Complexity of Finding Stationary Points of Nonsmooth Nonconvex Functions

Jingzhao Zhang [1] **Hongzhou Lin** [1] **Stefanie Jegelka** [1] **Suvrit Sra** [1] **Ali Jadbabaie** [1]

Abstract

We provide the first *non-asymptotic* analysis for finding stationary points of nonsmooth, nonconvex functions. In particular, we study the class of Hadamard semi-differentiable functions, perhaps the largest class of nonsmooth functions for which the chain rule of calculus holds. This class contains examples such as ReLU neural networks and others with non-differentiable activation functions. We first show that finding an ϵ-stationary point with first-order methods is impossible in finite time. We then introduce the notion of (δ, ϵ)-*stationarity*, which allows for an ϵ-approximate gradient to be the convex combination of generalized gradients evaluated at points within distance δ to the solution. We propose a series of randomized first-order methods and analyze their complexity of finding a (δ, ϵ)-stationary point. Furthermore, we provide a lower bound and show that our stochastic algorithm has min-max optimal dependence on δ. Empirically, our methods perform well for training ReLU neural networks.

1. Introduction

Gradient based optimization underlies most of machine learning and it has attracted tremendous research attention over the years. While non-asymptotic complexity analysis of gradient based methods is well-established for convex and *smooth* nonconvex problems, little is known for nonsmooth nonconvex problems. We summarize the known rates (black) in Table 1 based on the references (Nesterov, 2018; Carmon et al., 2017; Arjevani et al., 2019).

Within the nonsmooth nonconvex setting, recent research results have focused on asymptotic convergence analysis (Benaïm et al., 2005; Kiwiel, 2007; Majewski et al., 2018; Davis et al., 2018; Bolte & Pauwels, 2019). Despite their advances, these results fail to address finite-time, non-asymptotic convergence rates. Given the widespread use of nonsmooth nonconvex problems in machine learning, a canonical example being deep ReLU neural networks, obtaining a *non-asymptotic* convergence analysis is an important open problem of fundamental interest.

Table 1. When the problem is nonconvex and nonsmooth, finding a ϵ-stationary point is intractable, see Theorem 11. Thus we introduce a refined notion, (δ, ϵ)-stationarity, and provide non-asymptotic convergence rates for finding (δ, ϵ)-stationary point.

Deterministic rates	Convex	Nonconvex
L-smooth	$\mathcal{O}(\epsilon^{-0.5})$	$\mathcal{O}(\epsilon^{-2})$
L-Lipschitz	$\mathcal{O}(\epsilon^{-2})$	$\tilde{\mathcal{O}}(\epsilon^{-3}\delta^{-1})$

Stochastic rates	Convex	Nonconvex
L-smooth	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-4})$
L-Lipschitz	$\mathcal{O}(\epsilon^{-2})$	$\tilde{\mathcal{O}}(\epsilon^{-4}\delta^{-1})$

We tackle this problem for nonsmooth functions that are Lipschitz and directionally differentiable. This class is rich enough to cover common machine learning problems, including ReLU neural networks. Surprisingly, even for this seemingly restricted class, finding an ϵ-stationary point, i.e., a point $\bar{x}$ for which $d(0, \partial f(\bar{x})) \leq \epsilon$, is intractable. In other words, no algorithm can guarantee to find an ϵ-stationary point within a *finite* number of iterations.

This intractability suggests that, to obtain meaningful non-asymptotic results, we need to refine the notion of stationarity. We introduce such a notion and base our analysis on it, leading to the following main contributions of the paper:

- We show that a traditional ϵ-stationary point cannot be obtained in finite time (Theorem 5).
- We study the notion of (δ, ϵ)-stationary points (see Definition 4). For smooth functions, this notion reduces to usual ϵ-stationarity by setting $\delta = O(\epsilon/L)$. We provide a $\Omega(\delta^{-1})$ lower bound on the number of calls if algorithms are only allowed access to a generalized gradient oracle.
- We propose a normalized "gradient descent" style algorithm that achieves $\tilde{\mathcal{O}}(\epsilon^{-3}\delta^{-1})$ complexity in finding a (δ, ϵ)-stationary point in the deterministic setting.
- We propose a momentum based algorithm that achieves $\tilde{\mathcal{O}}(\epsilon^{-4}\delta^{-1})$ complexity in finding a (δ, ϵ)-stationary point in the stochastic finite variance setting.

[1]Massachusetts Institute of Technology. Correspondence to: Jingzhao Zhang <jzhzhang@mit.edu>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

As a proof of concept to validate our theoretical findings, we implement our stochastic algorithm and show that it matches the performance of empirically used SGD with momentum method for training ResNets on the Cifar10 dataset.

Our results attempt to bridge the gap from recent advances in developing a non-asymptotic theory for nonconvex optimization algorithms to settings that apply to training deep neural networks, where, due to non-differentiability of the activations, most existing theory does not directly apply.

1.1. Related Work

Asymptotic convergence for nonsmooth nonconvex functions. Benaïm et al. (2005) study the convergence of subgradient methods from a differential inclusion perspective; Majewski et al. (2018) extend the result to include proximal and implicit updates. Bolte & Pauwels (2019) focus on formally justifying the back propagation rule under nonsmooth conditions. In parallel, Davis et al. (2018) proved asymptotic convergence of subgradient methods assuming the objective function to be Whitney stratifiable. The class of Whitney stratifiable functions is broader than regular functions studied in (Majewski et al., 2018), and it does not assume the regularity inequality (see Lemma 6.3 and (51) in (Majewski et al., 2018)). Another line of work (Mifflin, 1977; Kiwiel, 2007; Burke et al., 2018) studies convergence of gradient sampling algorithms. These algorithms assume a deterministic generalized gradient oracle. Our methods draw intuition from these algorithms and their analysis, but are non-asymptotic in contrast.

