO calls.

The results are presented in Figure 2, which shows that our proposed mix framework was the *fastest* to escape the saddle point in terms of wall clock time. ADAM took considerably more time to escape the saddle point, especially in the case of MNIST. While APPROXCUBICDESCENT escaped the saddle point in relatively fewer iterations, each iteration required considerably large number of ISO calls; as a result, the method was extremely slow in terms of wall clock time, despite our efforts to improve it via approximations and code optimizations. On the other hand, our proposed framework, seamlessly balances these two methods, thereby, resulting in the fast decrease of training loss.

[1]Data available at: `www.cs.toronto.edu/~jmartens/digs3pts_1.mat`, `mnist_all.mat`

References

[1] Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. *arXiv:1410.0723*, 2014.

[2] Naman Agarwal, Zeyuan Allen Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima for nonconvex optimization in linear time. *CoRR*, abs/1611.01146, 2016.

[3] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8), 1991.

[4] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for non-convex optimization. *CoRR*, abs/1611.00756, 2016.

[5] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, pages 1–39, 2017. ISSN 1436-4646. doi: 10.1007/s10107-017-1137-4.

[6] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS 27*, pages 1646–1654, 2014.

[7] Aaron J Defazio, Tibério S Caetano, and Justin Domke. Finito: A faster, permutable incremental gradient method for big data problems. *arXiv:1407.2710*, 2014.

[8] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points - online stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015*, pages 797–842, 2015.

[9] Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. doi: 10.1137/120880811.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. *CoRR*, abs/1703.00887, 2017.

[12] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS 26*, pages 315–323, 2013.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] Jakub Konečný, Jie Liu, Peter Richtárik, and Martin Takáč. Mini-Batch Semi-Stochastic Gradient Descent in the Proximal Setting. *arXiv:1504.04407*, 2015.

[15] Harold Joseph Kushner and Dean S Clark. *Stochastic approximation methods for constrained and unconstrained systems*, volume 26. Springer Science & Business Media, 2012.

[16] Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *arXiv:1507.02000*, 2015.

[17] Kfir Y. Levy. The power of normalization: Faster evasion of saddle points. *CoRR*, abs/1611.04831, 2016.

[18] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization. In *NIPS*, 2015.

[19] Lennart Ljung. Analysis of recursive stochastic algorithms. *Automatic Control, IEEE Transactions on*, 22(4):551–575, 1977.

[20] James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

[21] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417, 2015.

[22] Yurii Nesterov. *Introductory Lectures On Convex Optimization: A Basic Course*. Springer, 2003.

[23] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[24] BT Poljak and Ya Z Tsypkin. Pseudogradient adaptation and training algorithms. *Automation and Remote Control*, 34:45–67, 1973.

[25] Sashank Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *NIPS 28*, pages 2629–2637, 2015.

[26] Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 314–323, 2016.

[27] Sashank J. Reddi, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola. Fast incremental method for nonconvex optimization. *CoRR*, abs/1603.06159, 2016.

[28] Sashank J. Reddi, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola. Fast stochastic methods for nonsmooth nonconvex optimization. *CoRR*, 2016.

[29] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22: 400–407, 1951.

[30] Mark W. Schmidt, Nicolas Le Roux, and Francis R. Bach. Minimizing Finite Sums with the Stochastic Average Gradient. *arXiv:1309.2388*, 2013.

[31] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[32] 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, 2013.

[33] Oriol Vinyals and Daniel Povey. Krylov subspace descent for deep learning. In *AISTATS*, pages 1261–1268, 2012.

Appendix: A Generic Approach for Escaping Saddle points

A Related Works

There is a vast literature on algorithms for solving optimization problems of the form (1). A classical approach for solving such optimization problems is SGD, which dates back at least to the seminal work of [29]. Since then, SGD has been a subject of extensive research, especially in the convex setting [3, 15, 19, 24]. Recently, new faster methods, called variance reduced (VR) methods, have been proposed for convex finite-sum problems. VR methods attain faster convergence by reducing the variance in the stochastic updates of SGD, see e.g., [6, 7, 12, 14, 30, 31]. Accelerated variants of these methods achieve the lower bounds proved in [1, 16], thereby settling the question of their optimality. Furthermore, [25] developed an asynchronous framework for VR methods and demonstrated their benefits in parallel environments.

Most of the aforementioned prior works study stochastic methods in convex or very specialized nonconvex settings that admit theoretical guarantees on sub-optimality. For the general nonconvex setting, it is only recently that non-asymptotic convergence rate analysis for SGD and its variants was obtained in [9], who showed that SGD ensures $\|\nabla f\| \leq \epsilon$ (in expectation) in $O(1/\epsilon^4)$ iterations. A similar rate for parallel and distributed SGD was shown in [18]. For these problems, Reddi et al. [26, 27, 28] proved faster convergence rates that ensure the same optimality criteria in $O(n + n^{2/3}/\epsilon^2)$, which is an order $n^{1/3}$ faster than GD. While these methods ensure convergence to *stationary* points at a faster rate, the question of convergence to local minima (or in general to second-order critical points) has not been addressed. To our knowledge, convergence rates to second-order critical points (defined in Definition 1) for general nonconvex functions was first studied by [23]. However, each iteration of the algorithm in [23] is prohibitively expensive since it requires eigenvalue decompositions, and hence, is unsuitable for large-scale high-dimensional problems. More recently, Agarwal et al. [2], Carmon et al. [4] presented algorithms for finding second-order critical points by tackling some practical issues that arise in [23]. However, these algorithms are either only applicable to a restricted setting or heavily use Hessian based computations, making them unappealing from a practical standpoint. *Noisy* variants of first-order methods have also been shown to escape saddle points (see [8, 11, 17]), however, these methods have strong dependence on either n or d, both of which are undesirable.

