Optimization Theory and Machine Learning Practice: Mind the Gap

by

Jingzhao Zhang

B.S., University of California, Berkeley(2016)

M.S. Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author .....

Department of Electrical Engineering and Computer Science

November 30, 2021

Certified by .....

Suvrit Sra

Associate Professor of Electrical Engineering and Computer Science

Thesis Supervisor

Certified by .....

Ali Jadbabaie

JR East Professor of Engineering

Department of Civil and Environmental Engineering

Institute for Data, Systems and Society

Thesis Supervisor

Accepted by .....

Leslie A. Kolodziejski

Professor of Electrical Engineering and Computer Science

Chair, Department Committee on Graduate Students

Optimization Theory and Machine Learning Practice: Mind the Gap

by

Jingzhao Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on November 30, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Machine learning is a technology developed for extracting predictive models from data so as to be able to generalize predictions to unobserved data. The process of selecting a good model based on a known dataset requires optimization. In particular, an optimization procedure generates a variable in a constraint set to minimize an objective. This process subsumes many machine learning pipelines including neural network training, which will be our main testing ground for theoretical analyses in this thesis.

Among different kinds of optimization algorithms, gradient methods have become the dominant algorithms in deep learning due to their scalability to high dimensions and their natural bound to backpropagation. However, despite the popularity of gradient-based algorithms, our understanding of such algorithms in a machine learning context from a theory perspective seems far from sufficient. On one hand, within the current theory framework, most upper and lower bounds are closed, and the theory problems seem solved. On the other hand, the theoretical analyses hardly generate empirically faster algorithms than those found by practitioners. In this thesis, we review the theoretical analyses of gradient methods, and point out the discrepancy between theory and practice. We then provide an explanation for why the mismatch happens and propose some initial solutions by developing theoretical analyses driven by empirical observations.

Thesis Supervisor: Suvrit Sra
Title: Associate Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Ali Jadbabaie
Title: JR East Professor of Engineering
Department of Civil and Environmental Engineering
Institute for Data, Systems and Society

Acknowledgments

After twenty years of schoolwork, I will now graduate and move on. In the past four and a half years, I have enjoyed my life at MIT. Life always has up and downs but when I look back now, I feel genuinely fortunate to have been a part of this community.

I want to thank my advisors, Suvrit and Ali. Over the past 4 years, Ali has almost always been available for weekly individual meetings. I didn't realize this until recently and found it unbelievable. My discussions with Ali connected me to the campus when MIT was closed and they accumulated into one project after another. Ali gives advice from the student's prospective and cares about the student's long term development. Suvrit always provides help when I ask for it, which sometimes made me feel sorry to have to bother him so much. Suvrit's critical thinking motivates the works in this thesis. His sense of peacefulness enabled me to pursue bold research ideas and to hold when no progress was made.

I also want to thank my thesis committee members, Professor Ozdalglar and Professor Shamir for their crucial feedbacks on this thesis.

I wish to thank my wonderful collaborators Aditya, Andreas, Aryan, Cesar, Haochuan, Hongyi, Hongzhou, Lu, Macheng, Praneeth, Sashank, Sanjiv, Seungyeon, Srinadh, Stefanie, Subhro, Tiancheng, Tianxing, Xiang and Yi, for their patience and dedication. Learning from their knowledge and insights were among my most treasured moments in my PhD study. Moreover, I wish to thank all my friends for their help and support in everyday life.

I hope to also thank fundings from the EECS LIM Graduate fellowship, the Darpa Lagrange program, the NSF CAREER grant IIS-1846088, the Scalable Nonconvex Optimization grant, the Vinton Hayes grant, the MIT-IBM Watson grant W1771646 and the NSF-Collaborative Research grant DMS-2022448.

I wish to thank my wife, my mom and my dad. With them, I fear not challenges.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Outline

In this thesis, we aim to identify, study and reduce the gap between optimization existing theory and machine learning practice. We start by first zooming out and thinking about the historical context of optimization. We hope to convince the readers that the mismatch between theory and experiments existed in various forms throughout the development of optimization and has motivated the discovery of beautiful ideas and results. Hence, this problem can lead to promising future work and practical impact. We will conclude the chapter by outlining the contents of this thesis.

Although the focus of this thesis is addressing the gap between *modern* optimization theory and machine learning practice, we would like to take a detour and look at the history of optimization. We will see through this history to provide a better context that the theory results were not done under a single complexity framework, but instead theory and practice are interleaved and can facilitate the development of one another. We believe that such a view may shed light on how we could reduce the mismatch between machine learning experiments and known theoretical results. A fraction of the information below is extracted from a talk given by Stephan Wright at OPTML seminar[1].

Optimization means finding a feasible variable to minimize an objective function. Therefore, in a more general sense, optimization subsume many canonical theoretical computer science problems, including shortest-path, max-flow, SAT, traveling sales-

[1] `https://www.youtube.com/watch?v=hZmHcalwJLA`

man problem, Hamiltonian path, etc. Hence, early optimization analysis stemmed from theoretical computer science. It studies the number of operations performed by a Turing machine and focuses on *bit complexity*.

Bit complexity studies the number of operations required to solve a problem that is encoded in an integer number of bits. Therefore, early in 1960s, one major question for the continuous optimization community was whether a linear program is solvable in number of operations that are polynomial in the linear program's definition. Hence researchers only aimed for linear (i.e., exponential) convergence rates of the objective suboptimality. At that time, simplex method [Dantzig, 1990] was widely adopted for its good empirical performance. However in 1972, Klee and Minty [1972] showed that simplex method is not a polynomial time algorithm. Soon, Karmarkar [1984] proposed the famous ellipsoid method and showed that linear programming can be solved in (pseudo)polynomial time.

At this time, the mismatch between theory and practice already happened. Simplex, despite being theoretically slow, was observed to be much faster than the ellipsoid method. Such a discrepancy motivated the design of interior point methods [Khachiyan, 1980]. These methods enjoy both good empirical performances and theoretical guarantees. Later, interior points method also became the major approach to solving constrained nonlinear problems.

The inconsistency between theory and real-world linear problems not only motivated the design of interior point methods, but also altered the framework used for analysis. As the original P vs NP complexity hierarchy is inherently connected to combinatorics, doing analysis under this framework on continuous problems incurs extra complications. Furthermore, researchers also found that an exponential time algorithm may not necessarily be slower than a polynomial time algorithm in practice. Consequently, a new framework known as *oracle complexity* [Nemirovskii et al., 1983] was proposed and sublinear (i.e., polynomial convergence) rates became acceptable.

The oracle complexity framework studies the number of oracle calls required to achieve a target convergence measure. Oracles are generally computation bottlenecks in optimization such as function value or gradient evaluations. Oracle complexity

abstracts away many details and allows researchers to focus on reducing the cost of the computationally expensive subprocedures. As oracle complexity also allows for elegant and rigorous theory, it became the mainstream framework for optimization theory and remains so to this day.

1.1 Optimization in the machine learning context

Though the analysis framework has not changed, optimization problems today are very different from those thirty years ago. With the rise of data science and machine learning, optimization algorithms become essential components of AI applications.

One important modern optimization problem is neural network training. Neural networks are the state of art models in many data-driven tasks, and have become one of the most important engines behind the rapid progress of machine learning. In particular, over past two decades, model performance has beaten human performance in many applications, including image classification, text understanding, speech recognition, games etc, and neural networks are an essential building block for all of these models.

The power of neural networks lies in their strong expressive power. In particular, with millions and billions of parameters, the neural network architecture can almost approximate any function encountered in practice when enough data point is given. However, the huge number of parameters also makes neural network training computationally expensive. The cost of training a state of the art model from scratch could already be several hundred million dollars in 2020. For this reason, researchers care much about finding an optimization algorithm that can accelerate training in a memory efficient manner.

Among different optimization algorithms, gradient based algorithms are the most widely adopted in deep learning applications for a few reasons. First, gradient methods scale to high dimensional problems better than than zeroth order methods that only use function value information. Second, unlike higher order methods, the memory complexity of gradient methods is also only linear in the number of parameters.

Furthermore, neural networks permits backpropagation, and can evaluate gradient efficiently. Therefore, over the past years, our understanding of gradient methods in neural network training has improved greatly on the experimental side. Techniques such as momentum, adaptive step sizes, normalization have been discovered to accelerate and stabilize neural network training.

Great progress was also made on the theory side. Many new gradient based algorithms were developed under the dominant framework and proven to be minimax optimal under both deterministic and stochastic settings. These algorithms utilize beautiful ideas such as variance reduction, Nesterov's acceleration, restart and local regularization. The convergence criteria studied involve function suboptimality, first-order stationarity and second-order stationarity. Aside from convergence rate, many analyses also focus on the study benign properties of neural network loss landscape. Since neural network training is inherently a nonconvex problem that in general can be NP-hard, much research effort provides explanation for why first order methods can find good solutions. Explanations include that all saddle points are strict saddle points, model being over-parametrized, etc.

However, despite all the progress in optimization research, the development of theory and practice seems to be proceeding in different directions. For example, theoretically optimal algorithms do not always yield good empirical performance. We will discuss this aspect in more detail in Chapter 2. On the other hand, many techniques are known to improve experimental convergence rates. Examples include batch normalization, heavy-ball style momentum and adaptive step sizes. However, though these techniques were discovered based on high level motivations, formal proofs of acceleration are in general still missing.

The goal of this thesis is to identify the theory-practice mismatch and to provide an explanation for why the mismatch happens in modern machine learning setups. After understanding why most of the theory results do not apply to neural network training, we will propose a fix that can unify theory and practice. We will provide initial analysis under a new framework and show how that can lead to better algorithm designs.

1.2 Outline

Closing the gap between theory and practice requires huge efforts way beyond the limit of one PhD program. However, we hope that in this thesis, we could provide a clear description of the problem. We will introduce ideas drawn from experiments and show how these empirical ideas can benefit theoretical analysis.

In particular, we will start in Chapter 2 by reviewing the current complexity analysis. After examining the formal definition of convergence rates, we will understand the conditions under which "optimal algorithms" remain optimal. We will then move forward to describe the mismatch between theory and practice. In particular, we will show that there is limited theory on why fast empirical algorithms are fast, whereas theoretically fast algorithms are usually slow in practice.

Once we understand the definitions of complexity and the meaning of "minimax optimal algorithms", we can move to investigate the cause of inconsistency between empirical observations and theoretical predictions. In particular,we provide four concrete examples in the next four chapters. They examine the smoothness, bounded variance, differentiability and optimality conditions in real world experiments and discuss how they can result in unexpected behaviors that defies known theory.

In Chapter 3, we start by analyzing a simple but popular technique used in practice known as gradient clipping. Gradient clipping is crucial in many deep learning tasks such as deep reinforcement learning and language modeling, and can speed up convergence several times. However, there is yet limited explanation for why clipping can be faster than vanilla gradient descent. In this chapter, we show that if the objective function is not globally smooth but has a more relaxed smoothness condition, then gradient clipping can be arbitrarily faster than vanilla gradient descent.

In Chapter 4, we provide empirical and theoretical evidence that a heavy-tailed distribution of the noise in stochastic gradients is one cause of SGD's poor performance. Instead, we provide the first tight upper and lower convergence bounds for adaptive gradient methods under heavy-tailed noise. Further, we demonstrate how gradient clipping plays a key role in addressing heavy-tailed gradient noise. Subse-

quently, we show how clipping can be applied in practice by developing an *adaptive* coordinate-wise clipping algorithm (ACClip) and demonstrate its better performance on large-scale tasks.

In Chapter 5, we focus on complexity analysis for nonsmooth nonconvex functions. In practice, neural networks with ReLU activations not only do not have global smoothness constants, but can also be nondifferentiable. Motivated by this, 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.

In Chapter 6, we examine the applicability of the existing theoretical analyses, and focus on a major disconnect between deep learning practice and optimization theory. Specifically, we provide numerical evidence that in large-scale neural network training (ImageNet + ResNet and WT103 + TransformerXL), the weight variables *do not* converge to stationary points where the gradient of the loss function vanishes. Remarkably, however, despite the oscillatory behavior of the variables due to non-stationarity, we observe that the loss function converges. Inspired by this observation, we propose a perspective based on the theory of dynamical systems to prove convergence to an invariant measure that explains this phenomenon. We further discuss how this perspective can better align the theory with empirical observations.

The above chapters provided a few concrete examples on how the gap between theory and practice can be identified, analyzed, and reduced. In Chapter 7, we discuss several implications of these ideas. We will finally conclude this thesis by elaborating on a few future directions.

Chapter 2

Convergence of optimization algorithms: Theory vs practice

In this chapter, we aim to provide a more detailed description of the mismatch between modern optimization theory and neural network training practices. To do so , we will start by introducing the oracle complexity framework. This framework was first systematically discussed by Nemirovski and Yudin [1983] and is still the dominant framework (see Nesterov [2013] for a textbook introduction). Under this framework, great progress was made in recent years. Optimal algorithms and convergence rates have been developed in most setups.

After presenting some of the recent theoretical progress, we will then move on to explain how these results conflicts with many interesting empirical observations. We will identify the mismatch between neural network training and the oracle complexity framework, and then suggest potential solutions and explanations.

2.1 A brief overview of complexity theory in optimization

In this section, we will go over the definition of oracle complexity in optimization. By doing so, we hope to identify the critical conditions under which the theoretical

convergence results hold for gradient based algorithms. A large part of this chapter is based on the textbook [Nesterov, 2013]. Some notations are from Carmon et al. [2017]. The following definitions are standard and can be found in any introductory text. We add them here for clarity of presentation.