Structured nonsmooth nonconvex problems. Another line of research in nonconvex optimization is to exploit structure: Duchi & Ruan (2018); Drusvyatskiy & Paquette (2019); Davis & Drusvyatskiy (2019) consider the composition structure $f \circ g$ of convex and smooth functions; Bolte et al. (2018); Zhang & He (2018); Beck & Hallak (2020) study composite objectives of the form $f + g$ where one function is differentiable or convex/concave. With such structure, one can apply proximal gradient algorithms if the proximal mapping can be efficiently evaluated. However, this usually requires weak convexity, i.e., adding a quadratic function makes the function convex, which is not satisfied by several simple functions, e.g., $-|x|$.

Stationary points under smoothness. When the objective function is smooth, SGD finds an ϵ-stationary point in $O(\epsilon^{-4})$ gradient calls (Ghadimi & Lan, 2013), which improves to $O(\epsilon^{-2})$ for convex problems. Fast upper bounds under a variety of settings (deterministic, finite-sum, stochastic) are studied in (Carmon et al., 2018; Fang et al., 2018; Zhou et al., 2018; Nguyen et al., 2019; Allen-Zhu, 2018; Reddi et al., 2016). More recently, lower bounds have also been developed (Carmon et al., 2017; Drori & Shamir, 2019; Arjevani et al., 2019; Foster et al., 2019). When the function enjoys high-order smoothness, a stronger goal is to find an approximate second-order stationary point and could thus escape saddle points too. Many methods focus on this goal (Ge et al., 2015; Agarwal et al., 2017; Jin et al., 2017; Daneshmand et al., 2018; Fang et al., 2019).

2. Preliminaries

In this section, we set up the notion of generalized directional derivatives that will play a central role in our analysis. Throughout the paper, we assume that the nonsmooth function f is L-Lipschitz continuous (more precise assumptions on the function class are outlined in §2.3).

2.1. Generalized gradients

We start with the definition of generalized gradients, following (Clarke, 1990), for which we first need:

Definition 1. Given a point $x \in \mathbb{R}^d$, and direction d, the ***generalized directional derivative*** of f is defined as

$$f^\circ(x; d) := \limsup_{y \to x, t \downarrow 0} \tfrac{f(y+td)-f(y)}{t}.$$

Definition 2. The ***generalized gradient*** of f is defined as

$$\partial f(x) := \{g \mid \langle g, d\rangle \le f^\circ(x, d),\ \forall d \in \mathbb{R}^d\}.$$

We recall below the following basic properties of the generalized gradient, see e.g., (Clarke, 1990) for details.

Proposition 1 (Properties of generalized gradients)**.**

1. $\partial f(x)$ is a nonempty, convex compact set. For all vectors $g \in \partial f(x)$, we have $\|g\| \le L$.
2. $f^\circ(x; d) = \max\{\langle g, d\rangle \mid g \in \partial f(x)\}$.
3. $\partial f(x)$ is an upper-semicontinuous set valued map.
4. f is differentiable almost everywhere (as it is L-Lipschitz); let $\mathrm{conv}(\cdot)$ denote the convex hull, then
$$\partial f(x) = \mathrm{conv}\big(\{g | g = \lim_{k\to\infty} \nabla f(x_k),\ x_k \to x\}\big).$$
5. Let B denote the unit Euclidean ball. Then,
$$\partial f(x) = \cap_{\delta>0} \cup_{y \in x+\delta B} \partial f(y).$$
6. For any y, z, there exists $\lambda \in (0, 1)$ and $g \in \partial f(\lambda y + (1-\lambda)z)$ such that $f(y) - f(z) = \langle g, y - z\rangle$.

2.2. Directional derivatives

Since general nonsmooth functions can have arbitrarily large variations in their "gradients," we must restrict the function class to be able to develop a meaningful complexity theory. We show below that directionally differentiable functions match this purpose well.

Definition 3. A function f is called ***directionally differentiable in the sense of Hadamard*** (*cf.* (Sova, 1964; Shapiro, 1990)) if for any mapping $\varphi : \mathbb{R}_+ \to X$ for which $\varphi(0) = x$ and $\lim_{t\to 0^+} \frac{\varphi(t)-\varphi(0)}{t} = d$, the following limit exists:

$$f'(x; d) = \lim_{t\to 0^+} \tfrac{1}{t}(f(\varphi(t)) - f(x)). \qquad (1)$$

In the rest of the paper, we will say a function f is **directionally differentiable** if it is directionally differentiable in the sense of Hadamard at all x.

This directional differentiabilility is also referred to as Hadamard semidifferentiability in (Delfour, 2019). Notably, such directional differentiability is satisfied by most problems of interest in machine learning. It includes functions such as $f(x) = -|x|$ that *do not* satisfy the so-called regularity inequality (equation (51) in (Majewski et al., 2018)). Moreover, it covers the class of semialgebraic functions, as well as o-minimally definable functions (see Lemma 6.1 in (Coste, 2000)) discussed in (Davis et al., 2018). Currently, we are unaware whether the notion of Whitney stratifiability (studied in some recent works on nonsmooth optimization) implies directional differentiability.

A very important property of directional differentiability is that it is preserved under composition.

Lemma 2 (Chain rule)**.** *Let ϕ be Hadamard directionally differentiable at x, and ψ be Hadamard directionally differentiable at $\phi(x)$. Then the composite mapping $\psi \circ \phi$ is Hadamard directionally differentiable at x and*

$$(\psi \circ \phi)'_x = \psi'_{\phi(x)} \circ \phi'_x.$$

A proof of this lemma can be found in (Shapiro, 1990, Proposition 3.6). As a consequence, any neural network function composed of directionally differentiable functions, including ReLU/LeakyReLU, is directionally differentiable.