B Proof of Theorem 1

The case of $\tau = \varnothing$ can be handled in a straightforward manner, so let us focus on the case where $\tau = \diamond$. We split our analysis into cases, each analyzing the change in objective function value depending on second-order criticality of y^t.

We start with the case where the gradient condition of second-order critical point is violated and then proceed to the case where the Hessian condition is violated.

Case I: $\mathbb{E}[\|\nabla f(y^t)\|] \geq \epsilon$ for some $t > 0$

We first observe the following: $\mathbb{E}[\|\nabla f(y^t)\|^2] \geq (\mathbb{E}\|\nabla f(y^t)\|)^2 \geq \epsilon^2$. This follows from a straightforward application of Jensen's inequality. From this inequality, we have the following:

$$\epsilon^2 \leq \mathbb{E}[\|\nabla f(y^t)\|^2] \leq \frac{1}{g(n,\epsilon)}\mathbb{E}[f(x^{t-1}) - f(z^t)]. \tag{2}$$

This follows from the fact that y^t is the output of GRADIENT-FOCUSED-OPTIMIZER subroutine, which satisfies the condition that for (y, z) = GRADIENT-FOCUSED-OPTIMIZER(x, n, ϵ), we have

$$\mathbb{E}[\|\nabla f(y)\|^2] \leq \frac{1}{g(n,\epsilon)}\mathbb{E}[f(x) - f(z)].$$

From Equation (2), we have

$$\mathbb{E}[f(z^t)] \leq \mathbb{E}[f(x^{t-1})] - \epsilon^2 g(n,\epsilon).$$

Furthermore, due to the property of non-increasing nature of GRADIENT-FOCUSED-OPTIMIZER, we also have $\mathbb{E}[y^t] \leq \mathbb{E}[f(x^{t-1})]$.

We now focus on the HESSIAN-FOCUSED-OPTIMIZER subroutine. From the property of HESSIAN-FOCUSED-OPTIMIZER that the objective function value is non-increasing, we have

$\mathbb{E}[f(x^t)] \leq \mathbb{E}[f(u^t)]$. Therefore, combining with the above inequality, we have

$$\begin{aligned}\mathbb{E}[f(x^t)] &\leq \mathbb{E}[f(u^t)] \\ &= p\mathbb{E}[f(y^t)] + (1-p)\mathbb{E}[f(z^t)] \\ &\leq p\mathbb{E}[f(x^{t-1})] + (1-p)(\mathbb{E}[f(x^{t-1})] - \epsilon^2 g(n,\epsilon)) \\ &= \mathbb{E}[f(x^{t-1})] - (1-p)\epsilon^2 g(n,\epsilon). \end{aligned} \tag{3}$$

The first equality is due to the definition of u^t in Algorithm 1. Therefore, when the gradient condition is violated, irrespective of whether $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ or $\nabla^2 f(y^t) \succeq -\gamma\mathbb{I}$, the objective function value always decreases by at least $\epsilon^2 g(n,\epsilon)$.

Case II: $\mathbb{E}[\|\nabla f(y^t)\|] < \epsilon$ and $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ for some $t > 0$

In this case, we first note that for $y = \text{HESSIAN-FOCUSED-OPTIMIZER}(x, n, \epsilon, \gamma)$ and $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$, we have $\mathbb{E}[f(y)] \leq f(x) - h(n,\epsilon,\gamma)$. Observe that $x^t = \text{HESSIAN-FOCUSED-OPTIMIZER}(u^t, n, \epsilon, \gamma)$. Therefore, if $u^t = y^t$ and $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$, then we have

$$\mathbb{E}[f(x^t)|u^t = y^t] \leq f(y^t) - h(n,\epsilon,\gamma) \leq f(x^{t-1}) - h(n,\epsilon,\gamma).$$

The second inequality is due to the non-increasing property of GRADIENT-FOCUSED-OPTIMIZER. On the other hand, if $u^t = z^t$, we have hand, if we have $\mathbb{E}[f(x^t)|u^t = z^t] \leq f(z^t)$. This is due to the non-increasing property of HESSIAN-FOCUSED-OPTIMIZER. Combining the above two inequalities and using the law of total expectation, we get

$$\begin{aligned}\mathbb{E}[f(x^t)] &= p\mathbb{E}[f(x^t)|u^t = y^t] + (1-p)\mathbb{E}[f(x^t)|u^t = z^t] \\ &\leq p\left(\mathbb{E}[f(y^t)] - h(n,\epsilon,\gamma)\right) + (1-p)\mathbb{E}[f(z^t)] \\ &\leq p\left(\mathbb{E}[f(x^{t-1})] - h(n,\epsilon,\gamma)\right) + (1-p)\mathbb{E}[f(x^{t-1})] \\ &= \mathbb{E}[f(x^{t-1})] - ph(n,\epsilon,\gamma). \end{aligned} \tag{4}$$

The second inequality is due to he non-increasing property of GRADIENT-FOCUSED-OPTIMIZER. Therefore, when the hessian condition is violated, the objective function value always decreases by at least $ph(n,\epsilon,\gamma)$.

Case III: $\mathbb{E}[\|\nabla f(y^t)\|] < \epsilon$ and $\nabla^2 f(y^t) \succeq -\gamma\mathbb{I}$ for some $t > 0$

This is the favorable case for the algorithm. The only condition to note is that the objective function value will be non-increasing in this case too. This is, again, due to the non-increasing properties of subroutines GRADIENT-FOCUSED-OPTIMIZER and HESSIAN-FOCUSED-OPTIMIZER. In general, greater the occurrence of this case during the course of the algorithm, higher will the probability that the output of our algorithm satisfies the desired property.

