A Generic Approach for Escaping Saddle points

Sashank J Reddi*, Manzil Zaheer*
Carnegie Mellon University

Suvrit Sra
MIT LIDS

Barnabás Póczos
Carnegie Mellon University

Ruslan Salakhutdinov
Carnegie Mellon University

Francis Bach
INRIA-ENS-Université PSL

Alexander J Smola
Amazon Web Services

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 a 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 neither $f : \mathbb{R}^d \to \mathbb{R}$ nor the individual functions $f_i : \mathbb{R}^d \to \mathbb{R}$ ($i \in [n]$) are necessarily convex. We operate in a general nonconvex setting except for few smoothness assumptions like Lipschitz continuity of the gradient and Hessian. Optimization problems of this form arise naturally in machine learning and statistics as empirical risk minimization (ERM) and M-estimation respectively.

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 has the iterative update:

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t), \tag{2}$$

where $i_t \in [n]$ is a randomly chosen index and η_t is a learning rate. Under suitable selection of the learning rate, we can show that SGD converges to a point x that, in expectation, satisfies the stationarity condition $\|\nabla f(x)\| \leq \epsilon$ in $O(1/\epsilon^4)$ iterations [Ghadimi and Lan, 2013]. This result has two critical weaknesses: (i) It does not ensure convergence to local optima or second-order critical points; (ii) The rate of convergence of the SGD algorithm is slow.

For general nonconvex problems, one has to settle for a more modest goal than sub-optimality, as finding the global minimizer of finite-sum nonconvex problem will be in general intractably hard. Unfortunately, SGD does not even ensure second-order critical conditions such as local optimality since it can get stuck at saddle points. This issue has recently received considerable attention in the ML community, especially in the context of deep learning [Dauphin et al., 2014, 2015, Choromanska et al., 2014]. These works argue that saddle points are highly prevalent in most optimization paths, and are the primary obstacle for training large deep networks. To tackle this issue and achieve a second-order critical point for which $\|\nabla f\| \leq \epsilon$ and $\nabla^2 f \succeq -\sqrt{\epsilon}\mathbb{I}$, we need algorithms that either use the Hessian explicitly or exploit its structure.

A key work that explicitly uses Hessians to obtain faster convergence rates is the cubic regularization (CR) method [Nesterov and Polyak, 2006]. In particular, Nesterov and Polyak [2006] 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. [2016a] designed an

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

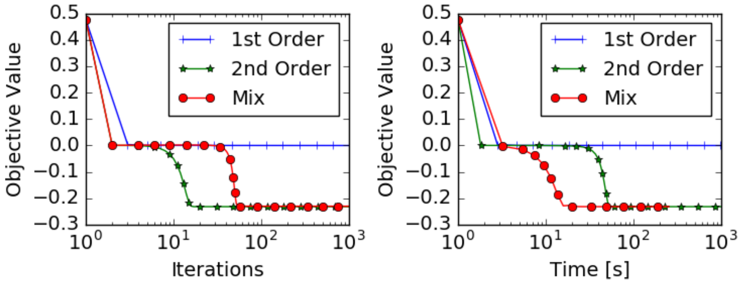

Figure 1: First order methods like GD can potentially get stuck at saddle points. Second-order methods can escape it in very few iterations (as observed in the left plot) but at the cost of expensive Hessian based iterations (see time plot to the right). The proposed framework, which is a novel mix of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

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 [Ge et al., 2015], 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 [Ge et al., 2015]. More recently, Carmon et al. [2016] presented a new faster algorithm that alternates between first-order and second-order subroutines. However, their algorithm is designed for the simple case of $n = 1$ in (1) and hence, can be expensive in practice.

Inspired by this line of work, we develop a general framework for finding second-order critical points. The key idea of our framework 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. We summarize the key idea and main contributions of this paper below.

Main Contributions: We develop an algorithmic framework for converging to second-order critical points and provide convergence analysis for it. Our framework carefully alternates between two subroutines that use gradient and Hessian information, respectively, and ensures second-order criticality. Furthermore, we present two instantiations of our framework and provide convergence rates for them. In particular, we show that a simple instance of our framework, based on SVRG, achieves convergence rates competitive with the current state-of-the-art methods; thus highlighting the simplicity and applicability of our framework. Finally, we demonstrate the empirical performance of a few algorithms encapsulated by our framework and show their superior performance.