Directional differentiability also implies key properties useful in the analysis of nonsmooth problems. In particular, it enables the use of (Lebesgue) path integrals as follows.

Lemma 3. *Given any x, y, let $\gamma(t) = x+t(y-x)$, $t \in [0, 1]$. If f is directionally differentiable and Lipschitz, then*

$$f(y) - f(x) = \int_{[0,1]} f'(\gamma(t); y - x)dt.$$

The following important lemma further connects directional derivatives with generalized gradients.

Lemma 4. *Assume that the directional derivative exists. For any x, d, there exists $g \in \partial f(x)$ s.t. $\langle g, d\rangle = f'(x; d)$.*

2.3. Nonsmooth function class of interest

Throughout the paper, we focus on the set of Lipschitz, directionally differentiable and bounded (below) functions:

$$\begin{aligned}\mathcal{F}(\Delta, L) := \{f | &f \text{ is } L\text{-Lipschitz};\\ &f \text{ is directionally differentiable};\\ &f(x_0) - \inf_x f(x) \le \Delta\},\end{aligned} \qquad (2)$$

where a function $f : \mathbb{R}^n \to \mathbb{R}$ is $L-$Lipschitz if

$$|f(x) - f(y)| \le L\|x - y\|, \forall\, x, y \in \mathbb{R}^n.$$

As indicated previously, ReLU neural networks with bounded weight norms are included in this function class.

3. Stationary points and oracles

We now formally define our notion of stationarity and discuss the intractability of the standard notion. Afterwards, we formalize the optimization oracles and define measures of complexity for algorithms that use these oracles.

3.1. Stationary points

With the generalized gradient in hand, commonly a point is called stationary if $0 \in \partial f(x)$ (Clarke, 1990). A natural question is, what is the necessary complexity to obtain an *ϵ-stationary point*, i.e., a point x for which

$$\min\{\|g\| \mid g \in \partial f(x)\} \le \epsilon.$$

It turns out that attaining such a point is intractable. In particular, there is no finite time algorithm that can guarantee ϵ-stationarity in the nonconvex nonsmooth setting. We make this claim precise in our first main result.

Theorem 5. *Given any algorithm $\mathcal{A}$ that accesses function value and generalized gradient of f in each iteration, for any $\epsilon \in [0, 1)$ and for any finite iteration T, there exists $f \in \mathcal{F}(\Delta, L)$ such that the sequence $\{x_t\}_{t\in[1,T]}$ generated by $\mathcal{A}$ on the objective f does not contain any ϵ-stationary point with probability more than $\frac{1}{2}$.*

A key ingredient of the proof is that an algorithm $\mathcal{A}$ is uniquely determined by $\{f(x_t), \partial f(x_t)\}_{t\in[1,T]}$, the function values and gradients at the query points. For any two functions f_1 and f_2 that have the same function values and gradients at the same set of queried points $\{x_1, ..., x_t\}$, the distribution of the iterate x_{t+1} generated by $\mathcal{A}$ is identical for f_1 and f_2. However, due to the richness of the class of nonsmooth functions, we can find f_1 and f_2 such that the set of ϵ-stationary points of f_1 and f_2 are disjoint. Therefore, the algorithm cannot find a stationary point with probability more than $\frac{1}{2}$ for both f_1 and f_2 simultaneously. Intuitively, such functions exist because a nonsmooth function could vary arbitrarily—e.g., a nonsmooth nonconvex function could have constant gradient norms except at the (local) extrema, as happens for a piecewise linear zigzag function. Moreover, the set of extrema could be of measure zero. Therefore, unless the algorithm lands exactly in this measure-zero set, it cannot find any ϵ-stationary point.

Theorem 5 suggests the need for rethinking the definition of stationary points. Intuitively, even though we are unable to find an ϵ-stationary point, one could hope to find a point that is close to an ϵ-stationary point. This motivates us to adopt the following more refined notion:

Definition 4. A point x is called (δ, ϵ)-*stationary* if

$$d(0, \partial f(x + \delta B)) \leq \epsilon,$$

where $\partial f(x + \delta B) := \text{conv}(\cup_{y \in x + \delta B} \partial f(y))$ is the Goldstein δ-subdifferential, introduced in (Goldstein, 1977).

Note that if we can find a point y at most distance δ away from x such that y is ϵ-stationary, then we know x is (δ, ϵ)-stationary. However, the contrary is not true. In fact, (Shamir, 2020) shows that finding a point that is δ close to an $\epsilon-$stationary point requires exponential dependence on the dimension of the problem.

At first glance, Definition 4 appears to be a weaker notion since if x is ϵ-stationary, then it is also a (δ, ϵ)-stationary point for any $\delta \geq 0$, but not vice versa. We show that the converse implication indeed holds, assuming smoothness.

Proposition 6. The following statements hold:

(i) ϵ-stationarity implies (δ, ϵ)-stationarity for any $\delta \geq 0$.

(ii) If f is smooth with an L-Lipschitz gradient and if x is $(\frac{\epsilon}{3L}, \frac{\epsilon}{3})$-stationary, then x is also ϵ-stationary, i.e.

$$d\left(0, \partial f\left(x + \tfrac{\epsilon}{3L} B\right)\right) \leq \tfrac{\epsilon}{3} \implies \|\nabla f(x)\| \leq \epsilon.$$

Consequently, the two notions of stationarity are equivalent for differentiable functions. It is then natural to ask: *does* (δ, ϵ)*-stationarity permit a finite time analysis?*

The answer is positive, as we will show later, revealing an intrinsic difference between the two notions of stationarity. Besides providing algorithms, in Theorem 11 we also prove an $\Omega(\delta^{-1})$ lower bound on the dependency of δ for algorithms that can only access a generalized gradient oracle.

We also note that (δ, ϵ)-stationarity behaves well as $\delta \downarrow 0$.