We start by briefly reviewing the formulation of optimization problems. Let x be a vector in $\mathbb{R}^d$. Let $f : \mathbb{R}^d \to \mathbb{R}$ be the objective function. We would like to choose x in order to solve the following minimization problem,

$$\min \ f(x) \quad \text{s.t. } x \in \mathcal{S}, \tag{2.1}$$

where $\mathcal{S} \subseteq \mathbb{R}^d$ denotes the feasible set and encodes problem constraints. This problem formulation covers a large number of applications in machine learning and operations research. We will now define some properties of the objective that will be used later.

Definition 2.1.1. A set $\mathcal{S} \in \mathbb{R}^d$ is convex if for all $x, y \in \mathcal{S}, \lambda \in [0, 1]$,

$$\lambda x + (1 - \lambda)y \in \mathcal{S}.$$

The definition can be interpreted in a geometrical way that the line segment between any two points of $\mathcal{S}$ is contained in $\mathcal{S}$. Based on this notion, we can define the convexity of a real valued function.

Definition 2.1.2. A function $f : \mathcal{X} \to \mathbb{R}$ is convex if it has a convex epigraph,

$$\text{epi}(f) = \{(x, y) | x \in \mathcal{X}, y \geq f(x)\}.$$

For simplicity, we assume throughout this thesis that there is a global minimum x^*. Convexity also leads to a few equivalent definitions if $f(x)$ is continuously differentiable. Please see [Nesterov, 2013] for proofs.

Lemma 2.1.1. *A function continuously differentiable f is convex if and only if for all x, y in $\mathbb{R}^d$, and α in $[0, 1)$ one of the following holds,*

1. $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$,

2. $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$,

3. $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0$.

Based on the above definition, we can define μ-strong convexity to quantify the strength of convexity.

Definition 2.1.3. A continuously differentiable function f is μ-strongly convex if for all $x, y \in \mathbb{R}^d, \mu \geq 0$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|x - y\|^2.$$

To further ease analysis, we very often bound the variation of the function f by bounding the Lipschitz constant of its gradient.

Definition 2.1.4. A differentiable function $f(x)$ is L-smooth if for all x, y in $\mathbb{R}^d$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

With the above definitions, we could conveniently add assumptions to the objective function class of interest. We are now ready to define algorithms and complexity in the next subsection.

2.1.1 Gradient methods and oracle complexity

For ease of presentation and understanding, in this section we focus on complexity definition in the convex case. The complexity definitions in the nonconvex case are similar. We will discuss those counterparts later in this thesis.

Throughout this thesis, we will study the oracle complexity of **first order optimization algorithms**. In particular, a first order optimization algorithm has access to a gradient oracle that takes in a point x and returns the function value and the (potentially stochastic) gradient evaluated at x, $(f(x), \nabla f(x))$. An iterative algorithm generates a sequence of points $\{x_k\}_{k \geq 0}$ based on the feedback from oracle calls.

Following [Nesterov, 2013], we focus on iterative algorithms satisfying

$$x_k \in x_0 + \text{Lin}\{\nabla f(x_0), ..., \nabla f(x_{k-1})\}, \quad \text{for } k \geq 1, \tag{2.2}$$

where Lin denotes the linear span. We denote an algorithm by $\mathcal{A}$ which is determined by the mappings $\mathcal{A}_k$ from previous oracle calls to the next point in the sequence, i.e.

$$x_{k+1} = \mathcal{A}_k(x_0, \nabla f(x_0), ..., x_k, \nabla f(x_k)).$$

Now we are ready to define iteration complexity. Denote by $\mathcal{A}[f, x_0]$ the sequence of points $\{x_k\}$ generated by algorithm $\mathcal{A}$ with function f starting from x_0.

The complexity in the convex case is measured in terms of function suboptimality. Let the convergence rate for a particular objective f be defined as,

$$T_\epsilon(\{x_k\}_{k\in\mathbb{Z}^+}, f) = \inf\{N \in \mathbb{Z}^+ \mid f(x_N) - f(x^*) \leq \epsilon\}, \qquad \text{(deterministic)}$$
$$T_\epsilon(\{x_k\}_{k\in\mathbb{Z}^+}, f) = \inf\Big\{N \in \mathbb{Z}^+ | \text{Prob}(\ f(x_N) - f(x^*) \leq \epsilon) \geq \tfrac{1}{2}\Big\}. \qquad \text{(stochastic)}$$

Then the complexity of algorithm $\mathcal{A}$ on function class $\mathcal{F}$ is defined as the worst convergence rate

$$T(\mathcal{A}, \mathcal{F}) = \sup_{f\in\mathcal{F}} T_\epsilon(\mathcal{A}[f, x_0], f).$$

To provide a concrete example, we take a close look at the gradient descent algorithm. Let $\mathcal{F}_{\mu,L}$ denote the class of functions that are differentiable, L-smooth and $\mu-$strongly convex. Consider the following gradient descent update,

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x). \tag{2.3}$$

Then by smoothness and strong convexity (see Thm 2.1.15 in Nesterov [2013] for

details), one can show that

$$f(x_k) - f^* \leq \frac{L}{2}\left(\frac{L}{\mu + L}\right)^{2k} \|x_0 - x^*\|^2.$$

Therefore, the iteration complexity of gradient descent (2.3) for minimizing functions in the $\mu-$strongly convex and $L-$smooth function class $\mathcal{F}_{\mu,L}$ is of order $\mathcal{O}(\frac{L}{\mu}\log(\frac{L\|x_0-x^*\|}{\epsilon}))$.

The definitions and examples above provided upper bounds for convergence rates. Next, we discuss the minimax optimality of an algorithm. To do so, we need to understand lower bounds in addition to upper bounds.

2.1.2 Optimal gradient methods

As discussed in Section 2.1.1, we know that gradient descent (GD) has iteration complexity $\mathcal{O}(\frac{L}{\mu}\log(\frac{1}{\epsilon}))$. One natural question is whether this rate can be improved. To answer such question, we care about the lower bounds for the class of first order methods defined below

$$T_{\text{lower}}(\mathcal{F}) = \inf_{\mathcal{A}} \sup_{f\in\mathcal{F}} T_\epsilon(\mathcal{A}[f, x_0], f).$$

We say an algorithm $\mathcal{A}$ is minimax optimal if there exists a global constant C such that

$$T(\mathcal{A}, \mathcal{F}) \leq C T_{lower}(\mathcal{F}).$$

In other words, an algorithm is minimax optimal if it matches best worst case complexity up to a constant factor. To prove an algorithm is minimax optimal, one needs to first construct strong lower bounds. Nemirovski and Yudin [1983] proves a lower bound on oracle complexity for convex optimization problems.

Theorem 2.1.2 (Nemirovski and Yudin [1983]). *There exists a function $f \in \mathcal{F}_{\mu,L}$ such that no first order algorithm satisfying the linear span property can achieve faster rate than $\mathcal{O}(\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$.*

This shows that the gradient descent rate may not be optimal. Soon Nesterov closed the gap in Nesterov [1983] by proposing an accelerated algorithm now known as Nesterov's accelerated gradient descent (NAG).

$$\begin{aligned} y_{k+1} &= x_k - \alpha \nabla f(x_k) \\ x_{k+1} &= (1+\beta) y_{k+1} - \beta y_k. \end{aligned} \tag{2.4}$$

With this algorithm, Nesterov[Nesterov, 1983] proved the following result.

Theorem 2.1.3. *Let f be $\mu-$strongly convex and $L-$smooth, then NAG defined in* (2.4) *with parameter choice $\alpha = \frac{1}{L}, \beta = \frac{\sqrt{Q}-1}{\sqrt{Q}+1}$ satisfies*

$$f(y_k) - f(x^*) \leq \frac{\mu + L}{2} \|x_0 - x^*\|^2 \exp(-\tfrac{k-1}{\sqrt{Q}}).$$

This implies that the iteration complexity of NAG for $\mu-$strongly convex, $L-$smooth functions is $\mathcal{O}(\sqrt{\frac{L}{\mu}} \log(\frac{1}{\epsilon}))$.

With the above two theorems, we see that NAG is a minimax optimal algorithm for optimizing the $\mathcal{F}_{\mu,L}$ function class. In fact, NAG is also an optimal algorithm for the class of convex $L-$smooth functions. The rates are summarized in Table 2.1.

We have now defined oracle complexity and introduced the notion of optimal algorithms through Nesterov's accelerated gradient method. We notice that the oracle complexity framework is very powerful. With reasonable assumptions, this framework could motivate researchers to design elegant algorithms and close upper and lower bounds for convergence rates (i.e., the bounds are same up to constant scaling). However, we also realize that despite its intuitive definition, the optimality statement is critically conditioned on the assumptions on the oracle, function class, algorithm class and worst case performance. We will elaborate on these aspects in later chapters.

2.1.3 A short note on optimal algorithms in various cases

The recent literature of optimization upper and lower bounds is vast. Here we present a small subset that is more relevant to our arguments. We focus on common setups

	Gradient descent	Nesterov's method	Lower bound
Smooth & Convex	$\mathcal{O}(\frac{L}{\epsilon})$	$\mathcal{O}(\frac{L}{\sqrt{\epsilon}})$	$\mathcal{O}(\frac{L}{\sqrt{\epsilon}})$
Smooth & strongly convex	$\mathcal{O}(\frac{L}{\mu}\log(\frac{1}{\epsilon}))$	$\mathcal{O}(\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$	$\mathcal{O}(\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$

Table 2.1: Iteration complexities of GD and NAG with and without strong convexity.

where upper and lower bounds are closed. We do not intend to do a literature review but aim to provide a few resources for readers to start the exploration and to show that rates are closed for most common setups. All the cases below have closed bounds.

We first define two extensions beyond the deterministic gradient setup we described in the previous section. The first one is the **stochastic** setup. In this setup, an oracle instead of returning the exact gradient, it maps an variable x to a stochastic gradient $g(x)$ such that the stochastic gradient is unbiased with bounded variance.

$$\mathbb{E}[g(x)] = \nabla f(x),$$
$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq \sigma^2.$$

The second setup is known as the **finite sum** setup, where the objective function can be decomposed into n components,

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

Here, when one makes an oracle call, the oracle instead of giving a full gradient, can return the gradient evaluated for any single function $f_i(x)$. When each component function f_i corresponds to the loss from one data point, this problem becomes the empirical risk minimization problem from machine learning, and it has hence attracted great research attention. Consequently, many optimal algorithms are known.

The above two setups, along with the deterministic setup are the most standard setups for gradient oracle. We will discuss below that within these setups, optimal algorithms are known for most common function classes.

Convex and Lipschitz Functions In this case, the function is not necessarily differentiable and but its subgradient can always be evaluated. The optimal rates are well understood and achieved by standard subgradient descent algortihms. More details can be found in Nesterov's textbook [Nesterov, 2013]. Interestingly, the rates for both the deterministic case and the stochastic case are the same, except that the optimality measure is different. No better results are known in the finite sum setting beyond reducing the finite sum problem to stochastic or deterministic problems, as to exploit finite sum structure, all known algorithms require smoothness of component functions.

Convex and Smooth Optimal algorithms in the deterministic setting are discussed in the previous section. In the stochastic setup, when the objective function is strongly convex, optimal upper bounds can be found in [Ghadimi and Lan, 2012], whereas one short proof of lower bound can be found in [Agarwal et al., 2009]. When the function is not strongly convex, lower bound can be established by setting the strong convexity constant to be ϵ dependent. Upper bounds are achieved by SGD. In the finite sum setup, some of the methods that closed the rate are [Allen-Zhu, 2017, Lin et al., 2015]. The corresponding lower bounds can be found in [Agarwal and Bottou, 2015] for the strongly convex case, and in [Woodworth and Srebro, 2016] for the convex case.

Nonconvex smooth The gradient descent algorithm was shown to be optimal in [Carmon et al., 2017] for the deterministic case, whereas the SGD algorithm was shown to be optimal in [Arjevani et al., 2019]. When the Hessian is further Lipschitz, faster rates can be achieved by cubic-regularized Newton's method [Nesterov and Polyak, 2006], which was also proven to be optimal in [Carmon et al., 2017]. In the stochastic case, higher order smoothness does not seem to help. In the finite sum setting, tight upper and lower bounds can be found in [Fang et al., 2018a] and [Arjevani et al., 2019] respectively.

The above results conclude the standard convex and nonconvex setups. Other more specialized topics include min-max optimization problems, distributed/federated

learning problems, composite problems, second order stationarity, Geometric optimization etc. We refrain ourselves from further summaries, and will now get back to our main story line.