Related Work. 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 Robbins and Monro [1951]. Since then, SGD has been a subject of extensive research, especially in the convex setting [Poljak and Tsypkin, 1973, Ljung, 1977, Bottou, 1991, Kushner and Clark, 2012]. 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., [Defazio et al., 2014a, Johnson and Zhang, 2013, Schmidt et al., 2013, Konečný et al., 2015, Shalev-Shwartz and Zhang, 2013, Defazio et al., 2014b]. Accelerated variants of these methods achieve the lower bounds proved in [Agarwal and Bottou, 2014, Lan and Zhou, 2015], thereby settling the question of their optimality. Furthermore, Reddi et al. [2015] 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 by Ghadimi and Lan [2013], 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 [Lian et al., 2015]. For these problems, Reddi et al. [2016a,b,c] 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 Nesterov and Polyak [2006]. However, each iteration of the algorithm in [Nesterov and Polyak, 2006] is prohibitively expensive since it requires eigenvalue decompositions, and hence, is unsuitable for large-scale high-dimensional problems. More recently, Carmon et al. [2016], Agarwal et al. [2016a] presented algorithms for finding second-order critical points by tackling some practical issues that arise in [Nesterov and Polyak, 2006]. However, these algorithms are either only applicable to a restricted setting or heavily use Hessian based computations, making them unappealing from a practical standpoint. Also, recently Carmon et al. [2017] proposed a fast accelerated first-order method for nonconvex optimization; however, similar to Carmon et al. [2016], it requires computation of the full gradient at each iteration and only provides convergence to stationary points. *Noisy* variants of first-order methods have also been shown to escape saddle points (see [Ge et al., 2015, Jin et al., 2017, Levy, 2016]), however, they have strong dependence on either n or d, both of which are undesirable.

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 Lipschitz, i.e., we have

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq M\|x-y\|, \tag{3}$$

for all $x, y \in \mathbb{R}^d$. Such a condition is typically necessary to ensure convergence of algorithms to the second-order critical points [Nesterov and Polyak, 2006]. In addition to the above smoothness conditions, 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 Nesterov [2003], Ghadimi and Lan [2013], Nesterov and Polyak [2006], 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 $\nabla^2 f(x) \succeq -\gamma\mathbb{I}$ [Nesterov and Polyak, 2006]. For iterative algorithms, we require both $\epsilon, \gamma \to 0$ as the number of iterations $T \to \infty$. When all saddle points are non-degenerate, such a condition implies convergence to a local optimum.

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 $\nabla^2 f(x) \succeq -\gamma\mathbb{I}$, where the expectation is over any randomness in $\mathcal{A}$.*

We must exercise caution while interpreting results pertaining to (ϵ, γ)-second order critical points. Such points need not be close to any local minima either in objective function value, or in the domain of (1). For our algorithms, we use only an Incremental First-order Oracle (IFO) [Agarwal and Bottou, 2014] and an Incremental Second-order Oracle (ISO), defined below.

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 $\nabla^2 f_i(x)v$.*

IFO and ISO calls are typically cheap, with ISO call being relatively more expensive. In many practical settings that arise in machine learning, the time complexity of these oracle calls is linear in d [Agarwal et al., 2016b, Pearlmutter, 1994]. 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

In this section, we propose a generic framework for escaping saddle points while solving nonconvex problems of form (1). One of the primary difficulties in reaching a second-order critical point is the presence of saddle points. To evade such 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 local minimizer since it can get stuck at a saddle point. Hence, we require the subroutine Hessian-Focused-Optimizer to help avoid saddle points. A natural idea is to interleave these subroutines to obtain a second-order critical point. But it is not even clear if such a procedure even converges. We propose a carefully designed procedure that effectively balances these two subroutines, 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, since it does not specify the subroutines Gradient-Focused-Optimizer and Hessian-Focused-Optimizer. These subroutines determine the crucial update mechanism of the algorithm. We will present specific instance of these

Algorithm 1 Generic Framework

```
Input - Initial point: $x^0$, total iterations $T$, error threshold parameters $\epsilon$, $\gamma$ and probability $p$
for $t = 1$ to $T$ do
    $(y^t, z^t)$ = Gradient-Focused-Optimizer$(x^{t-1}, \epsilon)$ (refer to G.1 and G.2)
    Choose $u^t$ as $y^t$ with probability $p$ and $z^t$ with probability $1 - p$
    $(x^{t+1}, \tau^{t+1})$ = Hessian-Focused-Optimizer$(u^t, \epsilon, \gamma)$ (refer to H.1 and H.2)
    if $\tau^{t+1} = \emptyset$ then
        Output set $\{x^{t+1}\}$
    end if
end for
Output set $\{y^1, ..., y^T\}$
```

subroutines in the next section, but we assume the following properties to hold for these subroutines.