The key observation is that Case I & II cannot occur large number of times since each of these cases strictly decreases the objective function value. In particular, from Equation (3) and (4), it is easy to see that each occurrence of Case I & II the following holds:

$$\mathbb{E}[f(x^t)] \leq \mathbb{E}[f(x^{t-1})] - \theta,$$

where $\theta = \min((1-p)\epsilon^2 g(n,\epsilon), ph(n,\epsilon,\gamma))$. Furthermore, the function f is lower bounded by B, thus, Case I & II cannot occur more than $(f(x^0) - B)/\theta$ times. Therefore, the probability of occurrence of Case III is at least $1 - (f(x^0) - B)/(T\theta)$, which completes the first part of the proof.

The second part of the proof simply follows from first part. As seen above, the probability of Case I & II is at most $(f(x^0) - B)/T\theta$. Therefore, probability that an element of the set S falls in Case III is at least $1 - ((f(x^0) - B)/T\theta)^k$, which gives us the required result for the second part.

C SVRG and Proof of Lemma 1

SVRG [12, 26] is a stochastic algorithm recently shown to be very effective for reducing variance in finite-sum problems. We seek to understand its benefits for nonconvex optimization, with a particular focus on the issue of escaping saddle points. Algorithm 2 presents SVRG's pseudocode.

Observe that Algorithm 2 is an epoch-based algorithm. At the start of each epoch s, a full gradient is calculated at the point $\tilde{x}^s$, requiring n calls to the IFO. Within its inner loop SVRG performs m stochastic updates. Suppose m is chosen to be $O(n)$ (typically used in practice), then the total IFO calls per epoch is $\Theta(n)$. Strong convergence rates have been proved Algorithm 2 in the context of convex and nonconvex optimization [12, 26]

Algorithm 2 SVRG(x^0, ϵ)

```
1: Input: $x_m^0 = x^0 \in \mathbb{R}^d$, epoch length $m$, step sizes $\{\eta_i > 0\}_{i=0}^{m-1}$, iterations $T_g$, $S = \lceil T_g/m \rceil$
2: for $s = 0$ to $S - 1$ do
3:     $\tilde{x}^s = x_0^{s+1} = x_m^s$
4:     $g^{s+1} = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x}^s)$
5:     for $t = 0$ to $m - 1$ do
6:         Uniformly randomly pick $i_t$ from $\{1, \dots, n\}$
7:         $v_t^{s+1} = \nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s) + g^{s+1}$
8:         $x_{t+1}^{s+1} = x_t^{s+1} - \eta_t v_t^{s+1}$
9:     end for
10: end for
11: Output: $(y, z)$ where $y$ is Iterate $x_a$ chosen uniformly random from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$ and $z = x_m^S$.
```

Proof of Lemma 1. The proof follows from the analysis in [26] with some additional reasoning. We need to show two properties: **G.1** and **G.2**, both of which are based on objective function value. To this end, we start with an update in the s^{th} epoch. We have the following:

$$\begin{aligned}\mathbb{E}[f(x_{t+1}^{s+1})] &\leq \mathbb{E}[f(x_t^{s+1}) + \langle \nabla f(x_t^{s+1}), x_{t+1}^{s+1} - x_t^{s+1}\rangle + \tfrac{L}{2}\|x_{t+1}^{s+1} - x_t^{s+1}\|^2] \\ &\leq \mathbb{E}[f(x_t^{s+1}) - \eta_t\|\nabla f(x_t^{s+1})\|^2 + \tfrac{L\eta_t^2}{2}\|v_t^{s+1}\|^2]. \end{aligned} \tag{5}$$

The first inequality is due to L-smoothness of the function f . The second inequality simply follows from the unbiasedness of SVRG update in Algorithm 2. For the analysis of the algorithm, we need the following Lyapunov function:

$$A_t^{s+1} := \mathbb{E}[f(x_t^{s+1}) + \mu_t\|x_t^{s+1} - \tilde{x}^s\|^2].$$

This function is a combination of objective function and the distance of the current iterate from the latest snapshot $\tilde{x}_s$. Note that the term μ_t is introduced only for the analysis and is not part of the algorithm (see Algorithm 2). Here $\{\mu_t\}_{t=0}^m$ is chosen such the following holds:

$$\mu_t = \mu_{t+1}(1 + \eta_t\beta_t + 2\eta_t^2 L^2) + \eta_t^2 L^3,$$

for all $t \in \{0, \cdots, m-1\}$ and $\mu_m = 0$. For bounding the Lypunov function A, we need the following bound on the distance of the current iterate from the latest snapshot:

$$\begin{aligned}&\mathbb{E}[\|x_{t+1}^{s+1} - \tilde{x}^s\|^2] = \mathbb{E}[\|x_{t+1}^{s+1} - x_t^{s+1} + x_t^{s+1} - \tilde{x}^s\|^2] \\ &= \mathbb{E}[\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 + \|x_t^{s+1} - \tilde{x}^s\|^2 + 2\langle x_{t+1}^{s+1} - x_t^{s+1}, x_t^{s+1} - \tilde{x}^s\rangle] \\ &= \mathbb{E}[\eta_t^2\|v_t^{s+1}\|^2 + \|x_t^{s+1} - \tilde{x}^s\|^2] - 2\eta_t\mathbb{E}[\langle \nabla f(x_t^{s+1}), x_t^{s+1} - \tilde{x}^s\rangle] \\ &\leq \mathbb{E}[\eta_t^2\|v_t^{s+1}\|^2 + \|x_t^{s+1} - \tilde{x}^s\|^2] + 2\eta_t\mathbb{E}\left[\tfrac{1}{2\beta_t}\|\nabla f(x_t^{s+1})\|^2 + \tfrac{1}{2}\beta_t\|x_t^{s+1} - \tilde{x}^s\|^2\right]. \end{aligned} \tag{6}$$