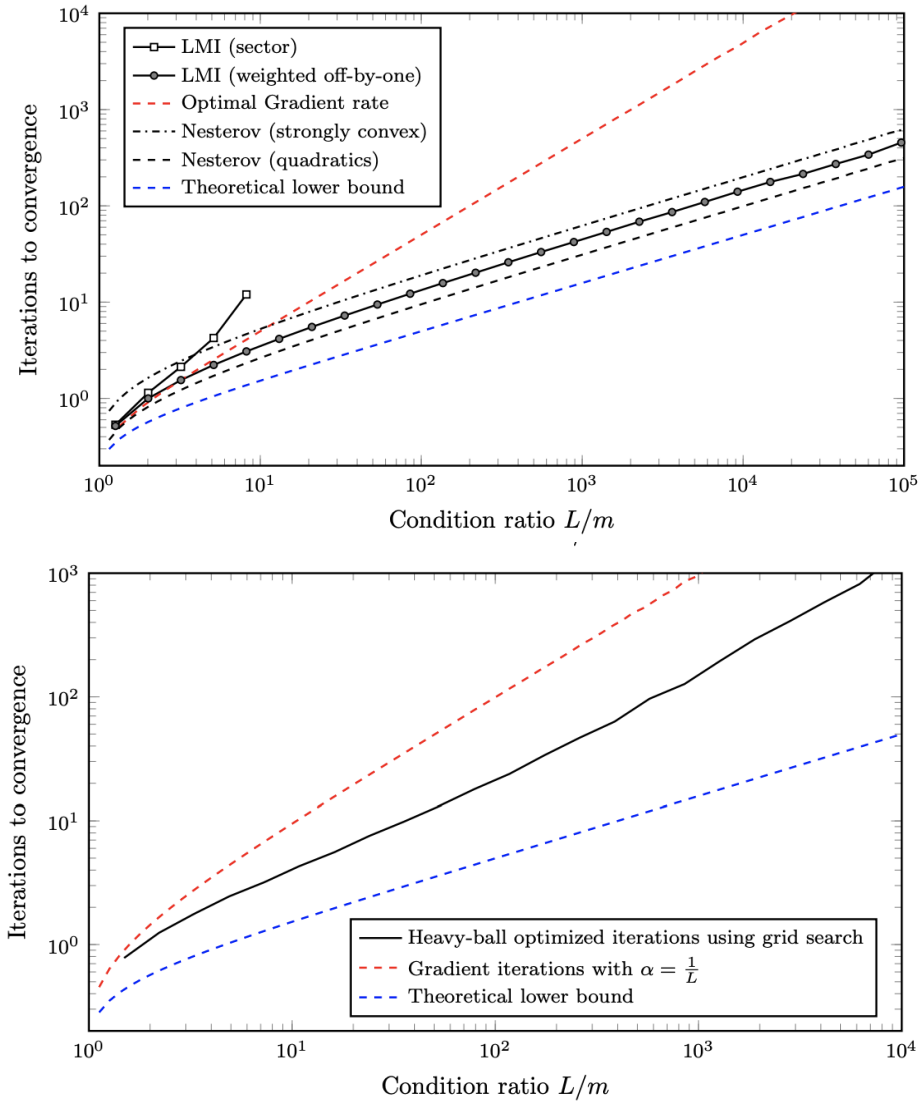


Figure 2-1: Numerically estimated convergence rate for Nesterov's method (upper row) and heavyball method (lower row) from [Lessard et al., 2016]. The black solid line denotes the numerical approach (known as LMI) used to estimate convergence rates. We can see that Nesterov's method has better convergence properties.

2.2 Theory vs neural network training

We have seen that significant progress had been made along the oracle complexity framework. New algorithms and rates were discovered. Minimax optimal rates are known in most common setups. These theoretical results are reasonably predictive of their performance in synthetic problems such as optimizing quadratic losses. However, up till now, optimal algorithms are not quite optimal in training neural networks.

Neural network training is naturally an optimization problem, and often entails solving the problem below:

$$\min_{\theta} f(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i) + \gamma(\theta),$$

where $\{x_1, x_2, \cdots, x_n\}$ denotes the dataset, θ denotes neural network parameters and γ denotes a regularization function such as L_2 norm regularization. It is worth noting that neural network training also cares about generalization performance in addition to optimization. However, generalization gap is beyond the scope of thesis. Admittedly, the interaction between optimization and generalization may be one source of the gap between the theory and practice. We leave that as a future direction and instead focus on optimization for now, which alone already has many mysteries.

Due to the natural optimization formulation of neural network training and the availability of gradient evaluations through back-propagation, one might expect that the powerful theorems for gradient based optimization algorithms can help improve the practice. However, as we will see theory results do not match with neural network experiments. We give a few concrete examples below.

2.2.1 Theory in practice

In this section, we show how some positive theory results fail to improve neural network practice. A great number of optimal algorithms utilizes two ideas: Nesterov's momentum and variance reduction. Nesterov's momentum can improve upon gradient descent for convex problems. Variance reduction can speed up SGD for solving finite-

sum problems. We will discuss the performance of these two ideas in practice.

As we have shown, Nesterov's momentum accelerated gradient descent in multiple settings. The convergence rate of heavyball momentum, however, is only known to accelerate quadratic problems. In other cases, the theoretical rates do not even match that of gradient descent [Ghadimi et al., 2014, Yang et al., 2016]. There are also evidence that heavyball may not be stable for general strongly convex functions beyond quadratics (e.g. see Figure 2-1)

However, despite this gap between Nesterov's momentum and heavyball momentum in theory, there is limited difference in their empirical performances. Sutskever et al. [2013] compared the two on smaller models that show Nesterov's momentum is slightly better. A more modern work [Choi et al., 2019] showed that for larger scale experiments, the difference between Nesterov's momentum and heavyball momentum is indistinguishable.

A second example is variance reduction. Variance reduction techniques were motivated by finite sum problems that subsume empirical risk minimization problems. Variance reduction is specifically designed to reduce noise in stochastic gradients, and it has generated a vast body of literature. However, despite its theoretical success, it struggles to improve practical performance (see Figure 2-2). Defazio and Bottou [2018] presented a thorough set of experiments and explanations.

In summary, many optimal gradient based algorithms were designed, yet these algorithms did not become popular in neural network training. Instead, as we will see in the next section, the dominant algorithms are those motivated by practice and with suboptimal theoretical properties.

2.2.2 Practice in theory

Deep learning experiments have led theoretical understandings for long. Many empirically fast techniques still lack rigorous theoretical understandings. In particular, within the current theoretical framework, it is very challenging to rigorously justify the effectiveness of many heuristics such as batch normalization, adaptive step sizes, learning rate warm up, etc.

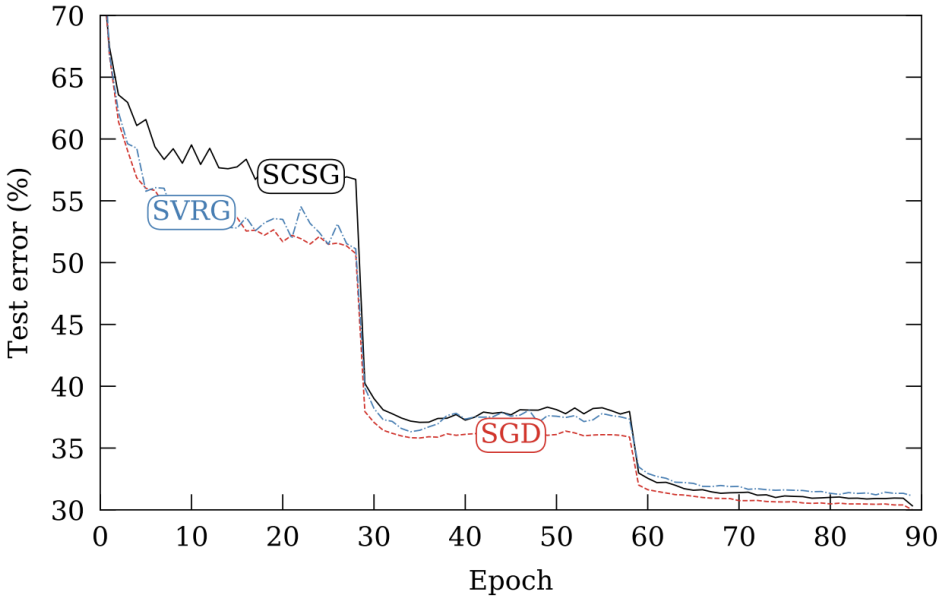


Figure 2-2: The figure from [Defazio and Bottou, 2018] showed that variance reduction methods fail to accelerate large scale neural network training despite great efforts.

We give two examples in this category. The first example is batch normalization. Batch normalization [Ioffe and Szegedy, 2015] was introduced to speed up neural network training. Its original motivation was to reduce covariance shift. Later, Santurkar et al. [2018] rebutted the intuition with numerical measurements and observed that batch normalization reduces the nonsmoothness of the loss landscape. Up to the author's knowledge, there is no follow up work that explains why batch norm can encourage smoothness, and how we could further accelerate training.

The second example is the ADAM optimizer. ADAM was proposed by Kingma and Ba [2014] and has been the most widely used optimizer since then. However, it was pointed later by Reddi et al. [2019] that the original ADAM proof was incorrect and ADAM can be divergent (see Figure 2-3). Yet, such finding does not stop researchers from using ADAM. Though later work was able to fix the proof of ADAM with different parameter choice and tricks, the rate has never been better than the vanilla SGD.

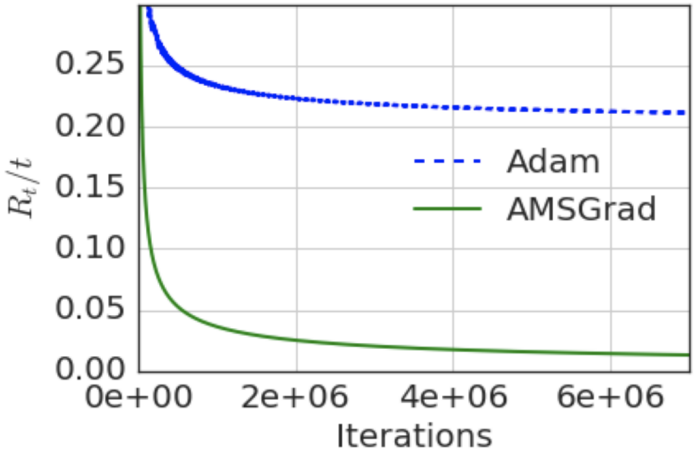


Figure 2-3: The figure from [Reddi et al., 2019] showed that the average regret of ADAM doesn't converge to zero even for convex losses.

2.2.3 Closing the gap

We have now seen the mismatch between theory and practice. In particular, we noticed that with the fast progress in both experimental and theoretical understandings of gradient methods, the gap between the two fields has become even larger. Understanding why this gap exists and how we could adjust theoretical analysis to reduce the gap will be the main focus in the rest of this thesis.

Before we dive into the concrete detailed discussions, we hope to provide a high level intuition for our approach to this problem. Ideally, mathematical derivations can never be wrong. The hint to why mismatch happens lies in the explicit or implicit assumptions made for defining oracle complexity. In the rest of this thesis, we will see how theoretical optimality critically depends on some assumptions that does not hold in practice. We will also propose new analysis and algorithms that help close the gap between theory and practice.

Chapter 3

Relaxed Smoothness and Gradient Clipping

In this chapter, we start a journey by examining one condition on which many theoretical analysis are based on: *global smoothness*. We want to understand how well this condition holds in neural network training and whether it can help explain the gap between theory and machine learning practice. Interestingly, we observe that gradient smoothness demonstrates significant variability along the training trajectory of deep neural networks. Further, this smoothness positively correlates with the gradient norm, and contrary to standard assumptions in the literature, it can grow with the norm of the gradient. These empirical observations limit the applicability of existing theoretical analyses of algorithms that rely on a fixed bound on smoothness. These observations motivate us to introduce a novel relaxation of gradient smoothness that is weaker than the commonly used Lipschitz smoothness assumption. Under the new condition, we prove that two popular methods, namely, *gradient clipping* and *normalized gradient*, converge arbitrarily faster than gradient descent with fixed stepsize. We further explain why such adaptively scaled gradient methods can accelerate empirical convergence and verify our results empirically in popular neural network training settings.

3.1 Introduction

We study optimization algorithms for neural network training and aim to resolve the mystery of why adaptive methods converge fast. Specifically, we study gradient-based methods for minimizing a differentiable nonconvex function $f : \mathbb{R}^d \to \mathbb{R}$, where $f(x)$ can potentially be stochastic, i.e., $f(x) = \mathbb{E}_\xi[F(x, \xi)]$. Such choices of f cover a wide range of problems in machine learning, and their study motivates a vast body of current optimization literature.

A widely used (and canonical) approach for minimizing f is the (stochastic) gradient descent (GD) algorithm. Despite its simple form, GD often achieves superior empirical [Wilson et al., 2017] performances and theoretical [Carmon et al., 2017] guarantees. However, in many tasks such as reinforcement learning and natural language processing (NLP), adaptive gradient methods (e.g., Adagrad [Duchi et al., 2011], ADAM [Kingma and Ba, 2014], and RMSProp [Tieleman and Hinton, 2012]) outperform SGD. Despite their superior empirical performance, our understanding of the fast convergence of adaptive methods is limited. Previous analysis has shown that adaptive methods are more robust to variation in hyper-parameters [Ward et al., 2018] and adapt to sparse gradients [Duchi et al., 2011]. However, in practice, the gradient updates are dense, and even after extensively tuning the SGD hyperparameters, it still converges much slower than adaptive methods in NLP tasks.

We analyze the convergence of clipped gradient descent and provide an explanation for its fast convergence. Even though gradient clipping is a standard practice in tasks such as language models [e.g. Merity et al., 2018, Gehring et al., 2017, Peters et al., 2018], it lacks a firm theoretical grounding. Goodfellow et al. [2016], Pascanu et al. [2013, 2012] discuss the gradient explosion problem in recurrent models and consider clipping as an intuitive workaround. We formalize this intuition and prove that clipped GD can *converge arbitrarily faster than fixed-step gradient descent.* This result is shown to hold under a novel smoothness condition that is *strictly weaker* than the standard Lipschitz-gradient assumption pervasive in the literature. Hence our analysis captures many functions that are not globally Lipschitz smooth. Importantly,

the proposed smoothness condition is derived on the basis of extensive NLP training experiments, which are precisely the same type of experiments for which adaptive gradient methods empirically perform superior to gradient methods.