Lemma 7. *The set* $\partial f(x + \delta B)$ *converges as* $\delta \downarrow 0$ *as*

$$\lim_{\delta \downarrow 0} \partial f(x + \delta B) = \partial f(x).$$

Lemma 7 enables a straightforward routine for transforming non-asymptotic analyses for finding (δ, ϵ)-stationary points to asymptotic results for finding ϵ-stationary points. Indeed, assume that a finite time algorithm for finding (δ, ϵ)-stationary points is provided. Then, by repeating the algorithm with decreasing δ_k, (e.g., $\delta_k = 1/k$), any accumulation points of the repeated algorithm is an ϵ-stationary point with high probability.

3.2. Gradient Oracles

We assume that our algorithm has access to a generalized gradient oracle in the following manner:

Assumption 1. Given x, d, the oracle $\mathbb{O}(x, d)$ returns a function value f_x, and a generalized gradient g_x,

$$(f_x, g_x) = \mathbb{O}(x, d),$$

such that

(a) In the **deterministic** setting, the oracle returns

$$f_x = f(x),\ g_x \in \partial f(x) \text{ satisfying } \langle g_x, d \rangle = f'(x, d).$$

(b) In the **stochastic finite-variance** setting, the oracle only returns a stochastic gradient g with $\mathbb{E}[g] = g_x$, where $g_x \in \partial f(x)$ satisfies $\langle g_x, d \rangle = f'(x, d)$. Moreover, the variance $\mathbb{E}[\|g - g_x\|^2] \leq \sigma^2$ is bounded. In particular, no function value is accessible.

We remark that one cannot generally evaluate the generalized gradient ∂f in practice at any point where f is not differentiable. When the function f is not directionally differentiable, one needs to incorporate gradient sampling to estimate ∂f (Burke et al., 2002). Our oracle queries only an element of the generalized gradient and is thus **weaker** than querying the entire set ∂f. Still, finding a vector g_x such that $\langle g_x, d \rangle$ equals the directional derivative $f'(x, d)$ is non-trivial in general. Yet, when the objective function is a composition of directionally differentiable functions, such as ReLU neural networks, and if a closed form directional derivative is available for each function in the composition, then we can find the desired g_x by appealing to the chain rule in Lemma 2. This property justifies our choice of oracles.

3.3. Algorithm class and complexity measures

An algorithm A maps a function $f \in \mathcal{F}(\Delta, L)$ to a sequence of points $\{x_k\}_{k \geq 0}$ in $\mathbb{R}^n$. We denote $A^{(k)}$ to be the mapping from previous k iterations to x_{k+1}. Each x_k can potentially be a random variable, due to the stochastic oracles or algorithm design. Let $\{\mathcal{F}_k\}_{k \geq 0}$ be the filtration generated by $\{x_k\}$ such that x_k is adapted to $\mathcal{F}_k$. Based on the definition of the oracle, we assume that the iterates follow the structure

$$x_{k+1} = A^{(k)}(x_1, g_1, f_1, x_2, g_2, f_2, ..., x_k, g_k, f_k), \quad (3)$$

where $(f_k, g_k) = \mathbb{O}(y_k, d_k)$, and the point y_k and direction d_k are (stochastic) functions of the iterates $x_1, \ldots, x_k$.

For a random process $\{x_k\}_{k\in\mathbb{N}}$, we define the complexity of $\{x_k\}_{k\in\mathbb{N}}$ for a function f as the value

$$T_{\delta,\epsilon}(\{x_t\}_{t\in\mathbb{N}}, f) := \inf\big\{t\in\mathbb{N} \mid \text{Prob}\{d(0,\partial f(x+\delta B))\geq \epsilon \text{ for all } k\leq t\}\leq \tfrac{1}{3}\big\}. \quad (4)$$

Let $A[f, x_0]$ denote the sequence of points generated by algorithm A for function f. Then, we define the iteration complexity of an algorithm class $\mathcal{A}$ on a function class $\mathcal{F}$ as

$$\mathcal{N}(\mathcal{A},\mathcal{F},\epsilon,\delta) := \inf_{A\in\mathcal{A}}\sup_{f\in\mathcal{F}} T_{\delta,\epsilon}(A[f,x_0], f). \quad (5)$$

At a high level, (5) is the minimum number of oracle calls required for a fixed algorithm to find a (δ,ϵ)-stationary point with probability at least $2/3$ for all functions is class $\mathcal{F}$.

4. Deterministic Setting

For optimizing L-smooth functions, a crucial inequality is

$$f\big(x-\tfrac{1}{L}\nabla f(x)\big) - f(x) \leq -\tfrac{1}{2L}\|\nabla f(x)\|^2. \quad (6)$$

In other words, either the gradient is small or the function value decreases sufficiently along the negative gradient. However, when the objective function is nonsmooth, this descent property is no longer satisfied. Thus, defining an appropriate descent direction is non-trivial. Our key innovation is to solve this problem via randomization.

More specifically, in our algorithm, Interpolated Normalized Gradient Descent (INGD), we derive a local search strategy to find the descent direction at an iterate x_t. The vector $m_{t,k}$ plays the role of descent direction and we sequentially update it until the condition

$$f(x_{t,k}) - f(x_t) < -\frac{\delta\|m_{t,k}\|}{4}, \qquad \text{(descent condition)}$$

is satisfied. To connect with the descent property (6), observe that when f is smooth, with $m_{t,k} = \nabla f(x_t)$ and $\delta = \|m_{t,k}\|/L$, (descent condition) is the same as (6) up to a factor 2. This connection motivates our choice of descent condition.

When the descent condition is satisfied, the next iterate x_{t+1} is obtained by taking a normalized step from x_t along the direction $m_{t,k}$. Otherwise, we stay at x_t and continue the search for a descent direction. We raise special attention to the fact that inside the k-loop, the iterates $x_{t,k}$ are always obtained by taking a normalized step from x_t. Thus, all the inner iterates $x_{t,k}$ have distance exactly δ from x_t.