The second equality is due to the unbiasedness of the update of SVRG. The last inequality follows from a simple application of Cauchy-Schwarz and Young's inequality. Substituting Equation (5) and Equation (6) into the Lypunov function A_{t+1}^{s+1}, we obtain the following:

$$\begin{aligned}A_{t+1}^{s+1} &\leq \mathbb{E}[f(x_t^{s+1}) - \eta_t\|\nabla f(x_t^{s+1})\|^2 + \tfrac{L\eta_t^2}{2}\|v_t^{s+1}\|^2] \\ &\quad + \mathbb{E}[\mu_{t+1}\eta_t^2\|v_t^{s+1}\|^2 + \mu_{t+1}\|x_t^{s+1} - \tilde{x}^s\|^2] \\ &\quad + 2\mu_{t+1}\eta_t\mathbb{E}\left[\tfrac{1}{2\beta_t}\|\nabla f(x_t^{s+1})\|^2 + \tfrac{1}{2}\beta_t\|x_t^{s+1} - \tilde{x}^s\|^2\right] \\ &\leq \mathbb{E}[f(x_t^{s+1}) - \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t}\right)\|\nabla f(x_t^{s+1})\|^2 \\ &\quad + \left(\tfrac{L\eta_t^2}{2} + \mu_{t+1}\eta_t^2\right)\mathbb{E}[\|v_t^{s+1}\|^2] + (\mu_{t+1} + \mu_{t+1}\eta_t\beta_t)\,\mathbb{E}\left[\|x_t^{s+1} - \tilde{x}^s\|^2\right]. \end{aligned} \tag{7}$$

To further bound this quantity, we use Lemma 3 to bound $\mathbb{E}[\|v_t^{s+1}\|^2]$, so that upon substituting it in Equation (7), we see that

$$\begin{aligned}A_{t+1}^{s+1} &\leq \mathbb{E}[f(x_t^{s+1})] - \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t} - \eta_t^2 L - 2\mu_{t+1}\eta_t^2\right)\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \\ &\quad + \left[\mu_{t+1}\left(1 + \eta_t\beta_t + 2\eta_t^2 L^2\right) + \eta_t^2 L^3\right]\mathbb{E}\left[\|x_t^{s+1} - \tilde{x}^s\|^2\right] \\ &\leq A_t^{s+1} - \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t} - \eta_t^2 L - 2\mu_{t+1}\eta_t^2\right)\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2]. \end{aligned}$$

The second inequality follows from the definition of μ_t and A_t^{s+1}. Since $\eta_t = \eta = 1/(4Ln^{2/3})$ for $j > 0$ and $t \in \{0, \dots, j-1\}$,

$$A_j^{s+1} \leq A_0^{s+1} - \upsilon_n \sum_{t=0}^{j-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2], \tag{8}$$

where

$$\upsilon_n = \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t} - \eta_t^2 L - 2\mu_{t+1}\eta_t^2\right).$$

We will prove that for the given parameter setting $\upsilon_n > 0$ (see the proof below). With $\upsilon_n > 0$, it is easy to see that $A_j^{s+1} \leq A_0^{s+1}$. Furthermore, note that $A_0^{s+1} = \mathbb{E}[f(x_0^{s+1}) + \mu_0\|x_0^{s+1} - \tilde{x}^s\|^2] = \mathbb{E}[f(x_0^{s+1})]$ since $x_0^{s+1} = \tilde{x}^s$ (see Algorithm 2). Also, we have

$$\mathbb{E}[f(x_j^{s+1}) + \mu_j\|x_j^{s+1} - \tilde{x}^s\|^2] \leq \mathbb{E}[f(x_0^{s+1})]$$

and thus, we obtain $\mathbb{E}[f(x_j^{s+1})] \leq \mathbb{E}[f(x_0^{s+1})]$ for all $j \in \{0, ..., m\}$. Furthermore, using simple induction and the fact that $x_0^{s+1} = x_m^s$ for all epoch $s \in \{0, ..., S-1\}$, it easy to see that $\mathbb{E}[f(x_j^{s+1})] \leq f(x^0)$. Therefore, with the definition of y specified in the output of Algorithm 2, we see that the condition **G.1** of GRADIENT-FOCUSED-OPTIMIZER is satisfied for SVRG algorithm.

We now prove that $\upsilon_n > 0$ and also **G.2** of GRADIENT-FOCUSED-OPTIMIZER is satisifed for SVRG algorithm. By using telescoping the sum with $j = m$ in Equation (8), we obtain

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{A_0^{s+1} - A_m^{s+1}}{\upsilon_n}.$$

This inequality in turn implies that

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{\mathbb{E}[f(\tilde{x}^s) - f(\tilde{x}^{s+1})]}{\upsilon_n}, \tag{9}$$

where we used that $A_m^{s+1} = \mathbb{E}[f(x_m^{s+1})] = \mathbb{E}[f(\tilde{x}^{s+1})]$ (since $\mu_m = 0$), and that $A_0^{s+1} = \mathbb{E}[f(\tilde{x}^s)]$ (since $x_0^{s+1} = \tilde{x}^s$). Now sum over all epochs to obtain

$$\frac{1}{T_g}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{\mathbb{E}[f(x^0) - f(x_m^S)]}{T_g \upsilon_n}. \tag{10}$$

Here we used the the fact that $\tilde{x}^0 = x^0$. To obtain a handle on υ_n and complete our analysis, we will require an upper bound on μ_0. We observe that $\mu_0 = \frac{L}{16n^{4/3}} \frac{(1+\theta)^m - 1}{\theta}$ where $\theta = 2\eta^2 L^2 + \eta\beta$. This is obtained using the relation $\mu_t = \mu_{t+1}(1 + \eta\beta + 2\eta^2 L^2) + \eta^2 L^3$ and the fact that $\mu_m = 0$. Using the specified values of β and η we have

$$\theta = 2\eta^2 L^2 + \eta\beta = \frac{1}{8n^{4/3}} + \frac{1}{4n} \leq \frac{3}{4n}.$$