By identifying a new smoothness condition through experiments and then using it to analyze the convergence of adaptively-scaled methods, we reduce the following gap between theory and practice. On one hand, powerful techniques such as Nesterov's momentum and variance reduction theoretically accelerate convex and nonconvex optimization. But, at least for now, they seem to have limited applicability in deep learning [Defazio and Bottou, 2018]. On the other hand, some widely used techniques (e.g., heavy-ball momentum, adaptivity) lack theoretical acceleration guarantees. We suspect that a major reason here is the misalignment of the theoretical assumptions with practice. Our work demonstrates that the concept of acceleration critically relies on the problem assumptions and that the standard global Lipschitz-gradient condition may not hold in the case of some applications and thus must be relaxed to admit a wider class of objective functions.

In light of the above background, we will later show in this section:

- Inspired and supported by neural network training experiments, we introduce a new smoothness condition that allows the local smoothness constant to increase with the gradient norm. This condition is *strictly weaker* than the pervasive Lipschitz-gradient assumption.
- We provide a convergence rate for clipped GD under our smoothness assumption (Theorem 3.4.1).
- We prove an upper-bound (Theorem 3.4.3) and a lower-bound (Theorem 3.4.2) on the convergence rate of GD under our relaxed smoothness assumption. The lower-bound demonstrates that GD with fixed step size can be *arbitrarily slower* than clipped GD.
- We provide upper bounds for stochastic clipped GD (Theorem 3.4.4) and SGD (Theorem 3.4.5). Again, stochastic clipped GD can be arbitrarily faster than SGD with a fixed step size.

We support our proposed theory with realistic neural network experiments. First, in the state of art LSTM language modeling (LM) setting, we observe the function smoothness has a strong correlation with gradient norm (see Figure 3-2). This aligns with the known fact that gradient clipping accelerates LM more effectively compared to computer vision (CV) tasks. Second, our experiments in CV and LM demonstrate that clipping accelerates training error convergence and allows the training trajectory to cross non-smooth regions of the loss landscape. Furthermore, gradient clipping can also achieve good generalization performance even in image classification (e.g., 95.2% test accuracy in 200 epochs for ResNet20 on Cifar10). Please see Section 3.5 for more details.

3.2 A More General Relaxed Smoothness Condition

In this section, we motivate and develop a relaxed smoothness condition that is weaker (and thus, more general) than the usual global Lipschitz smoothness assumption. We start with the traditional definition of smoothness.

3.2.1 Function smoothness (Lipschitz gradients)

Recall that f denotes the objective function that we want to minimize. We say that f is L-smooth if

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \text{for all } x, y \in \mathbb{R}^d. \tag{3.1}$$

For twice differentiable functions, condition (3.1) is equivalent to $\|\nabla^2 f(x)\| \leq L, \forall x \in \mathbb{R}^d$. This smoothness condition enables many important theoretical results. For example, Carmon et al. [2017] show that GD with $h = 1/L$ is up to a constant optimal for optimizing smooth nonconvex functions.

But the usual L-smoothness assumption (3.1) also has its limitations. Assuming existence of a global constant L that upper bounds the variation of the gradient is very restrictive. For example, simple polynomials such as $f(x) = x^3$ break the as-

sumption. One workaround is to assume that L exists in a compact region, and either prove that the iterates do not escape the region or run projection-based algorithms. However, such assumptions can make L very large and slow down the theoretical convergence rate. In Section 3.4, we will show that a slow rate is unavoidable for gradient descent with fixed step size, whereas clipped gradient descent can greatly improve the dependency on L.

The above limitations force fixed-step gradient descent (which is tailored to Lipschitz smooth functions) to converge slowly in many tasks. In Figure 3-1, we plot the estimated function smoothness at different iterations during training neural networks. We find that function smoothness varies greatly at different iterations. From Figure 3-1, we further find that local smoothness positively correlates with the full gradient norm, especially in the language modeling experiment. A natural question is:

> *Can we find a fine-grained smoothness condition under which we can design theoretically and empirically fast algorithms at the same time?*

To answer this question, we introduce a relaxed smoothness condition in the next section, which is developed on the basis of extensive experiments— Figure 3-1 provides an illustrative example.

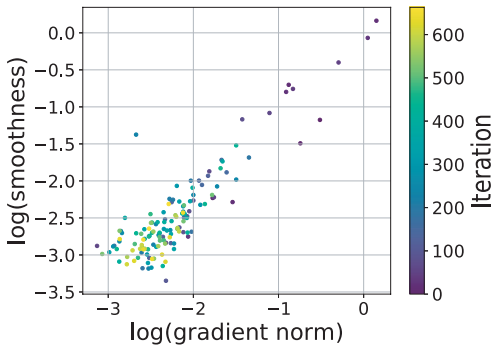


Figure 3-1: Gradient norm vs local gradient Lipschitz constant on a log-scale along the training trajectory for AWD-LSTM [Merity et al., 2017] on PTB dataset. The colorbar indicates the number of iterations during training. More experiments can be found in Section 3.5. Experiment details are in Appendix.

3.2.2 A new relaxed smoothness condition

We observe strong positive correlation between function smoothness and gradient norm in language modeling experiments (Figure 3-1(a)). This observation leads us to propose the following smoothness condition that allows local smoothness to grow with function gradients.

Definition 3.2.1. A second order differentiable function f is (L_0, L_1)-smooth if

$$\|\nabla^2 f(x)\| \leq L_0 + L_1 \|\nabla f(x)\|. \tag{3.2}$$

Definition 3.2.1 *strictly relaxes* the usual (and widely used) L-smoothness. There are two ways to interpret the relaxation: First, when we focus on a compact region, we can balance the constants L_0 and L_1 such that $L_0 \ll L$ while $L_1 \ll L$. Second, there exist functions that are (L_0, L_1)-smooth globally, but not L-smooth. Hence the constant L for L-smoothness gets larger as the compact set increases but L_0 and L_1 stay fixed. An example is given in Lemma 3.2.1.

Remark 3.2.1. It is worth noting that we do not need the Hessian operator norm and gradient norm to necessarily satisfy the linear relation (3.2). As long as these norms are positively correlated, gradient clipping can be shown to achieve faster rate than fixed step size gradient descent. We use the linear relationship (3.2) for simplicity of exposition.

Lemma 3.2.1. *Let f be the univariate polynomial $f(x) = \sum_{i=1}^{d} a_i x^i$. When $d \geq 3$, then f is (L_0, L_1)-smooth for some L_0 and L_1 but not L-smooth.*

Proof. The first claim follows from $\lim_{x\to\infty} \left|\frac{f'(x)}{f''(x)}\right| = \lim_{x\to-\infty} \left|\frac{f'(x)}{f''(x)}\right| = \infty$. The second claim follows by the unboundedness of $f''(x)$. □

3.2.3 Smoothness in neural networks

We saw that our smoothness condition relaxes the traditional smoothness assumption and is motivated empirically (Figure 3-1). Below we develop some intuition for this

phenomenon. We conjecture that the proposed positive correlation results from the common components in expressions of the gradient and the Hessian. We illustrate the reasoning behind this conjecture by considering an ℓ-layer linear network with quadratic loss—a similar computation also holds for nonlinear networks.

The L_2 regression loss of a deep linear network is $\mathcal{L}(Y, f(X)) := \|Y - W_\ell \cdots W_1 X\|^2$, where Y denotes labels, X denotes the input data matrix, and W_i denotes the weights in the i^{th} layer. By [Kawaguchi, 2016][Lemma 4.3], we know that

$$\nabla_{\text{vec}(w_i)} \mathcal{L}(Y, f(X)) = ((W_\ell \cdots W_{i+1}) \otimes (W_{i-1} \cdots W_2 W_1 X)^T)^T \,\text{vec}(f(X) - Y),$$

where $\text{vec}(\cdot)$ flattens a matrix in $\mathbb{R}^{m \times n}$ into a vector in $\mathbb{R}^{mn}$; $\otimes$ denotes the Kronecker product. For constants i, j such that $\ell \geq j > i > 0$, the second order derivative

$$\begin{aligned}\nabla_{\text{vec}(w_j)} \nabla_{\text{vec}(w_i)} \mathcal{L}(Y, f(X)) = \\ ((W_\ell \cdots W_{i+1}) \otimes (W_{i-1} \cdots W_2 W_1 X)^T)^T ((W_\ell \cdots W_{j+1}) \otimes (W_{j-1} \cdots W_2 W_1 X)^T) + \\ ((W_{j-1} \cdots W_{i+1}) \otimes (W_{i-1} \cdots W_2 W_1 X))(I \otimes ((f(X) - Y) W_\ell \cdots W_{j+1})).\end{aligned}$$

When $j = i$, the second term equals 0. Based on the above expressions, we notice that the gradient norm and Hessian norm may be positively correlated due to the following two observations. First, the gradient and the Hessian share many components such as the matrix product of weights across layers. Second, if one naively upper bounds the norm using Cauchy-Schwarz, then both upper-bounds would be monotonically increasing with respect to $\|W_i\|$ and $\|f(X) - Y\|$. Consequently, the gradient norm and the smoothness constant might be positively correlated for neural networks.

3.3 Problems setup and algorithms

In this section, we state the optimization problems and introduce gradient based algorithms for them that work under the new smoothness condition (3.2). Convergence analysis follows in Section 3.4.

Recall that we wish to solve the nonconvex optimization problem $\min_{x\in\mathbb{R}^d} f(x)$. Since in general this problem is intractable, following common practice we also seek an ϵ-stationary point, i.e., a point x such that $\|\nabla f(x)\| \le \epsilon$. Furthermore, we make the following assumptions to regularize the function class studied and subsequently provide nonasymptotic convergence rate analysis.

Assumption 3.3.1. The function f is lower bounded by $f^* > -\infty$.

Assumption 3.3.2. The function f is twice differentiable.

Assumption 3.3.3 (**(L_0, L_1)-smoothness).** The function f is (L_0, L_1)-smooth, i.e., there exist positive constants L_0 and L_1 such that $\|\nabla^2 f(x)\| \le L_0 + L_1\|\nabla f(x)\|$—see condition (3.2).

The first assumption is standard. Twice differentiability in Assumption 3.3.2 can relaxed to first-order differentiability by modifying the definition of (L_0, L_1)-smoothness as

$$\limsup_{\delta\to\vec{0}} \frac{\|\nabla f(x)-\nabla f(x+\delta)\|}{\|\delta\|} \le L_1\|\nabla f(x)\| + L_0.$$

The above inequality implies $\nabla f(x)$ is locally Lipschitz, and hence almost everywhere differentiable. Therefore, all our results can go through by handling the integrations more carefully. But to avoid complications and simplify exposition, we assume that the function is twice differentiable.

To further relax the global assumptions, by showing that GD and clipped GD are monotonically decreasing in function value, we require the above assumptions to hold just in a neighborhood determined by the sublevel set $\mathcal{S}$[1] for a given initialization x_0, where

$$\mathcal{S} := \{x \mid \exists\ y \text{ such that } f(y) \le f(x_0),\ \text{and } \|x - y\| \le 1\}. \tag{3.3}$$

[1]The constant "1" in the expression (3.3) is arbitrary and can be replaced by any fixed positive constant.

3.3.1 Gradient descent algorithms

In this section, we review a few well-known variants of gradient based algorithms that we analyze. We start with the ordinary *gradient descent* with a fixed step size η,

$$x_{k+1} = x_k - \eta \nabla f(x_k). \tag{3.4}$$

This algorithm (more precisely, its stochastic version) is widely used in neural network training. Many modifications of it have been proposed to stabilize or accelerate training. One such technique of particular importance is *clipped gradient descent*, which performs the following updates:

$$x_{k+1} = x_k - h_c \nabla f(x_k), \quad \text{where } h_c := \min\{\eta_c, \tfrac{\gamma \eta_c}{\|\nabla f(x)\|}\}. \tag{3.5}$$

Another algorithm that is less common in practice but has attracted theoretical interest is *normalized gradient descent*. The updates for normalized GD method can be written as

$$x_{k+1} = x_k - h_n \nabla f(x_k), \quad \text{where } h_n := \tfrac{\eta_n}{\|\nabla f(x)\| + \beta}. \tag{3.6}$$

The stochastic version of the above algorithms replaced the gradient with a stochastic estimator.

We note that Clipped GD and NGD are almost equivalent. Indeed, for any given η_n and β, if we set $\gamma \eta_c = \eta_n$ and $\eta_c = \eta_n / \beta$, then we have

$$\tfrac{1}{2} h_c \leq h_n \leq 2 h_c.$$

Therefore, clipped GD is equivalent to NGD up to a constant factor in the step size choice. Consequently, the nonconvex convergence rates in Section 3.4 and Section 3.4.2 for clipped GD also apply to NGD. We omit repeating the theorem statements and the analysis for conciseness.

3.4 Theoretical analysis

In this section, we analyze the oracle complexities of GD and clipped GD under our relaxed smoothness condition. All the proofs are in the Section 3.7.

Since we are analyzing the global iteration complexity, let us recall the formal definition being used. We follow the notation from [Carmon et al., 2017]. For a deterministic sequence $\{x_k\}_{k\in\mathbb{N}}$, define the complexity of $\{x_k\}_{k\in\mathbb{N}}$ for a function f as

$$T_\epsilon(\{x_t\}_{t\in\mathbb{N}}, f) := \inf\{t \in \mathbb{N} | \|\nabla f(x_t)\| \leq \epsilon\}. \tag{3.7}$$

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

$$T_\epsilon(\{x_t\}_{t\in\mathbb{N}}, f) := \inf\Big\{t \in \mathbb{N} | \text{Prob}(\|\nabla f(x_k)\| \geq \epsilon \text{ for all } k \leq t) \leq \tfrac{1}{2}\Big\}. \tag{3.8}$$

In particular, if the condition is never satisfied, then the complexity is ∞. Given an algorithm A_θ, where θ denotes hyperparameters such as step size and momentum coefficient, we denote $A_\theta[f, x_0]$ as the sequence of (potentially stochastic) iterates generated by A when operating on f with initialization x_0. Finally, we define the iteration complexity of an algorithm class parameterized by p hyperparameters, $\mathcal{A} = \{A_\theta\}_{\theta\in\mathbb{R}^p}$ on a function class $\mathcal{F}$ as

$$\mathcal{N}(\mathcal{A}, \mathcal{F}, \epsilon) := \inf_{A_\theta\in\mathcal{A}} \sup_{x_0\in\mathbb{R}^d, f\in\mathcal{F}} T_\epsilon(A_\theta[f, x_0], f). \tag{3.9}$$

The definition in the stochastic setting simply replaces the expression (3.7) with the expression (3.8). In the rest of the paper, "iteration complexity" refers to the quantity defined above.