- Gradient-Focused-Optimizer: Suppose (y, z) = Gradient-Focused-Optimizer(x, n, ϵ), 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, τ) = Hessian-Focused-Optimizer(x, n, ϵ, γ) where $y \in \mathbb{R}^d$ and $\tau \in \{\emptyset, \diamond\}$. If $\tau = \emptyset$, 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 function $h : \mathbb{N} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$.

  Here the expectation is over any randomness in subroutine Hessian-Focused-Optimizer. The two conditions ensure that the objective function value, in expectation, never increases and furthermore, 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.

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. In order to provide a concrete implementation, we need to specify the aforementioned subroutines. Before we delve into those details, we will provide a generic convergence analysis for Algorithm 1.

Convergence Analysis

Theorem 1. *Let $\Delta = f(x^0) - B$, and $\theta = \min\{(1 - p)\epsilon^2 g(n, \epsilon), ph(n, \epsilon, \gamma)\}$. Also, let Γ be the set output by Algorithm 1 with* Gradient-Focused-Optimizer *satisfying **G.1** and **G.2** and* Hessian-Focused-Optimizer *satisfying **H.1** and **H.2**. Further, let T be such that $T > \Delta/\theta$.*

Suppose the multiset $S = \{i_1, ...i_k\}$ contains k indices selected independently and uniformly from $\{1, ..., |\Gamma|\}$. Then the following holds for the indices in S:

1. *y^t (where $t \in S$) is an (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 $t \in S$ is an (ϵ, γ)-critical point.*

The proof of the result is presented in Appendix A. 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 $1/p = 1/\epsilon^2 g(n, \epsilon) + 1/h(n, \epsilon, \gamma)$. We defer further discussion of convergence to next section, where we present more specific convergence and rate analysis.

4 Concrete Instantiations

We now present specific instantiations of our framework in this section. Before we state our key results, we discuss an important subroutine that is used as Gradient-Focused-Optimizer for rest of this paper: Svrg. We give a brief description of the algorithm in this section and show that it meets the conditions required for a Gradient-Focused-Optimizer. Svrg Johnson and Zhang [2013], Reddi et al. [2016a] 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.

Algorithm 2 SVRG$\big(x^0, \epsilon\big)$

```
Input: $x_m^0 = x^0 \in \mathbb{R}^d$, epoch length $m$, step sizes $\{\eta_i > 0\}_{i=0}^{m-1}$, iterations $T_g$, $S = \lceil T_g/m \rceil$
for $s = 0$ to $S - 1$ do
    $\tilde{x}^s = x_0^{s+1} = x_m^s$
    $g^{s+1} = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x}^s)$
    for $t = 0$ to $m - 1$ do
        Uniformly randomly pick $i_t$ from $\{1, \dots, n\}$
        $v_t^{s+1} = \nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s) + g^{s+1}$
        $x_{t+1}^{s+1} = x_t^{s+1} - \eta_t v_t^{s+1}$
    end for
end for
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 $z = x_m^S$.
```

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 [Johnson and Zhang, 2013, Reddi et al., 2016a]. The following result shows that Svrg meets the requirements of a Gradient-Focused-Optimizer.

Lemma 1. *Suppose $\eta_t = \eta = 1/4Ln^{2/3}$, $m = n$ and $T_g = T_\epsilon$, which depends on ϵ, then Algorithm 2 is a* Gradient-Focused-Optimizer *with $g(n, \epsilon) = T_\epsilon/40Ln^{2/3}$.*

In rest of this section, we discuss approaches using Svrg as a Gradient-Focused-Optimizer. In particular, we propose and provide convergence analysis for two different methods with different Hessian-Focused-Optimizer but which use Svrg as a Gradient-Focused-Optimizer.

4.1 Hessian descent

The first 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. The following result shows that HessianDescent is a Hessian-Focused-Optimizer.

Algorithm 3 HessianDescent (x, ϵ, γ)

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

Lemma 2. HessianDescent *is a* Hessian-Focused-Optimizer *with $h(n, \epsilon, \gamma) = \frac{\rho}{24M^2}\gamma^3$.*

The proof of the result is presented in Appendix C. With Svrg as Gradient-Focused-Optimizer and HessianDescent as Hessian-Focused-Optimizer, we 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.