Using the above bound on θ, we get

$$\begin{aligned}\mu_0 &= \frac{L}{16n^{4/3}} \frac{(1+\theta)^m - 1}{\theta} = \frac{L((1+\theta)^m - 1)}{2(1 + 2n^{1/3})} \\ &\leq \frac{L((1 + \frac{3}{4n})^{\lfloor 4n/3 \rfloor} - 1)}{2(1 + 2n^{1/3})} \leq n^{-1/3}(L(e-1)/4),\end{aligned} \tag{11}$$

wherein the second inequality follows upon noting that $(1 + \frac{1}{l})^l$ is increasing for $l > 0$ and $\lim_{l\to\infty}(1 + \frac{1}{l})^l = e$ (here e is the Euler's number). Now we can lower bound υ_n, as

$$\upsilon_n = \min_t\left(\eta - \tfrac{\mu_{t+1}\eta}{\beta} - \eta^2 L - 2\mu_{t+1}\eta^2\right) \geq \left(\eta - \tfrac{\mu_0\eta}{\beta} - \eta^2 L - 2\mu_0\eta^2\right) \geq \frac{1}{40Ln^{2/3}}.$$

The first inequality holds since μ_t decreases with t. The second inequality holds since (a) μ_0/β can be upper bounded by $(e-1)/4$ (follows from Equation (11)), (b) $\eta^2 L \leq \eta/4$ and (c) $2\mu_0\eta^2 \leq (e-1)\eta/8$ (follows from Equation (11)). Substituting the above lower bound in Equation (10), we obtain the following:

$$\frac{1}{T_g}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leq \frac{40Ln^{2/3}\mathbb{E}[f(x^0) - f(x_m^S)]}{T_g}. \tag{12}$$

From the definition of (y, z) in output of Algorithm 2 i.e., y is Iterate x_a chosen uniformly random from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$ and $z = x_m^S$, it is clear that Algorithm 2 satisfies the **G.2** requirement of GRADIENT-FOCUSED-OPTIMIZER with $g(n, \epsilon) = T_\epsilon / 40Ln^{2/3}$. Since both **G.1** and **G.2** are satisified for Algorithm 2, we conclude that SVRG is a GRADIENT-FOCUSED-OPTIMIZER. $\square$

Algorithm 3 HESSIANDESCENT (x, ϵ, γ)

```
1: Find $v$ such that $\|v\| = 1$, and with probability at least $\rho$ the following inequality holds: $\langle v, \nabla^2 f(x) v\rangle \leq \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2}$.
2: Set $\alpha = |\langle v, \nabla^2 f(x) v\rangle|/M$.
3: $u = x - \alpha\, \text{sign}(\langle v, \nabla f(x)\rangle) v$.
4: $y = \arg\min_{z \in \{u,x\}} f(z)$
5: Output: $(y, \diamond)$.
```

D Hessian Descent and Proof of Lemma 2

The approach is based on directly using the eigenvector corresponding to the smallest eigenvalue as a HESSIAN-FOCUSED-OPTIMIZER. More specifically, when the smallest eigenvalue of the Hessian is negative and reasonably large in magnitude, the Hessian information can be used to ensure descent in the objective function value. The pseudo-code for the algorithm is given in Algorithm 3.

The key idea is to utilize the minimum eigenvalue information in order to make a descent step. If $\lambda_{\min}(\nabla^2 f(x)) \leq -\gamma$ then the idea is to use this information to take a descent step. Note the subroutine is designed in a fashion such that the objective function value never increases. Thus, it naturally satisfies the requirement **H.1** of HESSIAN-FOCUSED-OPTIMIZER.

Preposition 1. *The time complexity of finding $v \in \mathbb{R}^d$ that $\|v\| = 1$, and with probability at least ρ the following inequality holds: $\langle v, \nabla^2 f(x) v\rangle \leq \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2}$ is $O(nd + n^{3/4} d/\gamma^{1/2})$.*

Note that each iteration of Algorithm 1 in this case has just linear dependence on d. Since the total number of HESSIANDESCENT iterations is $O(\Delta/\min(p, 1-p)\epsilon^{3/2})$ and each iteration has the complexity of $O(nd + n^{3/4}d/\epsilon^{1/4})$, using the above remark, we obtain an overall time complexity of HESSIANDESCENT is $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$.

Proof of Lemma 2. The first important observation is that the function value never increases because $y = \arg\min_{z\in\{u,x\}} f(z)$ i.e., $f(y) \leq f(x)$, thus satisfying **H.1** of HESSIAN-FOCUSED-OPTIMIZER. We now analyze the scenario where $\lambda_{min}(\nabla^2 f(x)) \leq -\gamma$. Consider the event where we obtain v such that

$$\langle v, \nabla^2 f(x) v\rangle \leq \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2}.$$

This event (denoted by $\mathcal{E}$) happens with at least probability ρ. Note that, since $\lambda_{min}(\nabla^2 f(x)) \leq -\gamma$, we have $\langle v, \nabla^2 f(x) v\rangle \leq -\frac{\gamma}{2}$. In this case, we have the following relationship:

$$\begin{aligned} f(y) &\leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x) + \frac{M}{6}\|y-x\|^3 \\ &= f(x) - \alpha|\langle \nabla f(x), v\rangle| + \frac{\alpha^2}{2} v^T \nabla^2 f(x) v + \frac{M\alpha^3}{6}\|v\|^3 \\ &\leq f(x) + \frac{\alpha^2}{2} v^T \nabla^2 f(x) v + \frac{M\alpha^3}{6} \\ &\leq f(x) - \frac{1}{2M^2}|v^T \nabla^2 f(x) v|^3 + \frac{1}{6M^2}|v^T \nabla^2 f(x) v|^3 \\ &= f(x) - \frac{1}{3M^2}|v^T \nabla^2 f(x) v|^3 \leq f(x) - \frac{1}{24M^2}\gamma^3. \end{aligned} \tag{13}$$