3.4.1 Convergence in the deterministic setting

In this section, we present the convergence rates for GD and clipped GD under deterministic setting. We start by analyzing the **clipped GD** algorithm with update defined in equation (3.5).

Theorem 3.4.1. *Let $\mathcal{F}$ denote the class of functions that satisfy Assumptions 3.3.1, 3.3.2, and L-smoothness in set $\mathcal{S}$ defined in* (3.3)*. Recall f^* is a global lower bound for function value. With $\eta_c = \frac{1}{10L_0}, \gamma = \min\{\frac{1}{\eta_c}, \frac{1}{10L_1\eta_c}\}$, we can prove that the iteration complexity of clipped GD (Equation 3.5) is upper bounded by*

$$\frac{20L_0(f(x_0) - f^*)}{\epsilon^2} + \frac{20\max\{1, L_1^2\}(f(x_0) - f^*)}{L_0} \quad .$$

The proof of Theorem 3.4.1 is included in Appendix 3.7.1.

Now, we discuss the convergence of vanilla GD. The standard GD is known to converge to first order ϵ-stationary points in $\mathcal{O}((L(f(x_0) - f^*))\epsilon^{-2})$ iterations for $(L, 0)-$smooth nonconvex functions. By Theorem 1 of Carmon et al. [2017], this rate is up to a constant optimal.

However, we will show below that gradient descent is suboptimal under our relaxed (L_0, L_1)-smoothness condition. In particular, to prove the convergence rate for gradient descent with fixed step size, we need to permit it to benefit from an additional assumption on gradient norms.

Assumption 3.4.1. Given an initialization x_0, we assume that

$$M := \sup\{\|\nabla f(x)\| \mid x \text{ such that } f(x) \leq f(x_0)\} < \infty.$$

This assumption is in fact *necessary*, as our next theorem reveals.

Theorem 3.4.2. *Let $\mathcal{F}$ be the class of objectives satisfying Assumptions 3.3.1, 3.3.2, 3.3.3, and 3.4.1 with fixed constants $L_0 \geq 1$, $L_1 \geq 1$, $M > 1$. The iteration complexity for the fixed-step gradient descent algorithms parameterized by step size h is at least*

$$\frac{L_1 M(f(x_0) - f^* - 5\epsilon/8)}{8\epsilon^2(\log M + 1)}.$$

The proof can be found in Appendix 3.7.2.

Remark 3.4.1. Theorem 1 of Carmon et al. [2017] and Theorem 3.4.2 together show that gradient descent with a fixed step size cannot converge to an ϵ-stationary point

faster than $\Omega\left((L_1 M/\log(M) + L_0)(f(x_0) - f^*)\epsilon^{-2}\right)$. Recall that clipped GD algorithm converges as $\mathcal{O}\left(L_0(f(x_0) - f^*)\epsilon^{-2} + L_1^2(f(x_0) - f^*)L_0^{-1}\right)$. Therefore, clipped GD can be arbitrarily faster than GD when $L_1 M$ is large, or in other words, when the problem has a *poor initialization*.

Below, we provide an iteration upper bound for the fixed-step gradient descent update (3.4).

Theorem 3.4.3. *Suppose assumptions 3.3.1, 3.3.2, 3.3.3 and 3.4.1 hold in set $\mathcal{S}$ defined in* (3.3)*. If we pick parameters such that $h = \frac{1}{(2(ML_1+L_0))}$, then the iteration complexity of GD with a fixed step size defined in Algorithm 3.4 is upper bounded by*

$$4(ML_1 + L_0)(f(x_0) - f^*)\epsilon^{-2}.$$

Please refer to Appendix 3.7.3 for the proof. Theorem 3.4.3 shows that gradient descent with a fixed step size converges in $\mathcal{O}((ML_1 + L_0)(f(x_0) - f^*)/\epsilon^2)$ iterations. This suggests that the lower bound in Remark 3.4.1 is tight up to a log factor in M.

3.4.2 Convergence in the stochastic setting

In the stochastic setting, we assume GD and clipped GD have access to an unbiased stochastic gradient $\nabla\hat{f}(x)$ instead of the exact gradient $\nabla f(x)$. For simplicity, we denote $g_k = \nabla\hat{f}(x_k)$ below. To prove convergence, we need the following assumption.

Assumption 3.4.2. There exists $\tau > 0$, such that $\|\nabla\hat{f}(x) - \nabla f(x)\| \leq \tau$ almost surely.

Bounded noise can be relaxed to sub-gaussian noise if the noise is symmetric. Furthermore, up to our knowledge, this is the first stochastic nonconvex analysis of adaptive methods that does not require the gradient norm $\|\nabla f(x)\|$ to be bounded globally.

The main result of this section is the following convergence guarantee for stochastic clipped GD (based on the stochastic version of the update (3.5)).

Theorem 3.4.4. *Let Assumptions 3.3.1–3.3.3 and 3.4.2 hold globally with* $L_1 > 0$. *Let* $h = \min\{\frac{1}{16\eta L_1(\|g_k\|+\tau)}, \eta\}$ *where* $\eta = \min\{\frac{1}{20L_0}, \frac{1}{128L_1\tau}, \frac{1}{\sqrt{T}}\}$. *Then we can show that iteration complexity for stochastic clipped GD after update* (3.5) *is upper bounded by*

$$\Delta \max\left\{\frac{128L_1}{\epsilon}, \frac{4\Delta}{\epsilon^4}, \frac{80L_0 + 512L_1\tau}{\epsilon^2}\right\},$$

where $\Delta = (f(x_0) - f^* + (5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1)$.

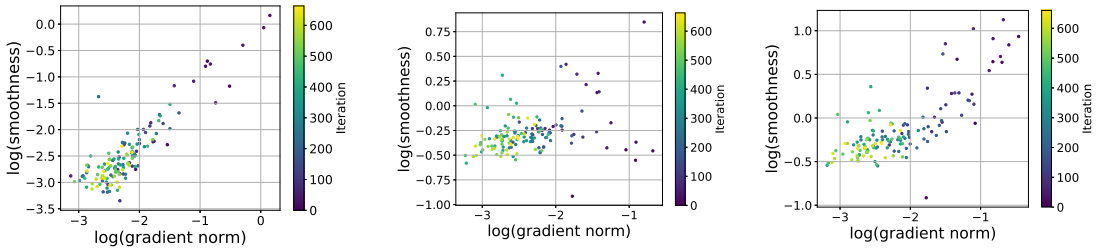
In comparison, we have the following upper bound for ordinary SGD.

Theorem 3.4.5. *Let Assumptions 3.3.1–3.3.3, and 3.4.2 hold globally with* $L_1 > 0$. *Let* $h = \min\{\frac{1}{\sqrt{T}}, \frac{1}{L_1(M+\tau)}\}$. *Then the iteration complexity for the stochastic version of GD* (3.4) *is upper bounded by*

$$\left(f(x_0) - f^* + (5L_0 + 4L_1M)(M+\tau)^2\right)^2\epsilon^{-4}.$$

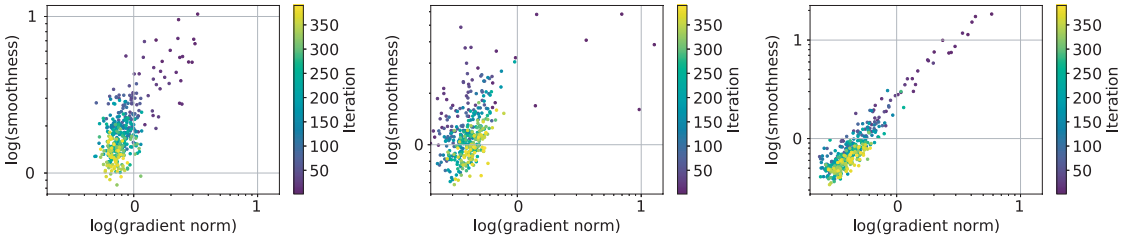
We do not know of a lower bound for this algorithm. However, the deterministic lower bound in Theorem 3.4.2 is still valid, though probably loose. Therefore, the convergence of SGD still requires additional assumption and can again **be arbitrarily slower** compared to clipped SGD when M is large.

3.5 Experiments



(a) Step size 30, with clipping. (b) Step zie 2, w/o clipping. (c) Step size 2, with clipping.

Figure 3-2: Gradient norm vs smoothness on log scale for LM training. The dot color indicates the iteration number. Darker ones correspond to earlier iterations. Note that the spans of x and y axis are not fixed.



(a) SGD with momentum. (b) Step size 1, w/o clipping. (c) Step size 5, with clipping.

Figure 3-3: Gradient norm vs smoothness on log scale for ResNet20 training. The dot color indicates the iteration number.

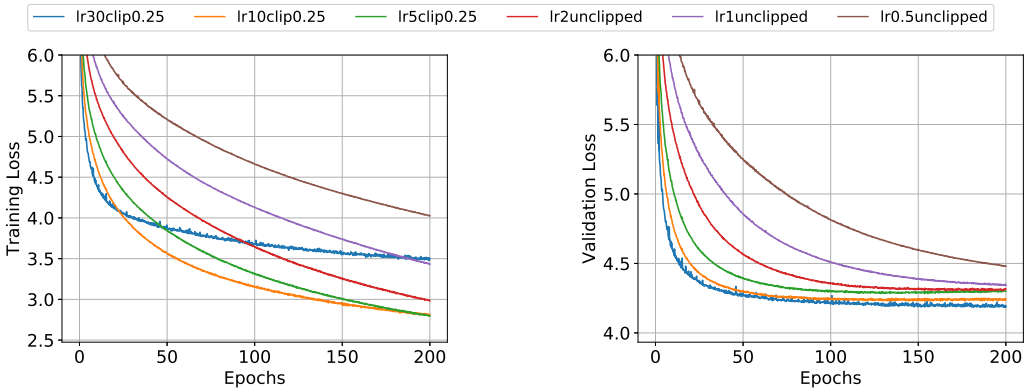
In this section, we summarize our empirical findings on the positive correlation between gradient norm and local smoothness. We then show that clipping accelerates convergence during neural network training. Our experiments are based on two tasks: language modeling and image classification. We run language modeling on the Penn Treebank (PTB) dataset with AWD-LSTM models [Merity et al., 2018][2]. We train ResNet20 [He et al., 2016] on the Cifar10 dataset [Krizhevsky and Hinton, 2009]. Details about the smoothness estimation and experimental setups are in Zhang et al. [2019a].

First, our experiments test whether the local smoothness constant increases with the gradient norm, as suggested by the relaxed smoothness conditions defined in (3.2) (Section 3.2). To do so, we evaluate both quantities at points generated by the optimization procedure. We then scatter the local smoothness constants against the gradient norms in Figure 3-2 and Figure 3-3. Note that the plots are on a log-scale.

We notice that the correlation exists in the default training procedure for language modeling (see Figure 3-2a) but not in the default training for image classification (see Figure 3-3a). This difference aligns with the fact that gradient clipping is widely used in language modeling but is less popular in ResNet training, *offering empirical support to our theoretical findings.*

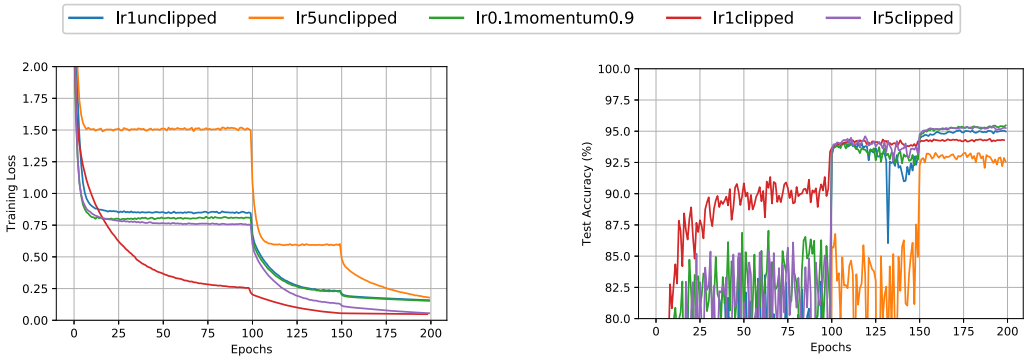
We further investigate the cause of correlation. The plots in Figures 3-2 and 3-3 show that correlation appears when the models are trained with clipped GD and large learning rates. We propose the following explanation. Clipping enables the training

[2]Part of the code is available at `https://github.com/JingzhaoZhang/why-clipping-accelerates`



(a) Training loss of LSTM with different optimization parameters.

(b) Validation loss of LSTM with different optimization parameters.



(c) Training loss of ResNet20 with different optimization parameters.

(d) Test accuracy of ResNet20 with different optimization parameters.