To update the descent direction, we incorporate a randomized strategy. We randomly sample an interpolation point $y_{t,k+1}$ on the segment $[x_t, x_{t,k}]$ and evaluate the generalized gradient $g_{t,k+1}$ at this random point $y_{t,k+1}$. Then, we update the descent direction as a convex combination of $g_{t,k+1}$ and the previous direction $m_{t,k}$. Due to lack of smoothness, the violation of the descent condition does not directly imply that $g_{t,k+1}$ is small. Instead, the projection of the generalized gradient is small along the direction $m_{t,k}$ on average. Hence, with a proper linear combination, the random interpolation allows us to guarantee the decrease of $\|m_{t,k}\|$ in expectation. This reasoning allows us to derive the non-asymptotic convergence rate in high probability.

Algorithm 1 Interpolated Normalized Gradient Descent

```
1:  Initialize $x_1 \in \mathbb{R}^d$
2:  for $t = 1, 2, ..., T$ do
3:    while $\|m_{t,K}\| > \epsilon$ do
4:      Call oracle $\sim, m_{t,1} = \mathbb{O}(x_t, \vec{0})$
5:      for $k = 1, ..., K$ do
6:        $x_{t,k} = x_t - \delta\frac{m_{t,k}}{\|m_{t,k}\|}$
7:        if $\|m_{t,k}\| \leq \epsilon$ then
8:          Terminate the algorithm and return $x_t$
9:        else if $f(x_{t,k}) - f(x_t) < -\frac{\delta\|m_{t,k}\|}{4}$ then
10:         Break while-loop
11:         Set $x_{t+1} = x_{t,k}$ and $t \leftarrow t+1$
12:       else
13:         Sample $y_{t,k+1}$ uniformly from $[x_t, x_{t,k}]$
14:         Call oracle $\sim, g_{t,k+1} = \mathbb{O}(y_{t,k+1}, -m_{t,k})$
15:         Update $m_{t,k+1} = \beta_{t,k}m_{t,k} + (1-\beta_{t,k})g_{t,k+1}$
            with $\beta_{t,k} = \frac{4L^2-\|m_{t,k}\|^2}{4L^2+2\|m_{t,k}\|^2}$
16:       end if
17:     end for
18:   end while
19: end for
20: Return $x_t$ such that $\|m_{t,K}\| \leq \epsilon$
```

Theorem 8. *In the deterministic setting and with Assumption 1(a), the* INGD *algorithm with parameters $K = \frac{48L^2}{\epsilon^2}$ and $T = \frac{4\Delta}{\epsilon\delta}$ finds a (δ,ϵ)-stationary point for function class $\mathcal{F}(\Delta, L)$ with probability $1-\gamma$ using at most*

$$\frac{192\Delta L^2}{\epsilon^3\delta}\log\left(\frac{4\Delta}{\gamma\delta\epsilon}\right) \quad \textit{oracle calls.}$$

Since we introduce random sampling for choosing the interpolation point, even in the deterministic setting we can only guarantee a high probability result. The detailed proof is deferred to Appendix C.

A sketch of the proof is as follows. Since $\|x_{t,k} - x_t\| = \delta$ for any k, the interpolation point $y_{t,k}$ is inside the ball $x_t + \delta B$. Hence $m_{t,k} \in \partial f(x_t + \delta B)$ for any k. In other words, as soon as $\|m_{t,k}\| \leq \epsilon$ (line 7), the reference point x_t is (δ,ϵ)-stationary. If this is not true, i.e., $\|m_{t,k}\| > \epsilon$, then we check whether (descent condition) holds, in which case

$$f(x_{t,k}) - f(x_t) < -\frac{\delta\|m_{t,k}\|}{4} < -\frac{\epsilon\delta}{4}.$$

Knowing that the function value is lower bounded, this can happen at most $T = \frac{4\Delta}{\epsilon\delta}$ times. Thus, for at least one x_t, the local search inside the while loop is not broken by the descent condition. Finally, given that $\|m_{t,k}\| > \epsilon$ and the descent condition is not satisfied, we show that

$$\mathbb{E}[\|m_{t,k+1}\|^2] \leq \left(1 - \frac{\mathbb{E}[\|m_{t,k}\|^2]}{3L^2}\right) \mathbb{E}[\|m_{t,k}\|^2]$$

This implies that $\mathbb{E}[\|m_{t,k}\|^2]$ follows a decrease of order $O(1/k)$. Hence with $K = O(1/\epsilon^2)$, we are guaranteed to find $\|m_{t,k}\| \leq \epsilon$ with high probability.

Remark 9. If the problem is smooth, the descent condition is always satisfied in one iteration. Hence the global complexity of our algorithm reduces to $T = O(1/\epsilon\delta)$. Due to the equivalence of the notions of stationarity (Prop. 6), with $\delta = O(\epsilon/L)$, our algorithm recovers the standard $O(1/\epsilon^2)$ convergence rate for finding an ϵ-stationary point. In other words, our algorithm can adapt to the smoothness condition.

5. Stochastic Setting

In the deterministic setting one of the key ingredients used INGD is to check whether the function value decreases sufficiently. However, evaluating the function value can be computationally expensive, or even infeasible in the stochastic setting. For example, when training neural networks, evaluating the entire loss function requires going through all the data, which is impractical. As a result, we do not assume access to function value in the stochastic setting and instead propose a variant of INGD that only relies on gradient information.