The first inequality follows from the M-lipschitz continuity of the Hessain $\nabla^2 f(x)$. The first equality follows from the update rule of HESSIANDESCENT. The second inequality is obtained by dropping the negative term and using the fact that $\|v\| = 1$. The second equality is obtained by substituting $\alpha = \frac{|v^T \nabla^2 f(x) v|}{M}$. The last inequality is due to the fact that$\langle v, \nabla^2 f(x) v\rangle \leq -\frac{\gamma}{2}$. In the other scenario where

$$\langle v, \nabla^2 f(x) v\rangle \leq \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2},$$

we can at least ensure that $f(y) \leq f(x)$ since $y = \arg\min_{z\in\{u,x\}} f(z)$. Therefore, we have

$$\begin{aligned} \mathbb{E}[f(y)] &= \rho\mathbb{E}[f(y)|\mathcal{E}] + (1-\rho)\mathbb{E}[f(y)|\bar{\mathcal{E}}] \\ &\leq \rho\mathbb{E}[f(y)|\mathcal{E}] + (1-\rho) f(x) \\ &\leq \rho\left[f(x) - \tfrac{\rho}{24M^2}\gamma^3\right] + (1-\rho) f(x) \\ &= f(x) - \tfrac{\rho}{24M^2}\gamma^3. \end{aligned} \tag{14}$$

The last inequality is due to Equation (13). Hence, HESSIAN-FOCUSED-OPTIMIZER satisfies **H.2** of HESSIAN-FOCUSED-OPTIMIZER with $h(n, \epsilon, \gamma) = \frac{\rho}{24M^2}\gamma^3$, thus concluding the proof. □

E Other Lemmas

The following bound on the variance of SVRG is useful for our proof [26].

Lemma 3. *[26] Let v_t^{s+1} be computed by Algorithm 2. Then,*

$$\mathbb{E}[\|v_t^{s+1}\|^2] \leq 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] + 2L^2\mathbb{E}[\|x_t^{s+1} - \tilde{x}^s\|^2].$$

Proof. We use the definition of v_t^{s+1} to get

$$\begin{aligned}\mathbb{E}[\|v_t^{s+1}\|^2] &= \mathbb{E}[\| \left(\nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s)\|^2] \\ &= \mathbb{E}[\| \left(\nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\right) + \nabla f(\tilde{x}^s) - \nabla f(x_t^{s+1}) + \nabla f(x_t^{s+1})\|^2] \\ &\leq 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] + 2\mathbb{E}\left[\left\|\nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s) - \mathbb{E}[\nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)]\right\|^2\right]\end{aligned}$$

The inequality follows from the simple fact that $(a+b)^2 \leq a^2 + b^2$. From the above inequality, we get the following:

$$\begin{aligned}\mathbb{E}[\|v_t^{s+1}\|^2] &\leq 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] + 2\mathbb{E}\|\nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s)\|^2 \\ &\leq 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] + 2L^2\mathbb{E}[\|x_t^{s+1} - \tilde{x}^s\|^2]\end{aligned}$$

The first inequality follows by noting that for a random variable ζ, $\mathbb{E}[\|\zeta - \mathbb{E}[\zeta]\|^2] \leq \mathbb{E}[\|\zeta\|^2]$. The last inequality follows from L-smoothness of f_{i_t}. □

F Cubic Regularization and its Approximation

In this section, we show that the cubic regularization method in [23] can be used as HESSIAN-FOCUSED-OPTIMIZER. More specifically, here HESSIAN-FOCUSED-OPTIMIZER approximately solves the following optimization problem:

$$y = \arg\min_z \langle \nabla f(x), z - x\rangle + \frac{1}{2}\left\langle z - x, \nabla^2 f(x)(z-x)\right\rangle + \frac{M}{6}\|z - x\|^3, \qquad \text{(CUBICDESCENT)}$$

and returns $(y, \diamond)$ as output. The following result can be proved for this approach.

Theorem 3. *Suppose* SVRG *(same as Theorem 2) is used as* GRADIENT-FOCUSED-OPTIMIZER *and* CUBICDESCENT *is used as* HESSIAN-FOCUSED-OPTIMIZER *with $q = 0$, then Algorithm 1 finds a $(\epsilon, \sqrt{\epsilon})$-second order critical point in $T = O(\Delta/\min(p, 1-p)\epsilon^{3/2})$ with probability at least* 0.9.

Proof. First note that cubic method is a descent method (refer to Theorem 1 of [23]); thus, **H.1** is trivially satisfied. Furthermore, cubic descent is a HESSIAN-FOCUSED-OPTIMIZER with $h(n, \epsilon, \gamma) = \frac{2\gamma^3}{81M^3}\gamma^3$. This, again, follows from Theorem 1 of [23]. The result easily follows from the aforementioned observations. □

In principle, Algorithm 1 with CUBICDESCENT as HESSIAN-FOCUSED-OPTIMIZER can converge without the use of GRADIENT-FOCUSED-OPTIMIZER subroutine at each iteration since it essentially reduces to the cubic regularization method of [23]. However, in practice, we would expect GRADIENT-FOCUSED-OPTIMIZER to perform most of the optimization and HESSIAN-FOCUSED-OPTIMIZER to be used for far fewer iterations. Using the method developed in [23] for solving CUBICDESCENT, we obtain the following corollary.

Corollary 2. *The overall running time of Algorithm 1 to find a $(\epsilon, \sqrt{\epsilon})$-second order critical point, with parameter settings used in Theorem 3, is $O(nd^\omega/\epsilon^{3/2} + n^{2/3}d/\epsilon^2)$.*

Here ω is the matrix multiplication constant. The dependence on ϵ is weaker in comparison to Corollary 1. However, each iteration of CUBICDESCENT is expensive (as seen from the factor d^ω in the corollary above) and thus, in high dimensional settings typically encountered in machine learning, this approach can be expensive in comparison to HESSIANDESCENT.