Figure 3-4: Training and validation loss obtained with different training methods for LSTM and ResNet training. The validation loss plots the cross entropy. The training loss additionally includes the weight regularization term. In the legend, 'lr30clip0.25' denotes that clipped SGD uses step size 30 and that the L_2 norm of the stochastic gradient is clipped by 0.25. In ResNet training, we threshold the stochastic gradient norm at 0.25 when clipping is applied.

trajectory to stably traverse non-smooth regions. Hence, we can observe that gradient norms and smoothness are positively correlated in Figures 3-2a and 3-3c. Without clipping, the optimizer has to adopt a small learning rate and stays in a region where local smoothness does not vary much, otherwise the sequence diverges, and a different learning rate is used. Therefore, in other plots of Figures 3-2 and 3-3, the correlation is much weaker.

As positive correlations are present in both language modeling and image classification experiments with large step sizes, our next set of experiments checks whether clipping helps accelerate convergence as predicted by our theory. From Figure 3-4,

we find that clipping indeed accelerates convergence. Because gradient clipping is a standard practice in language modeling, the LSTM models trained with clipping achieve the best validation performance and the fastest training loss convergence as expected. For image classification, surprisingly, clipped GD also achieves the fastest convergence and matches the test performance of SGD+momentum. These plots show that clipping can accelerate convergence and achieve good test performance at the same time.

3.6 Additional related work on accelerating gradient methods

Variance reduction. Many efforts have been made to accelerate gradient-based methods. One elegant approach is variance reduction e.g. [Schmidt et al., 2013, Johnson and Zhang, 2013, Defazio et al., 2014, Bach and Moulines, 2013, Xiao and Zhang, 2014, Gong and Ye, 2014, Fang et al., 2018a, Zhou et al., 2018]. This technique aims to solve stochastic and finite sum problems by averaging the noise in the stochastic oracle via utilizing the smoothness of the objectives.

Momentum methods. Another line of work focuses on achieving acceleration with momentum. Polyak [1964] showed that momentum can accelerate optimization for quadratic problems; later, Nesterov [1983] designed a variation that provably accelerate any smooth convex problems. Based on Nesterov's work, much theoretical progress was made to accelerate different variations of the original smooth convex problems e.g.[Ghadimi and Lan, 2016, 2012, Beck and Teboulle, 2009, Shalev-Shwartz and Zhang, 2014, Jin et al., 2017, Carmon et al., 2018, Lin et al., 2015, Nesterov, 2012].

Adaptive step sizes. The idea of varying step size in each iteration has long been studied. Armijo [1966] proposed the famous backtracking line search algorithm to choose step size dynamically. Polyak [1987] proposed a strategy to choose step size based on function suboptimality and gradient norm. More recently, Duchi et al. [2011] designed the Adagrad algorithm that can utilize the sparsity in stochastic gradients.

Since 2018, there has been a surge in studying the theoretical properties of adaptive gradient methods. One starting point is [Reddi et al., 2019], which pointed out that ADAM is not convergent and proposed the AMSGrad algorithm to fix the problem. Ward et al. [2018], Li and Orabona [2018] prove that Adagrad converges to stationary point for nonconvex stochastic problems. Zhou et al. [2018] generalized the result to a class of algorithms named Padam. Zou and Shen [2018], Staib et al. [2019], Chen et al. [2018], Zhou et al. [2018], Agarwal et al. [2018], Zhou et al. [2018], Zou and Shen [2018] also studied different interesting aspects of convergence of adaptive methods. In addition, Levy [2016] showed that normalized gradient descent may have better convergence rate in presence of injected noise. However, the rate comparison is under dimension dependent setting. Hazan et al. [2015] studied the convergence of normalized gradient descent for quasi-convex functions.

3.7 Proofs

3.7.1 Proof of Theorem 3.4.1

We start by proving a lemma that is repeatedly used in later proofs. The lemma bounds the gradient in a neighborhood of the current point by Grönwall's inequality (integral form).

Lemma 3.7.1. *Given x such that $f(x) \leq f(x_0)$, for any x^+ such that $\|x^+ - x\| \leq \min\{1/L_1, 1\}$, we have $\|\nabla f(x^+)\| \leq 4(L_0/L_1 + \|\nabla f(x)\|)$.*

Remark 3.7.1. Note that the constant "1" comes from the definition of $\mathcal{S}$ in (3.3). If Assumption 3.3.3 holds globally, then we do not need to constrain $\|x^+ - x\| \leq 1$. This version will be used in Theorem 3.4.4.

Proof. Let $\gamma(t)$ be a curve defined below,

$$\gamma(t) = t(x^+ - x) + x, \ t \in [0, 1].$$

Then we have

$$\nabla f(\gamma(t)) = \int_0^t \nabla^{(2)} f(\gamma(\tau))(x^+ - x)d\tau + \nabla f(\gamma(0)).$$

By Cauchy-Schwarz's inequality, we get

$$\begin{aligned}\|\nabla f(\gamma(t))\| &\leq \|x^+ - x\| \int_0^t \|\nabla^{(2)} f(\gamma(\tau))\|d\tau + \|\nabla f(x)\| \\ &\leq \frac{1}{L_1}\int_0^t (L_0 + L_1\|\nabla f(\gamma(\tau))\|)d\tau + \|\nabla f(x)\|.\end{aligned}$$

The second inequality follows by Assumption **??**. Then we can apply the integral form of Grönwall's inequality and get

$$\|\nabla f(\gamma(t))\| \leq \frac{L_0}{L_1} + \|\nabla f(x)\| + \int_0^t \left(\frac{L_0}{L_1} + \|\nabla f(x)\|\right)\exp(t-\tau)d\tau.$$

The Lemma follows by setting $t = 1$. □

Proof of the theorem 3.4.1

We parameterize the path between x_k and its updated iterate x_{k+1} as follows:

$$\gamma(t) = t(x_{k+1} - x_k) + x_k, \forall t \in [0, 1].$$

Since $x_{k+1} = x_k - h_k\nabla f(x_k)$, using Taylor's theorem, the triangle inequality, and Cauchy-Schwarz, we obtain

$$f(x_{k+1}) \leq f(x_k) - h_k\|\nabla f(x_k)\|^2 + \frac{\|x_{k+1} - x_k\|^2}{2}\int_0^1 \|\nabla^2 f(\gamma(t))\|dt.$$

Since

$$h_k \leq \frac{\gamma\eta}{\|\nabla f(x)\|} \leq \min\left\{\frac{1}{\|\nabla f(x)\|}, \frac{1}{L_1\|\nabla f(x_k)\|}\right\},$$

we know by Lemma 3.7.1

$$\|\nabla f(\gamma(t)\| \leq 4(\tfrac{L_0}{L_1} + \|\nabla f(x)\|).$$

Then by Assumption **??**, we obtain the "descent inequality":

$$f(x_{k+1}) \leq f(x_k) - h_k\|\nabla f(x_k)\|^2 + \frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\|\nabla f(x_k)\|^2 h_k^2.$$

Therefore, as long as $h_k \leq 1/(5L_0 + 4L_1\|\nabla f(x_k)\|)$ (which follows by our choice of η, γ), we can quantify the descent to be

$$f(x_{k+1}) \leq f(x_k) - \frac{h_k\|\nabla f(x_k)\|^2}{2}.$$

When $\|\nabla f(x_k)\| \geq L_0/L_1$, we have

$$\frac{h_k\|\nabla f(x_k)\|^2}{2} \geq \frac{L_0}{20\max\{1, L_1^2\}}.$$

When $\epsilon \leq \|\nabla f(x_k)\| \leq L_0/L_1$, we have

$$\frac{h_k\|\nabla f(x_k)\|^2}{2} \geq \frac{\|\nabla f(x_k)\|^2}{20L_0} \geq \frac{\epsilon^2}{20L_0}.$$

Therefore,

$$f(x_{k+1}) \leq f(x_k) - \min\left\{\frac{L_0}{20\max\{1, L_1^2\}}, \frac{\epsilon^2}{20L_0}\right\}.$$

Assume that $\epsilon \leq \|\nabla f(x_k)\|$ for $k \leq T$ iterations. By doing a telescopic sum, we get

$$\sum_{k=0}^{T-1} f(x_{k+1}) - f(x_k) \leq -T\min\left\{\frac{L_0}{20\max\{1, L_1^2\}}, \frac{\epsilon^2}{20L_0}\right\}.$$

Rearranging we get

$$T \leq \frac{20L_0(f(x_0) - f^*)}{\epsilon^2} + \frac{20\max\{1, L_1^2\}(f(x_0) - f^*)}{L_0}.$$

3.7.2 Proof of Theorem 3.4.2

We will prove a lower bound for the iteration complexity of GD with fixed step size. The high level idea is that if GD converges for all functions satisfying the assumptions, then the step size needs to be small. However, this small step size will lead to very slow convergence for another function.

Recall that the fixed step size GD algorithm is parameterized by the scalar: step size h. **First, we show that when $h > \frac{2\log(M)+2}{ML_1}$,**

$$\sup_{\substack{x_0 \in \mathbb{R}^d, \\ f \in \mathcal{F}}} T_\epsilon(A_h[f, x_0], f) = \infty$$

We start with a function that grows exponentially. Let $L_1 > 1, M > 1$ be fixed constants. Pick the initial point $x_0 = (\log(M) + 1)/L_1$. Let the objective be defined as follows,

$$f(x) = \begin{cases} \frac{e^{-L_1 x}}{L_1 e}, & \text{for } x < -\frac{1}{L_1}, \\ \frac{L_1 x^2}{2} + \frac{1}{2L_1}, & \text{for } x \in [-\frac{1}{L_1}, \frac{1}{L_1}], \\ \frac{e^{L_1 x}}{L_1 e}, & \text{for } x > \frac{1}{L_1}. \end{cases}$$

We notice that the function satisfies the assumptions with constants

$$L_0 = 1, \quad L_1 > 1, \quad M > 1. \tag{3.10}$$

When $h > 2x_0/M$, we would have $|x_1| > |x_0|$. By symmetry of the function and the super-linear growth of the gradient norm, we know that the iterates will diverge. Hence, in order for gradient descent with a fixed step size h to converge, h must be small enough. Formally,

$$h \leq \frac{2x_0}{M} = \frac{2\log(M) + 2}{ML_1}.$$

Second, we show that when $h \leq \frac{2\log(M)+2}{ML_1}$,

$$\sup_{\substack{x_0 \in \mathbb{R}^d, \\ f \in \mathcal{F}}} T_\epsilon(A_h[f, x_0], f) \geq \Delta L_1 M/(4\epsilon^2(\log M + 1))$$

Now, let's look at a different objective that grows slowly.

$$f(x) = \begin{cases} -2\epsilon(x+1) + \frac{5\epsilon}{4}, & \text{for } x < -1, \\ \frac{\epsilon}{4}(6x^2 - x^4), & \text{for } x \in [-1, 1], \\ 2\epsilon(x-1) + \frac{5\epsilon}{4}, & \text{for } x > 1. \end{cases}$$

This function is also second order differentiable and satisfies the assumptions with constants in (3.10). If we set $x_0 = 1 + \Delta/\epsilon$ for some constant $\Delta > 0$, we know that $f(x_0) - f^* = 2\Delta + 5\epsilon/4$. With the step size choice $h \leq (2\log M + 2)/(ML_1)$, we know that in each step, $x_{k+1} \geq x_k - (4\epsilon(\log M + 1))/(L_1 M)$. Therefore, for $k \leq \Delta L_1 M/(4\epsilon^2(\log M + 1))$,

$$\|\nabla f(x_k)\| = 2\epsilon.$$

After combining these two points, we proved the theorem by definition (5.5).

3.7.3 Proof of Theorem 3.4.3

We start by parametrizing the function value along the update,

$$f(\gamma(t)) := f(x_k - th\nabla f(x_k)), t \in [0, 1].$$

Note that with this parametrization, we have $\gamma(0) = x_k, \gamma(1) = x_{k+1}$. Now we would like to argue that if $f(x_k) \leq f(x_0)$, then $\|\nabla f(x(t))\| \leq M, \forall t \leq 1$. Assume by contradiction that this is not true. Then there exists $\epsilon > 0, t \in [0, 1]$ such that $\|\nabla f(x(t))\| \geq M + \epsilon$. Since ϵ can be made arbitrarily small below a threshold, we

assume $\epsilon < M$. Denote

$$t^* = \inf\{t \mid \|\nabla f(x(t))\| \geq M + \epsilon\}.$$

The value t^* exists by continuity of $\|\nabla f(x(t))\|$ as a function of t. Then we know by Assumption 3.4.1 that $f(x(t^*)) > f(x_k)$. However, by Taylor expansion, we know that

$$\begin{aligned} f(x(t^*)) &\leq f(x_k) - th\|\nabla f(x_k)\|^2 + (th)^2\|\nabla f(x_k)\|^2 \int_0^t \|\nabla^{(2)} f(x(\tau))\| d\tau \\ &\leq f(x_k) - th\|\nabla f(x_k)\|^2 + (th)^2\|\nabla f(x_k)\|^2 (L_1(M+\epsilon) + L_0) \\ &\leq f(x_k). \end{aligned}$$

The last inequality follows by $h = 1/(2(ML_1 + L_0))$. Hence we get a contradiction and conclude that for all $t \leq 1$, $\|\nabla f(x(t))\| \leq M$. Therefore, following the above inequality and Assumption **??**, we get

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - h\|\nabla f(x_k)\|^2 + h^2 \frac{L_1 M + L_0}{2} \|\nabla f(x_k)\|^2 \\ &\leq f(x_k) - \frac{\epsilon^2}{4(ML_1 + L_0)}. \end{aligned}$$

The conclusion follows by the same argument as in Theorem 3.4.1 via a telescopic sum over k.