Algorithm 2 Stochastic INGD (x_1, p, q, β, T, K)

```
1: Initialize $x_1 \in \mathbb{R}^d$.
2: Call oracle $g(x_1) = \mathbb{O}(x_1, \vec{0})$ and set $m_1 = g(x_1)$.
3: for $t = 1, 2, ..., T$ do
4:     Update $x_{t+1} = x_t - \eta_t m_t$ with $\eta_t = \frac{1}{p\|m_t\|+q}$.
5:     Sample $y_{t+1}$ uniformly from $[x_t, x_{t+1}]$
6:     Call oracle $g(y_{t+1}) = \mathbb{O}(y_{t+1}, -m_t)$
7:     Update $m_{t+1} = \beta m_t + (1-\beta) g(y_{t+1})$
8: end for
9: Randomly sample $i$ uniformly from $\{1, ..., T\}$.
10: Update $i = \max\{i - K, 1\}$
11: Return $x_i$.
```

One of the challenges of using stochastic gradients is the noisiness of the gradient evaluation. To control the variance of the associated updates, we introduce a parameter q into the normalized step size:

$$\eta_t = \frac{1}{p\|m_t\| + q}.$$

A similar strategy is used in adaptive methods like (Duchi et al., 2011; Kingma & Ba, 2015) to prevent instability. Here, we show that the constant q allows us to control the variance of $x_{t+1} - x_t$. In particular, it implies the bound

$$\mathbb{E}[\|x_{t+1} - x_t\|^2] \leq \frac{G^2}{q},$$

where $G^2 := L^2 + \sigma^2$ is a trivial upper-bound on the expected norm of any sampled gradient g.

Another substantial change (relative to INGD) is the removal of the explicit local search, since the stopping criterion can now no longer be tested without access to the function value. Instead, one may view $x_{t-K+1}, \ldots, x_{t-1}, x_t$ as an implicit local search with respect to the reference point x_{t-K}. In particular, we show that when the direction m_t has a small norm, then x_{t-K} is a (δ, ϵ)-stationary point, but not x_t. This discrepancy explains why we output x_{t-K} instead of x_t.

In the deterministic setting, the direction $m_{t,k}$ inside each local search is guaranteed to belong to $\partial f(x_t + \delta B)$. Hence, controlling the norm of $m_{t,k}$ implies the (δ, ϵ)-stationarity of x_t. In the stochastic case, however, we have two complications. First, only the expectation of the gradient evaluation satisfies the membership $\mathbb{E}[g(y_k)] \in \partial f(y_k)$. Second, the direction m_t is a convex combination of all the previous gradients $g(y_1), \ldots, g(y_t)$, with all coefficients being nonzero. In contrast, we use a re-initialization in the deterministic setting. We overcome these difficulties and their ensuing subtleties to finally obtain the following complexity result:

Theorem 10. *In the stochastic setting, with Assumption 1(b), the Stochastic-INGD algorithm (Algorithm 2) with parameters $G = \sqrt{L^2 + \sigma^2}$, $\beta = 1 - \frac{\epsilon^2}{64G^2}$, $p = \frac{64G^2 \ln(16G/\epsilon)}{\delta\epsilon^2}$, $q = 4Gp$, $K = p\delta$, $T = \frac{2^{16}G^3\Delta \ln(16G/\epsilon)}{\epsilon^4\delta} \max\{1, \frac{G\delta}{8\Delta}\}$ ensures*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|m_t\|] \leq \frac{\epsilon}{4}.$$

In other words, the number of gradient calls to achieve a $(\delta, \epsilon)-$stationary point is upper bounded by $\tilde{O}\left(\frac{G^3\Delta}{\epsilon^4\delta}\right)$.

For readability, the constants in Theorem 10 have not been optimized. The high level idea of the proof is to relate $\mathbb{E}[\eta_t\|m_t\|^2]$ to the function value decrease $f(x_t) - f(x_{t+1})$, and then to perform a telescopic sum.

We would like to emphasize the use of the adaptive step size η_t and the momentum term m_{t+1}. These techniques arise naturally from our goal to find a (δ, ϵ)-stationary point. The step size η_t helps us ensure that the distance moved is at most $\frac{1}{p}$, and hence we are certain that adjacent iterates are close to each other. The momentum term m_t serves as a convex combination of generalized gradients, as postulated by Definition 4.

Further, even though the parameter K does not directly influence the updates of our algorithm, it plays an important role in understanding our algorithm. Indeed, we show that

$$d\left(\mathbb{E}[m_t|x_{t-K}], \partial f(x_{t-K} + \delta B)\right) \leq \frac{\epsilon}{16}.$$

In other words, the conditional expectation $\mathbb{E}[m_t|x_{t-K}]$ is approximately in the δ-subdifferential $\partial f(x_{t-K} + \delta B)$ at x_{t-K}. This relationship is non-trivial.

On one hand, by imposing $K \leq \delta p$, we ensure that $x_{t-K+1}, \ldots, x_t$ are inside the δ-ball of center x_{t-K}. On the other hand, we guarantee that the contribution of m_{t-K} to m_t is small, providing an appropriate upper bound on the coefficient β^K. These two requirements help balance the different parameters in our final choice. Details of the proof may be found in Appendix D.

Recall that we do not access the function value in this stochastic setting, which is a strength of the algorithm. In fact, we can show that our δ^{-1} dependence is tight, when the oracle has only access to generalized gradients.

Theorem 11 (Lower bound on δ dependence)**.** *Let $\mathcal{A}$ denote the class of algorithms defined in Section 3.2 and $\mathcal{F}(\Delta, L)$ denote the class of functions defined in Equation (2). Assume $\epsilon \in (0,1)$ and $L = 1$. Then the iteration complexity is lower bounded by $\frac{\Delta}{8\delta}$ if the algorithm **only** has access to generalized gradients.*

The proof is inspired by Theorem 1.1.2 in (Nesterov, 2018). We show that unless more than $\frac{\Delta}{8\delta}$ different points are queried, we can construct two different functions in the function class that have gradient norm 1 at all the queried points, and the stationary points of both functions are $\Omega(\delta)$ away. For more details, see Appendix E.