Cubic regularization method of Nesterov and Polyak [23] is designed to operate on full batch, i.e., it does not exploit the finite-sum structure of the problem and requires the computation of the gradient and the Hessian on the entire dataset to make an update. However, such full-batch methods do not scale gracefully with the size of data and become prohibitively expensive on large datasets. To overcome this challenge, we devised an approximate cubic regularization method described below:

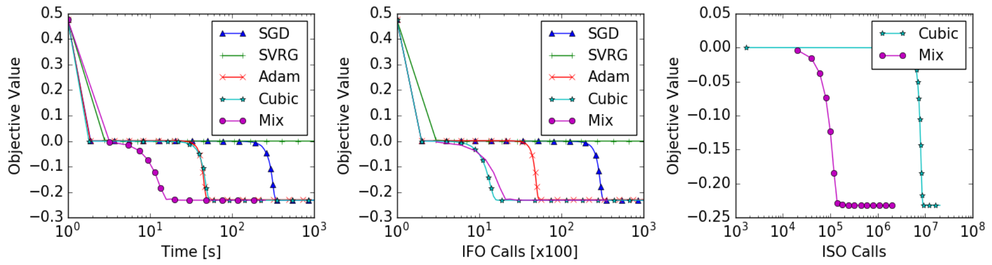

Figure 3: Comparison of various methods on a synthetic problem. Our mix framework successfully escapes saddle point and uses relatively few ISO calls in comparison to CUBICDESCENT.

1. Pick a mini-batch $\mathcal{B}$ and obtain the gradient and the hessian based on $\mathcal{B}$, i.e.,

$$g = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla f_i(x) \qquad H = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla^2 f_i(x) \tag{15}$$

2. Solve the sub-problem

$$v^* = \arg\min_v \langle g, v\rangle + \frac{1}{2}\langle v, Hv\rangle + \frac{M}{6}\|v\|^3 \tag{16}$$

3. Update: $x \leftarrow x + v^*$

We found that this mini-batch training strategy, which requires the computation of the gradient and the Hessian on a small subset of the dataset, to work well on a few datasets (CURVES, MNIST, CIFAR10). A similar method has been analysed in [5].

Furthermore, in many deep-networks, adaptive per-parameter learning rate helps immensely [13]. One possible explanation for this is that the scale of the gradients in each layer of the network often differ by several orders of magnitude. A well-suited optimization method should take this into account. This is the reason for popularity of methods like ADAM or RMSPROP in the deep learning community. On similar lines, to account for different per-parameter behaviour in cubic regularization, we modify the sub-problem by adding a diagonal matrix M_d in addition to the scalar regularization coefficient M, i.e.,

$$\min_v \langle g, v\rangle + \frac{1}{2}\langle v, Hv\rangle + \frac{1}{6}M\|M_d v\|^3. \tag{17}$$

Also we devised an adaptive rule to obtain the diagonal matrix as $M_d = \mathsf{diag}((s + 10^{-12})^{1/9})$, where s is maintained as a moving average of third order polynomial of the mini-batch gradient g, in a fashion similar to RMSPROP and ADAM:

$$s \leftarrow \beta s + (1-\beta)(|g|^3 + 2g^2), \tag{18}$$

where $|g|^3$ and g^2 are vectors such that $[|g|^3]_i = |g_i|^3$ and $[g^2]_i = g_i^2$ respectively for all $i \in [n]$. The experiments reported on CURVES and MNIST in this paper utilizes both the above modifications to the cubic regularization, with β set to 0.9. We refer to this modified procedure as ACubic in our results.

G Experiment Details

In this section we provide further experimental details and results to aid reproducibility.

Synthetic Problem To demonstrate the fast escape from a saddle point by the proposed method, we consider the following simple nonconvex finite-sum problem:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} x^T A_i x + b_i^T x + \|x\|_{10}^{10} \tag{19}$$

Here the parameters are designed such that $\sum_i b_i = 0$ and $\sum_i A_i$ matrix has exactly one negative eigenvalue of -0.001 and other eigenvalues randomly chosen in the interval $[1, 2]$. The total number of examples n is set to be 100,000 and d is 1000. It is not hard to see that this problem has a non-degenerate saddle point at the origin. This allows us to explore the behaviour of different optimization algorithms in the vicinity of the saddle point. In this experiment, we compare a mix of SVRG and HESSIANDESCENT (as in Theorem 2) with SGD (with constant step size), ADAM, SVRG and CUBICDESCENT. The parameter of these algorithms is chosen by grid search so that it gives the best performance. The subproblem of CUBICDESCENT was solved with gradient descent [4] until the gradient norm of the subproblem is reduced below 10^{-3}. We study the progress

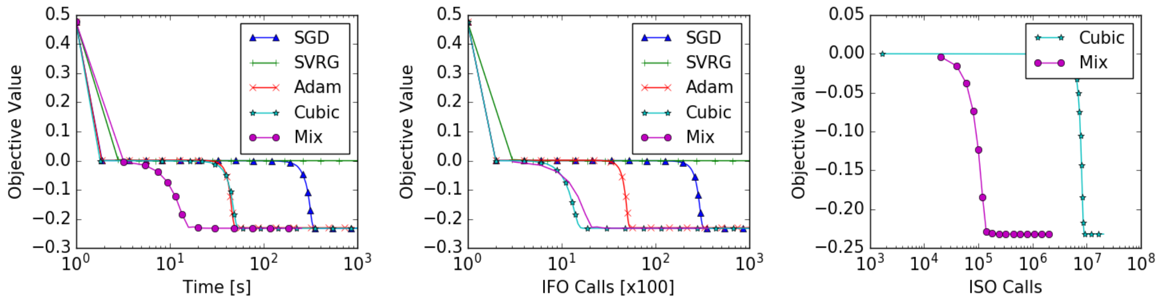

Figure 4: Comparison of various methods on a synthetic problem. Our mix framework successfully escapes saddle point.

of optimization, i.e., decrease in function value with wall clock time, IFO calls, and ISO calls. All algorithms were initialized with the same starting point very close to origin.