3.7.4 Proof of Theorem 3.4.4

Recall that we set the following parameters

$$\begin{aligned} h_k &= \min\left\{\frac{1}{16\eta L_1(\|g_k\| + \tau)}, \eta\right\} \\ \eta &= \min\left\{\frac{1}{20L_0}, \frac{1}{128L_1\tau}, \frac{1}{\sqrt{T}}\right\} \end{aligned} \tag{3.11}$$

Similar to proof of Theorem 3.4.1, we have

$$\begin{aligned}\mathbb{E}[f(x_{k+1})|] \leq & f(x_k) - \mathbb{E}[h_k\langle g_k, \nabla f(x_k)\rangle] + \frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2\|g_k\|^2] \quad (3.12)\\ \leq & f(x_k) - \mathbb{E}[h_k\langle g_k, \nabla f(x_k)\rangle] + \frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2(\|\nabla f(x_k)\|^2\\ & + \|g_k - \nabla f(x_k)\|^2 + 2\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle)]\\ \leq & f(x_k) + \mathbb{E}[-h_k + \frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}h_k^2]\|\nabla f(x_k)\|^2\\ & + \mathbb{E}[h_k(-1 + (5L_0 + 4L_1\|\nabla f(x_k)\|)h_k)\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle]\\ & + \frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2(\|g_k - \nabla f(x_k)\|^2)].\end{aligned}$$

First we show $(5L_0+4L_1\|\nabla f(x_k)\|)h_k \leq \frac{1}{2}$. This follows by $5L_0h_k \leq \frac{1}{4}, h_k4L_1\|\nabla f(x_k)\| \leq h_k4L_1(\|g_k\| + \tau) \leq \frac{1}{4}$. Substitute in (3.14) and we get

$$\begin{aligned}\mathbb{E}[f(x_{k+1})|] \leq & f(x_k) + \mathbb{E}[-\frac{3h_k}{4}]\|\nabla f(x_k)\|^2 \quad (3.13)\\ & + \underbrace{\mathbb{E}[-h_k\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle]}_{T_1}\\ & + \underbrace{\mathbb{E}[(5L_0 + 4L_1\|\nabla f(x_k)\|)h_k^2\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle]}_{T_2}\\ & + \underbrace{\frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2(\|g_k - \nabla f(x_k)\|^2)]}_{T_3}.\end{aligned}$$

Then we bound T_1, T_2, T_3 in Lemma 3.7.2,3.7.3,3.7.4 and get

$$\begin{aligned}\mathbb{E}[f(x_{k+1})|] \leq & f(x_k) + \mathbb{E}[-\frac{h_k}{4}]\|\nabla f(x_k)\|^2 \quad (3.14)\\ & + (5L_0 + 2L_1\tau)\eta^2\tau^2 + 9\eta^2\tau L_0^2/L_1.\end{aligned}$$

Rearrange (3.14) and do a telescopic sum, we get

$$\begin{aligned}\mathbb{E}[\sum_{k\leq T}\frac{h_k}{4}\|\nabla f(x_k)\|^2] &\leq f(x_0) - f^* + \eta^2 T((5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1)\\ &\leq f(x_0) - f^* + ((5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1).\end{aligned}$$

Furthermore, we know

$$\begin{aligned} h_k\|\nabla f_k\|^2 &= \min\{\eta, \frac{1}{16L_1(\|\nabla f_k\|+\tau)}\}\|\nabla f_k\|^2 \\ &\geq \min\{\eta, \frac{1}{32L_1\|\nabla f_k\|}, \frac{1}{32L_1\tau}\}\|\nabla f_k\|^2 \\ &\geq \min\{\eta, \frac{1}{32L_1\|\nabla f_k\|}\}\|\nabla f_k\|^2. \end{aligned}$$

Hence along with $\eta \leq T^{-1/2}$, we get

$$\mathbb{E}\left[\sum_{k\leq T}\min\left\{\eta\|\nabla f_k\|^2, \frac{\|\nabla f_k\|}{32L_1}\right\}\right] \leq f(x_0) - f^* + ((5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1).$$

Let $\mathcal{U} = \{k|\eta\|\nabla f_k\|^2 \leq \frac{\|\nabla f_k\|}{16L_1}\}$, we know that

$$\mathbb{E}\left[\sum_{k\in\mathcal{U}}\eta\|\nabla f_k\|^2\right] \leq f(x_0) - f^* + ((5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1),$$

and

$$\mathbb{E}\left[\sum_{k\in\mathcal{U}^c}\frac{\|\nabla f_k\|}{32L_1}\right] \leq f(x_0) - f^* + ((5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1).$$

Therefore,

$$\begin{aligned} \mathbb{E}[\min_k\|\nabla f(x_k)\|] &\leq \mathbb{E}\left[\min\left\{\frac{1}{|\mathcal{U}|}\sum_{k\in\mathcal{U}}\|\nabla f_k\|, \frac{1}{|\mathcal{U}^c|}\sum_{k\in\mathcal{U}^c}\|\nabla f_k\|\right\}\right] \\ &\leq \mathbb{E}\left[\min\left\{\sqrt{\frac{1}{|\mathcal{U}|}\sum_{k\in\mathcal{U}}\|\nabla f_k\|^2}, \frac{1}{|\mathcal{U}^c|}\sum_{k\in\mathcal{U}^c}\|\nabla f_k\|\right\}\right] \\ &\leq \max\Bigg\{\sqrt{f(x_0) - f^* + ((5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1)\frac{\sqrt{T} + 20L_0 + 128L_1\tau}{T}}, \\ &\quad (f(x_0) - f^* + (5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1)\frac{64L_1}{T}\Bigg\}. \end{aligned}$$

The last inequality follow by the fact that either $|\mathcal{U}| \geq T/2$ or $|\mathcal{U}^c| \geq T/2$. This

implies that $\mathbb{E}[\min_{k\leq T} \|\nabla f(x_k)\|] \leq 2\epsilon$ when

$$T \geq \Delta \max\Big\{\frac{128L_1}{\epsilon}, \frac{4\Delta}{\epsilon^4}, \frac{80L_0 + 512L_1\tau}{\epsilon^2}\Big\},$$

where $\Delta = (f(x_0) - f^* + (5L_0 + 2L_1\tau)\tau^2 + 9\tau L_0^2/L_1)$. By Markov inequality,

$$\mathbb{P}\{\min_{k\leq T} \|\nabla f(x_k)\| \leq \epsilon\} \geq \frac{1}{2}.$$

The theorem follows by the definition in (5.5).

Technical lemmas

Lemma 3.7.2.

$$\mathbb{E}[-h_k\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle] \leq \frac{1}{4}\mathbb{E}[h_k]\|*\|\nabla f(x_k)^2.$$

Proof. By unbiasedness of g_k and the fact that η is a constant, we have

$$\begin{aligned}
\mathbb{E}[-h_k\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle] &= \mathbb{E}[(\eta - h_k)\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle] \\
&= \mathbb{E}[(\eta - h_k)\langle \nabla f(x_k), g_k - \nabla f(x_k)\rangle \mathbb{1}_{\{\|g_k\| \geq \frac{1}{16L_1\eta} - \tau\}}] \\
&\leq \eta\|*\|\nabla f(x_k)\mathbb{E}[\|g_k - \nabla f(x_k)\|\mathbb{1}_{\{\|g_k\| \geq \frac{1}{16L_1\eta} - \tau\}}] \\
&\leq \eta\|\nabla f(x_k)\|^2 32L_1\mathbb{E}[h_k]\tau.
\end{aligned}$$

The second last inequality follows by $h_k \leq \eta$ and Cauchy-Schwartz inequality. The last inequality follows by

$$\|\nabla f(x_k)\| \geq \|g_k\| - \tau = \frac{1}{16L_1h_k} - 2\tau \geq \frac{1}{16L_1h_k} - \frac{1}{32L_1\eta} \geq \frac{1}{32L_1h_k}. \tag{3.15}$$

The equality above holds because $h_k = \frac{1}{16\eta L_1(\|g_k\|+\tau)}$. The lemma follows by $32\eta L_1\tau \leq 1/4$. □

Lemma 3.7.3.

$$\mathbb{E}[(5L_0 + 4L_1\|\nabla f(x_k)\|)h_k^2\langle\nabla f(x_k), g_k - \nabla f(x_k)\rangle] \leq 9\eta^2\tau L_0^2/L_1 + \frac{1}{8}\mathbb{E}[h_k]\|\nabla f(x_k)\|^2.$$

Proof. When $\|\nabla f(x_k)\| \geq L_0/L_1$,

$$\begin{aligned}\mathbb{E}[(5L_0 + 4L_1\|\nabla f(x_k)\|)h_k^2\langle\nabla f(x_k), g_k - \nabla f(x_k)\rangle] \leq & 9L_1\|\nabla f(x_k)\|\mathbb{E}[h_k^2]\|\nabla f(x_k)\|\tau \\ \leq & \frac{1}{8}\mathbb{E}[h_k]\|\nabla f(x_k)\|^2.\end{aligned}$$

The last inequality follows by (3.11).

When $\|\nabla f(x_k)\| \leq L_0/L_1$,

$$\mathbb{E}[(5L_0 + 4L_1\|\nabla f(x_k)\|)h_k^2\langle\nabla f(x_k), g_k - \nabla f(x_k)\rangle] \leq 9\eta^2\tau L_0^2/L_1.$$

□

Lemma 3.7.4.

$$\frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2(\|g_k - \nabla f(x_k)\|^2)] \leq (5L_0 + 2L_1\tau)\eta^2\tau^2 + \frac{1}{8}\|\nabla f(x_k)\|^2\mathbb{E}[h_k].$$

Proof. When $\|\nabla f(x_k)\| \geq L_0/L_1 + \tau$, we get

$$\frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2(\|g_k - \nabla f(x_k)\|^2)] \leq 5L_1\|\nabla f(x_k)\|^2\mathbb{E}[h_k]\eta\tau \leq \frac{1}{8}\|\nabla f(x_k)\|^2\mathbb{E}[h_k].$$

The first inequality follows by $h_k \leq \eta$ and $\|g_k - \nabla f(x_k)\| \leq \tau \leq \|\nabla f(x_k)\|$.The last inequality follows by (3.11).

When $\|\nabla f(x_k)\| \leq L_0/L_1 + \tau$, we get

$$\frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2(\|g_k - \nabla f(x_k)\|^2)] \leq (5L_0 + 2L_1\tau)\eta^2\tau^2.$$

□

3.7.5 Proof of Theorem 3.4.5

Similar to proof of Theorem 3.4.1, we have

$$\begin{aligned}\mathbb{E}[f(x_{k+1})|] \leq & f(x_k) - \mathbb{E}[h_k\langle g_k, \nabla f(x_k)\rangle] + \frac{5L_0 + 4L_1\|\nabla f(x_k)\|}{2}\mathbb{E}[h_k^2\|g_k\|^2] \\ \leq & f(x_k) - \frac{1}{\sqrt{T}}\|*\|\nabla f(x_k)^2 + \frac{5L_0 + 4L_1M(M+\tau)^2}{2T}.\end{aligned}$$

Sum across $k \in \{0, ..., T-1\}$ and take expectations, then we can get

$$0 \leq f(x_0) - \mathbb{E}[f(x_T)] - \frac{1}{\sqrt{T}}\sum_{k=1}^{T}\mathbb{E}\big[\|*\|\nabla f(x_k)^2\big] + \frac{5L_0 + 4L_1M(M+\tau)^2}{2}.$$

Rearrange and we get

$$\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\big[\|*\|\nabla f(x_k)^2\big] \leq \frac{1}{\sqrt{T}}\bigg(f(x_0) - f^* + \frac{5L_0 + 4L_1M(M+\tau)^2}{2}\bigg).$$

By Jensen's inequality,

$$\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}[\|*\|\nabla f(x_k)] \leq \sqrt{\frac{1}{\sqrt{T}}\bigg(f(x_0) - f^* + \frac{5L_0 + 4L_1M(M+\tau)^2}{2}\bigg)}.$$

By Markov inequality,

$$\mathbb{P}\bigg\{\frac{1}{T}\sum_{k=1}^{T}\big[\|*\|\nabla f(x_k)^2\big] > \frac{2}{\sqrt{T}}\bigg(f(x_0) - f^* + \frac{5L_0 + 4L_1M(M+\tau)^2}{2}\bigg)\bigg\} \leq 0.5.$$

The theorem follows by the definition in (5.5) and Jensen's inequality.

Chapter 4

The Role of Nosie Distribution in Gradient Methods

One mystery in neural network experiments is about when to use Adam and when to use SGD. Researchers found that these two most popular optimizers do not dominate one another. In this chapter, we aim to investigate this problem by looking at the noise distribution of the stochastic gradients.

In particular, we examine the properties of the oracle calls and show that in practice adaptive steps can be faster due to the existence of heavy-tailed noise. While stochastic gradient descent (SGD) is still the *de facto* algorithm in deep learning, adaptive methods like Clipped SGD/Adam have been observed to outperform SGD across important tasks, such as attention models. The settings under which SGD performs poorly in comparison to adaptive methods are not well understood yet. We provide empirical and theoretical evidence that a heavy-tailed distribution of the noise in stochastic gradients is one cause of SGD's poor performance. We provide the first tight upper and lower convergence bounds for adaptive gradient methods under heavy-tailed noise.

4.1 Introduction

Stochastic gradient descent (SGD) is the canonical algorithm for training neural networks [Robbins and Monro, 1951]. SGD iteratively updates model parameters in the negative gradient direction and seamlessly scales to large-scale settings. Though a well-tuned SGD outperforms adaptive methods [Wilson et al., 2017] in many tasks including ImageNet classification (see Figure 4-1a), certain tasks necessitate the use of *adaptive* variants of SGD (Adagrad, ADAM, etc), which employ adaptive learning rates. For instance, consider training an attention model [Vaswani et al., 2017] using BERT [Devlin et al., 2018]. Figure 4-1e shows that in spite of extensive hyperparameter tuning, SGD converges much slower than Adam during BERT training.