This theorem also implies the negative result for finite time analyses that we showed in Theorem 5. Indeed, when an algorithm finds an ϵ-stationary point, the point is also a (δ, ϵ)-stationary for any $\delta > 0$. Thus, the iteration complexity must be at least $\lim_{\delta \to 0} \frac{\Delta}{8\delta} = +\infty$, i.e., no finite time algorithm can guarantee to find an ϵ-stationary point.

Before moving on to the experimental section, we would like to make several comments related to different settings. First, since the stochastic setting is strictly stronger than the deterministic setting, the stochastic variant Stochastic-INGD is applicable to the deterministic setting too. Moreover, the analysis can be extended to $q = 0$, which leads to a complexity of $\mathcal{O}(1/\epsilon^3\delta)$. This is the same as the deterministic algorithm. However, the stochastic variant does not adapt to the smoothness condition. In other words, even if the function is differentiable, we will not obtain a faster convergence rate. In particular, if the function is smooth, by using the equivalence of the types of stationary points, Stochastic-INGD finds an ϵ-stationary point in $\mathcal{O}(1/\epsilon^5)$ while standard SGD enjoys a $\mathcal{O}(1/\epsilon^4)$ convergence rate. We do not know whether a better convergence result is achievable, as our lower bound does not provide an explicit dependency on ϵ; we leave this as a future research direction.

6. Experiments

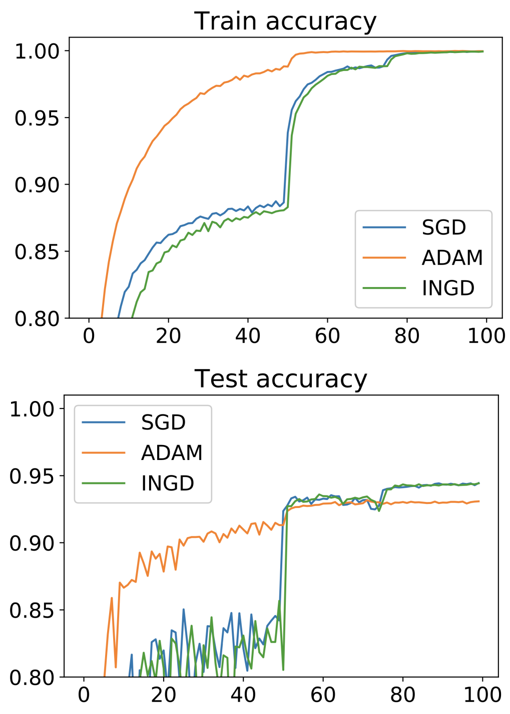

Figure 1. Learning curve of SGD, ADAM and INGD on training ResNet 20 on CIFAR10.

In this section, we evaluate the performance of our proposed algorithm Stochastic INGD on image classification tasks.

We train the ResNet20 (He et al., 2016) model on the CIFAR10 (Krizhevsky & Hinton, 2009) classification dataset. The dataset contains 50k training images and 10k test images in 10 classes.

We implement Stochastic INGD in PyTorch with the inbuilt auto differentiation algorithm (Paszke et al., 2017). We remark that except on the kink points, the auto differentiation matches the generalized gradient oracle, which justifies our choice. We benchmark the experiments with two popular machine learning optimizers, SGD with momentum and ADAM (Kingma & Ba, 2015). We train the model for 100 epochs with the standard hyper-parameters from the Github repository[1]:

- For SGD with momentum, we initialize the learning rate as 0.1, momentum as 0.9 and reduce the learning rate by

[1]https://github.com/kuangliu/pytorch-cifar

10 at epoch 50 and 75. The weight decay parameter is set to $5 \cdot 10^{-4}$.

- For ADAM, we use constant the learning rate 10^{-3}, betas in $(0.9, 0.999)$, and weight decay parameter 10^{-6} and $\epsilon = 10^{-3}$ for the best performance.
- For Stochastic-INGD, we use $\beta = 0.9$, $p = 1$, $q = 10$, and weight decay parameter 5×10^{-4}.

The training and test accuracy for all three algorithms are plotted in Figure 1. We observe that Stochastic-INGD matches the SGD baseline and outperforms the ADAM algorithm in terms of test accuracy. The above results suggests that the experimental implications of our algorithm could be interesting, but we leave a more systematic study as future direction.

7. Conclusions and Future Directions

In this paper, we investigate the complexity of finding first order stationary points of nonconvex nondifferentiable functions. We focus in particular on Hadamard semi-differentiable functions, which we suspect is perhaps the most general class of functions for which the chain rule of calculus holds—see the monograph (Delfour, 2019). We further extend the standard definition of ϵ-stationary points for smooth functions into a new notion of (δ, ϵ)-stationary points. We justify our definition by showing that no algorithm can find a $(0, \epsilon)$ stationary point for any $\epsilon < 1$ in a finite number of iterations and conclude that a positive δ is necessary for a finite time analysis. Using the above definition and a more refined gradient oracle, we prove that the proposed algorithms find stationary points within $\mathcal{O}(\epsilon^{-3}\delta^{-1})$ iterations in the deterministic setting and with $\mathcal{O}(\epsilon^{-4}\delta^{-1})$ iterations in the stochastic setting.

Our results provide the first non-asymptotic analysis of nonconvex optimization algorithms in the general Lipschitz continuous setting. Yet, they also open further questions. The first question is whether the current dependence on ϵ in our complexity bound is optimal. A future research direction is to try to find provably faster algorithms or construct adversarial examples that close the gap between upper and lower bounds on ϵ. Second, the rate we obtain in the deterministic case requires function evaluations and is randomized, leading to high probability bounds. Can similar rates be obtained by an algorithm oblivious to the function value? Another possible direction would be to obtain a deterministic convergence result. More specialized questions include whether one can remove the logarithmic factors from our bounds. Aside from the above questions on the rate, we can take a step back and ask high-level questions. Are there better alternatives to the current definition of (δ, ϵ)-stationary points? One should also investigate whether everywhere directional differentiability is necessary.