The results are presented in Figure 3, which shows that our proposed mix framework was the *fastest* to escape the saddle point in terms of wall clock time. We observe that performance of the first order methods suffered severely due to the saddle point. Note that SGD eventually escaped the saddle point due to inherent noise in the mini-batch gradient. CUBICDESCENT, a second-order method, escaped the saddle point faster in terms of iterations using the Hessian information. But operating on Hessian information is expensive as a result this method was slow in terms of wall clock time. The proposed framework, which is a mix of the two strategies, inherits the best of both worlds by using cheap gradient information most of the time and reducing the use of relatively expensive Hessian information (ISO calls) by 100x. This resulted in *faster* escape from saddle point in terms of wall clock time.

G.1 Synthetic Problem

The parameter selection for all the methods were carried as follows:

1. SGD: The scalar step-size was determined by a grid search.
2. ADAM: We performed a grid search over α and ε parameters of ADAM tied together, i.e., $\alpha = \varepsilon$.
3. SVRG: The scalar step-size was determined by a grid search.
4. CUBICDESCENT: The regularization parameter M was chosen by grid search. The sub-problem was solved with gradient descent [4] with the step-size of solver to be 10^{-2} and run till the gradient norm of the sub-problem is reduced below 10^{-3}.

Further Observations The results are presented in Figure 4. The other first order methods like ADAM with higher noise could escape relatively faster whereas SVRG with reduced noise stayed stuck at the saddle point.

G.2 Deep Networks

Methods The parameter selection for all the methods were carried as follows::

1. ADAM: We performed a grid search over α and ε parameters of ADAM so as to produce the best generalization on a held out test set. We found it to be $\alpha = 10^{-3}, \varepsilon = 10^{-3}$ for CURVES and $\alpha = 10^{-2}, \varepsilon = 10^{-1}$ for MNIST.
2. APPROXCUBICDESCENT: The regularization parameter M was chosen as the largest value such function value does not jump in first 10 epochs. We found it to be $M = 10^3$ for both CURVES and MNIST. The sub-problem was solved with gradient descent [4] with the step-size of solver to be 10^{-3} and run till the gradient norm of the sub-problem is reduced below 0.1.

H Discussion

In this paper, we examined a generic strategy to escape saddle points in nonconvex finite-sum problems and presented its convergence analysis. The key intuition is to alternate between a first-order and second-order based optimizers; the latter is mainly intended to escape points that are only stationary but are not second-order critical points. We presented two different instantiations of our framework and provided their detailed convergence analysis. While both our methods explicity use the Hessian information, one can also use noisy first-order methods as HESSIAN-FOCUSED-OPTIMIZER (see for e.g. noisy SGD in [8]). In such a scenario, we exploit the negative eigenvalues of the Hessian to escape saddle points by using isotropic noise, and do not explicitly use ISO. For these methods, under strict-saddle point property [8], we can show convergence to local optima within our framework.

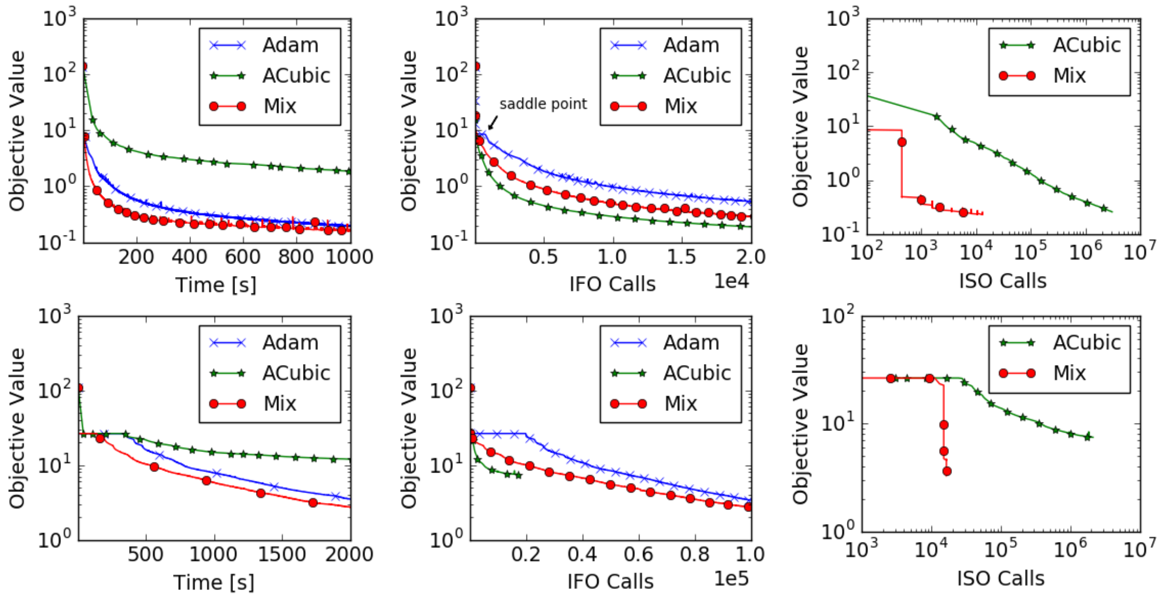

Figure 5: Comparison of various methods on a Deep Autoencoder on CURVES (top) and MNIST (bottom). Our mix approach converges faster than the baseline methods and uses relatively few ISO calls in comparison to APPROXCUBICDESCENT

We primarily focused on obtaining second-order critical points for nonconvex finite-sums (1). This does not necessarily imply low test error or good generalization capabilities. Thus, we should be careful when interpreting the results presented in this paper. A detailed discussion or analysis of these issues is out of scope of this paper. While a few prior works argue for convergence to local optima, the exact connection between generalization and local optima is not well understood, and is an interesting open problem. Nevertheless, we believe the techniques presented in this paper can be used alongside other optimization tools for faster and better nonconvex optimization.