The result directly follows from using Lemma 1 and 2 in Theorem 1. The result shows that the iteration complexity of Algorithm 1 in this case is $O(\Delta/\epsilon^{3/2}\min(p, 1-p))$. Thus, the overall IFO complexity of Svrg algorithm is $(n + T_g) \times T = O(n/\epsilon^{3/2} + n^{2/3}/\epsilon^2)$. Since each IFO call takes $O(d)$ time, the overall time complexity of all Gradient-Focused-Optimizer steps is $O(nd/\epsilon^{3/2} + n^{2/3}d/\epsilon^2)$. To understand the time complexity of HessianDescent, we need the following result due to Agarwal et al. [2016a].

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 $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$. Combining this with the time complexity of Svrg, 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 pa-*

rameter 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 [Ge et al., 2015]. Furthermore, our results are competitive with [Agarwal et al., 2016a, Carmon et al., 2016] in their respective settings, but with a much more practical algorithm and simpler analysis. More specifically, Agarwal et al. [2016a] has running time of $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$. When $\epsilon > c/n^{2/3}$ where c is some constant (which is very reasonable for most machine learning settings), our results would be better. However, our rates can be slightly worse when this is not the case. Next, Carmon et al. [2016] does not consider the finite-sum setting specifically, and if applied as it is for our finite-sum setup, it will have a running time of $O(nd/\epsilon^{7/4})$. In comparison, our rates are better in a larger regime where $\epsilon > c/n^{4/3}$. Note that in machine learning applications the goal is to obtain better generalization which roughly translates to such regimes in practice. Thus, in settings of interest to machine learning, our results are competitive (or better) in comparison to [Agarwal et al., 2016a, Carmon et al., 2016]. Finally, we also note that our algorithm is faster than the one proposed in [Jin et al., 2017], which has a time complexity of $O(nd/\epsilon^2)$.

4.2 Cubic Descent

In this section, we show that the cubic regularization method in Nesterov and Polyak [2006] can be used as Hessian-Focused-Optimizer. More specifically, here Hessian-Focused-Optimizer approximately solves the following optimization problem:

$$\begin{aligned} y = \arg\min_z \langle \nabla f(x), z - x\rangle \\ + \frac{1}{2}\left\langle z - x, \nabla^2 f(x)(z - x)\right\rangle \\ + \frac{M}{6}\|z - x\|^3, \end{aligned} \tag{\textsc{CubicDescent}}$$

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*.*

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 Nesterov and Polyak [2006]. 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 Nesterov and Polyak [2006] 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.

4.3 Practical Considerations

The focus of this section was to demonstrate the wide applicability of our framework; wherein using a simple instantiation of this framework, we could achieve algorithms with fast convergence rates. 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 [Kingma and Ba, 2014] 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 in the following section.

5 Experiments

We now present empirical results for our saddle point avoidance technique with an aim to highlight three aspects of our framework: (i) it successfully escapes non-degenerate saddle points, (ii) it is fast, and (iii) it is practical on large-scale problems. All the algorithms are implemented on TensorFlow [Abadi et al., 2015]. In case of deep networks, the Hessian-vector product is evaluated using the trick presented in [Pearlmutter, 1994]. We run our experiments on a commodity machine with Intel® Xeon® CPU E5-2630 v4 CPU, 256GB RAM, and NVidia® Titan X (Pascal) GPU.

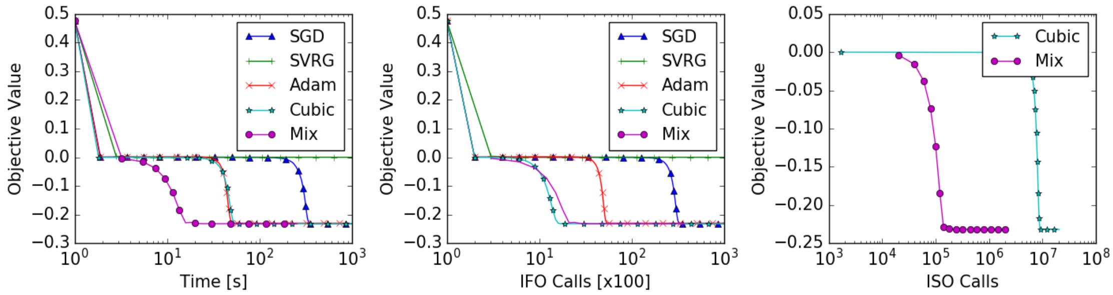

Figure 2: Comparison of various methods on a synthetic problem. Our mix framework successfully escapes saddle point and uses relatively few ISO calls in comparison to CubicDescent.