In addition to the open problems listed above, our work uncovers another very interesting observation. In the standard stochastic, nonconvex, and smooth setting, stochastic gradient descent is known to be theoretically optimal (Arjevani et al., 2019), while widely used practical techniques such as momentum-based and adaptive step size methods usually lead to worse theoretical convergence rates. In our proposed setting, momentum and adaptivity naturally show up in algorithm design, and become necessary for the convergence analysis. Hence we believe that studying optimization under more relaxed assumptions may lead to theorems that can better bridge the widening theory-practice divide in optimization for training deep neural networks, and ultimately lead to better insights for practitioners.

8. Acknowledgement

The authors thank Ohad Shamir for helpful discussions, and for pointing out the difference between being (δ, ϵ) stationary and being δ close to an ϵ stationary point. This work was partially supported by the MIT-IBM Watson AI Lab. SS and JZ also acknowledge support from NSF CAREER grant Number 1846088.

References

Agarwal, N., Allen-Zhu, Z., Bullins, B., Hazan, E., and Ma, T. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing*, pp. 1195–1199. ACM, 2017.

Allen-Zhu, Z. How to make the gradients small stochastically: Even faster convex and nonconvex SGD. In *Advances in Neural Information Processing Systems*, pp. 1157–1167, 2018.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N., and Woodworth, B. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

Beck, A. and Hallak, N. On the convergence to stationary points of deterministic and randomized feasible descent directions methods. *SIAM Journal on Optimization*, 30 (1):56–79, 2020.

Benaïm, M., Hofbauer, J., and Sorin, S. Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 44(1):328–348, 2005.

Bolte, J. and Pauwels, E. Conservative set valued fields, automatic differentiation, stochastic gradient method and deep learning. *arXiv preprint arXiv:1909.10300*, 2019.

Bolte, J., Sabach, S., Teboulle, M., and Vaisbourd, Y. First order methods beyond convexity and Lipschitz gradient continuity with applications to quadratic inverse problems. *SIAM Journal on Optimization*, 28(3):2131–2151, 2018.

Burke, J. V., Lewis, A. S., and Overton, M. L. Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research*, 27(3):567–584, 2002.

Burke, J. V., Curtis, F. E., Lewis, A. S., Overton, M. L., and Simões, L. E. Gradient sampling methods for nonsmooth optimization. *arXiv preprint arXiv:1804.11003*, 2018.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Lower bounds for finding stationary points I. *Mathematical Programming*, pp. 1–50, 2017.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

Clarke, F. H. *Optimization and nonsmooth analysis*, volume 5. Siam, 1990.

Coste, M. *An introduction to o-minimal geometry*. Istituti editoriali e poligrafici internazionali Pisa, 2000.

Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. Escaping saddles with stochastic gradients. In *International Conference on Machine Learning*, pp. 1163–1172, 2018.

Davis, D. and Drusvyatskiy, D. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Davis, D., Drusvyatskiy, D., Kakade, S., and Lee, J. D. Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, pp. 1–36, 2018.

Delfour, M. C. Introduction to optimization and Hadamard semidifferential calculus, 2019.

Drori, Y. and Shamir, O. The complexity of finding stationary points with stochastic gradient descent. *arXiv preprint arXiv:1910.01845*, 2019.

Drusvyatskiy, D. and Paquette, C. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178(1-2):503–558, 2019.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

Duchi, J. C. and Ruan, F. Stochastic methods for composite and weakly convex optimization problems. *SIAM Journal on Optimization*, 28(4):3229–3259, 2018.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pp. 689–699, 2018.

Fang, C., Lin, Z., and Zhang, T. Sharp analysis for nonconvex sgd escaping from saddle points. In *Conference on Learning Theory*, 2019.

Foster, D., Sekhari, A., Shamir, O., Srebro, N., Sridharan, K., and Woodworth, B. The complexity of making the gradient small in stochastic convex optimization. *arXiv preprint arXiv:1902.04686*, 2019.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle pointsonline stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pp. 797–842, 2015.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Goldstein, A. Optimization of lipschitz continuous functions. *Mathematical Programming*, 13(1):14–22, 1977.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1724–1732. JMLR. org, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kiwiel, K. C. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Majewski, S., Miasojedow, B., and Moulines, E. Analysis of nonsmooth stochastic approximation: the differential inclusion approach. *arXiv preprint arXiv:1805.01916*, 2018.

Mifflin, R. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 2(2):191–207, 1977.

Nesterov, Y. *Lectures on convex optimization*, volume 137. Springer, 2018.

Nguyen, L. M., van Dijk, M., Phan, D. T., Nguyen, P. H., Weng, T.-W., and Kalagnanam, J. R. Optimal finite-sum smooth non-convex optimization with SARAH. *arXiv preprint arXiv:1901.07648*, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pp. 314–323, 2016.

Shamir, O. Can we find near-approximately-stationary points of nonsmooth nonconvex functions? *arXiv preprint arXiv:2002.11962*, 2020.

Shapiro, A. On concepts of directional differentiability. *Journal of optimization theory and applications*, 66(3): 477–487, 1990.

Sova, M. General theory of differentiation in linear topological spaces. *Czechoslovak Mathematical Journal*, 14: 485–508, 1964.

Zhang, S. and He, N. On the convergence rate of stochastic mirror descent for nonsmooth nonconvex optimization. *arXiv preprint arXiv:1806.04781*, 2018.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduction for nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 3925–3936. Curran Associates Inc., 2018.