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

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} x^T A_i x + b_i^T x + \|x\|_{10}^{10} \qquad (4)$$

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 Carmon et al. [2016] until the gradient norm of the subproblem is reduced below 10^{-3}. We study the progress 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 2, 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.

Deep Networks. To investigate the practical performance of the framework for deep learning problems, we applied it to two deep autoencoder optimization problems from Hinton and Salakhutdinov [2006] 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. Martens and Grosse [2015], Sutskever et al. [2013], Vinyals and Povey [2012], Martens [2010]. The CURVES autoencoder consists of an encoder with layers of size (28×28)-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. [1] The MNIST autoencoder consists of an encoder with layers of size (28×28)-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.[2]

As an instantiation of our framework, we use a mix of Adam, which is popular in deep learning community, and approximate cubic regularization method described in Section 4.3. For comparison the algorithm in [Carmon et al., 2016] is impractical for our setting because it requires computing the full gradient at each iteration and is, thus, intractable for problems of our interest. Furthermore, we tried to compare our algorithm with the method in [Agarwal et al., 2016a], however, the algorithm as stated in the paper is not prac-

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

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

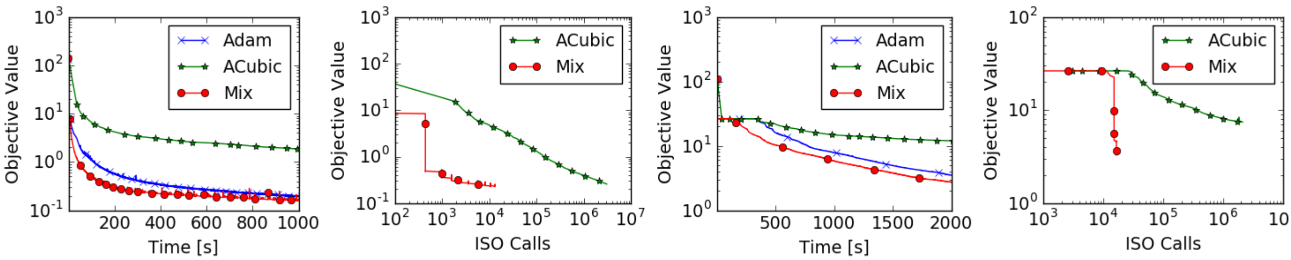

Figure 3: Comparison of various methods on CURVES and MNIST Deep Autoencoder. Our mix approach converges faster than the baseline methods and uses relatively few ISO calls in comparison to ApproxCubicDescent. Please refer to Appendix G for more results.

tical[3]. To this end, we compare our algorithm with a practical version of cubic method, ApproxCubicDescent, described in Section 4.3. We also compared our algorithm with Adam, which is typically the de-facto method for training large deep networks. 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 Martens [2010] 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 presented in Figure 3 show 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, resulting in fast decrease of training loss.

6 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. In this paper, we primarily used Svrg as Gradient-Focused-Optimizer, however, investigating the use of other first-order methods, like Carmon et al. [2017], is an interesting research direction. Also, 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 [Ge et al., 2015]). 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 [Ge et al., 2015], we can show convergence to local optima within our framework.

Our primary goal in this paper was to develop a well-founded, yet practical, framework for finding local minima in nonconvex optimization. While convergence rates in [Agarwal et al., 2016a] may seem slightly better at first glance, we would like to point out that these running times are worst-case time complexity and might involve large constants. For example, Agarwal et al. [2016a] solve a subroutine based on both gradient and Hessian information at *each* iteration. Such methods do not scale well in practice because they rely on expensive computations based on the Hessian at every iteration.

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.

[3] We would like to emphasize that [Agarwal et al., 2016b] does not contain any empirical results. In order to be fair, we also reached out to the authors of the paper, who via personal communication admitted that the algorithm as implemented verbatim in the paper is not practical and comparison to it is not useful.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

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

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, 2016a.

Naman Agarwal, Brian Bullins, and Elad Hazan. Second order stochastic optimization in linear time. *CoRR*, abs/1602.03943, 2016b.

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

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

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. "convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 654–663, 2017.

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.

Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. *CoRR*, abs/1412.0233, 2014.

Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1504–1512. Curran Associates, Inc., 2015.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 2933–2941, 2014.

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, 2014a.

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

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.

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, January 1994. ISSN 0899-7667.

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

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.

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, 2016a.

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

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

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

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

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.

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.

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