In this work, we provide one explanation for why adaptivity can facilitate convergence with theoretical and empirical evidence. The significant hint that initializes our work comes from the distribution of the stochastic gradients. For ImageNet, the norms of the mini-batch gradients are typically quite small and well concentrated around their mean. On the other hand, the mini-batch gradient norms for BERT take a wide range of values and are sometimes much larger than their mean value. More formally, while the distribution of the stochastic gradients in ImageNet is well approximated by a Gaussian, the distribution for BERT seems to be *heavy-tailed*. Such observation leads us to the question: *does adaptivity stabilize optimization under heavy-tailed noise?*

We provide a positive answer to the above question by performing both theoretical and empirical studies of the convergence of optimization methods under heavy-tailed noise. In this setting, some of the stochastic gradients are much larger than the mean and can excessively influence the updates of SGD. This makes SGD unstable and leads to its poor performance. A natural strategy to stabilize the updates is to *clip* the magnitude of the stochastic gradients. We prove that indeed it is sufficient to ensure convergence even under heavy-tailed noise. Based on the analysis, we then motivate the design of a novel algorithm (ACClip) that outperforms ADAM on BERT related tasks.

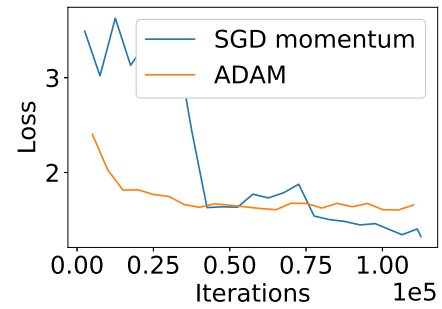
4.2 Related work on noise in neural networks

Noise in neural network. There has been little study of the actual stochastic gradient noise distributions in neural network training. To our knowledge, Şimşekli et al. [2019a], Nguyen et al. [2019], Şimşekli et al. [2019b] start the topic and observe heavy tailed noise in network training. Our work differs in two important ways: *First*, we treat the noise as a high dimensional vector, while [Şimşekli et al., 2019b] treat deviations in each coordinate as scaler noises to estimate tail index. Hence, we observe that the example given in [Şimşekli et al., 2019b] is well-concentrated when viewed as a random vector. This is also confirmed by Panigrahi et al. [2019]. More details can be found in our publication [Zhang et al., 2019c] *Second*, we focus on convergence of optimization algorithm, the previously mentioned works focus on Langevin dynamics and escaping saddle points. The convergence rate given in [Nguyen et al., 2019] is for global Hölder-continuous functions, which restricts the function variations and excludes examples like quadratic functions. Our analysis instead provides the first convergence rates under the standard L-smoothness setting. Further, Gorbunov et al. [2020] studies accelerated first order methods under less concentrated noise, however, there "heavy-tailedness" refers to non-sub-Gaussianity.

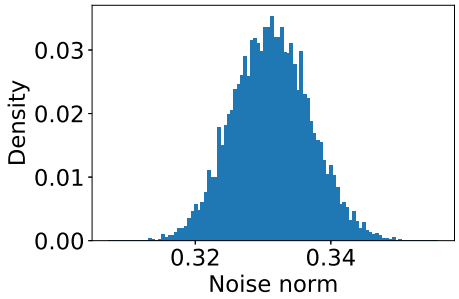
4.3 Heavy-tailed noise in stochastic gradients

To gain intuition about the difference between SGD and adaptive methods, we start our discussion with the study of noise distributions of stochastic gradient that arise during neural network training. In particular, we focus on noise distributions while training two popular deep learning models — BERT and ResNet. Note that BERT and ResNet are typically trained with Adam and SGD (with momentum) respectively and can thus, provide insights about difference between these optimizers.

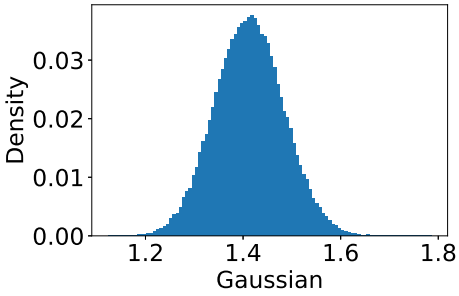
We first investigate the distribution of the gradient noise norm $\|g - \nabla f(x)\|$ in the aforementioned neural network models, where g is the stochastic gradient computed from a minibatch sample. In particular, we fix the model at initialization



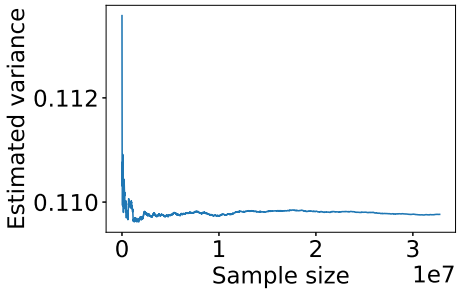
(a)



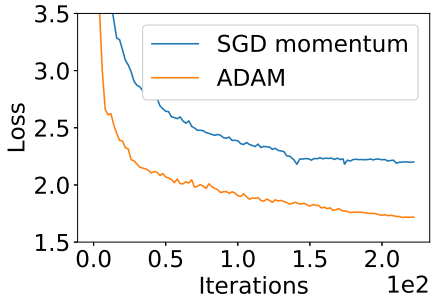
(b) ImageNet training



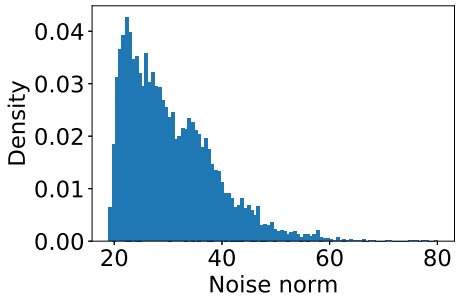
(c) Synthetic Gaussian



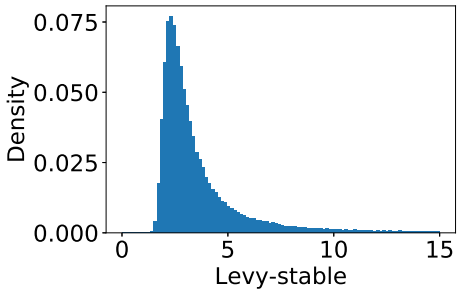
(d) ImageNet variance



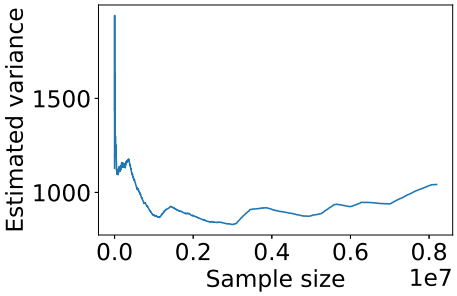
(e)



(f) Bert pretraining



(g) Synthetic Levy-stable



(h) Bert variance

Figure 4-1: (a) Validation loss for ResNet50 trained on ImageNet. SGD momentum outperforms Adam. (b) Histogram of sampled gradient noise for ResNet50 on ImageNet dataset. (c) Histogram of samples from a sum of squared Gaussians. (d) Estimated variance of the stochastic gradient for Resnet50. (e)Validation loss for BERT pretraining. Although hyperparameters for SGD are finetuned, a large performance gap is still observed between SGD and Adam. (f) Histogram of sampled gradient nosie for BERT on Wikipedia+Books dataset. (g) Histogram of samples from a sum of squared α-stable random variables. (h) Estimated variance of the stochastic gradient for BERT model.

without doing any updates. We then iterate through the dataset to compute the noise norm for each minibatch. Figure 4-1 (b) and (f) show these distributions for ResNet50 on ImageNet and BERT on the Wikipedia and books dataset at model initialization respectively. For comparison, we plot distributions of a normalized sum of squared Gaussians, a well-concentrated distribution, and a Levy-α-stable distribution, a heavy-tailed distribution, in Figure 4-1 (c) and (g) respectively. We observe that the noise distribution for BERT appears heavy-tailed, while that of ResNet50 is well-concentrated. Results for noise distributions at other stages of training are displayed in Figure 4-2.

To support this observation, in Figure 4-1 (d) and (h) we further show the empirical variance of stochastic gradients with respect to the sample size used in the estimation. The results highlight that while the corresponding estimator converges for ImageNet, the empirical variance does not converge in BERT training even as the sample size approaches 10^7.

From the obeservation that the noise can be heavy-tailed, we hypothesize that this is one major aspect that determines the performance of SGD and adaptive methods. In the rest of the chapter, we argue and provide evidence that adaptive methods can be faster than SGD in scenarios where heavy-tailed noise distributions arise. More experiment details can be found in Section 4.6.

4.4 Convergence of gradient methods under heavy-tailed noise

In this section we study the performance of SGD and adaptive methods under heavy-tailed noise. More precisely, we analyze algorithms of the following form

$$x_{k+1} = x_k - \eta_k g_k, \tag{4.1}$$

where x_k represent the current parameters, η_k is the step size and g_k is the stochastic (mini-batch) gradient evaluated at x_k. We show that if the stochasticity in the gra-

dient g_k is heavy-tailed, it is critical for the step sizes to be ***adaptive***, i.e., η_k must depend on the observed gradients. We propose to use one such algorithm GClip and prove that it obtains optimal convergence rates.

Heavy-tailed noise. Neural network training can be seen as minimizing a differentiable stochastic function $f(x) = \mathbb{E}_\xi[f(x,\xi)]$, where $f : \mathbb{R}^d \to \mathbb{R}$ can be potentially nonconvex and ξ represents the mini-batches. At each iteration, we assume access to an *unbiased* stochastic gradient $\mathbb{E}[g(x)] = \nabla f(x,\xi)$ corresponding to the parameters x, mini-batch ξ. We also need to bound how much noise is present in our stochastic gradients. In lieu of the usual bounded variance assumption, we use

Assumption 4.4.1 (Bounded $\alpha-$moment). There exists positive real numbers $\alpha \in (1,2]$ and $G > 0$ such that for all x, $\mathbb{E}[\|g(x) - \nabla f(x)\|^\alpha] \leq \sigma^\alpha$. We say noise is **heavy-tailed** if $\alpha < 2$.

The above assumption with $\alpha = 2$ corresponds to the standard variance bound, but in general is weaker. It is indeed possible (e.g., Pareto or α-stable Levy random variables) for the variance of $g(x)$ to be unbounded, while simultaneously satisfying assumption 4.4.1 for $\alpha < 2$. One should note that even if the variance may not actually be infinite in practice, it might be too large to be practically useful. All our analyses and insights carry over to this setting as well.

The possibility that the variance is unbounded has a profound impact on the optimization process.

Remark 4.4.1 (Nonconvergence of SGD)**.** Consider the function $f(x) = x^2/2$ with noise satisfying $\mathbb{E}[\|g(x) - \nabla f(x)\|^\alpha] = \sigma^\alpha$ for $\alpha < 2$, and $\mathbb{E}[\|g(x) - \nabla f(x)\|^2] = \infty$. Then, for any positive constants η_k that do not depend on g_k, we have that $\mathbb{E}[\|\nabla f(x_k)\|^2] = \infty$.

Proof. we denote the stochastic gradient $g_k := g(x_k) = \nabla f(x_k) + \xi_k = x_k + \xi_k$, where $\xi_k \in \mathbb{R}^d$ is a random variable with $\mathbb{E}\|\xi\|^2 = \infty, \mathbb{E}\|\xi\|^\alpha = \sigma^\alpha, \mathbb{E}[\xi] = \vec{0}$. Then, $\mathbb{E}[\|\nabla f(x_{k+1})\|^2] = \mathbb{E}[\|x_{k+1}\|^2] = \mathbb{E}\|x_k - \eta_k g_k\|^2 = \mathbb{E}\|x_k - \eta_k(x_k + \xi)\|^2 = \mathbb{E}\|(1 - \eta_k)x_k - \eta_k\xi\|^2 = \mathbb{E}\|(1-\eta_k)x_k\|^2 - 2(1-\eta_k)\eta_k x_k^\top \mathbb{E}[\xi] + \eta_k^2\mathbb{E}\|\xi\|^2 \geq \eta_k^2\mathbb{E}\|\xi\|^2 = \infty$. Note

Table 4.1: Error bounds. ($f(x) - f^*$ for convex functions, $\|\nabla f(x)\|$ for nonconvex functions) after k iterations: Define α-moment as $\mathbb{E}[\|g(x) - \nabla f(x)\|^\alpha] \leq \sigma^\alpha$ (Assump 4.4.1) in the smooth nonconvex case and $\mathbb{E}[\|g(x)\|^\alpha] \leq G^\alpha$ (Assump 4.4.3) in the strongly case. In the standard setting ($\alpha = 2$), GClip recovers the optimal rates. For heavy-tailed noise ($\alpha \in (1, 2)$), GClip converges both for convex (Thm 4.4.2) and non-convex functions (Thm 4.4.1). We also show matching lower-bounds for all $\alpha \in (1, 2]$ proving the optimality of clipping methods (Thm 4.4.3).

	Strongly Convex Function		Non-Convex Function	
	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)	Heavy-tailed noise ($\alpha \in (1,2)$)	Standard noise ($\alpha \geq 2$)
SGD	N/A	$\mathcal{O}\big(k^{-1}\big)$	N/A	$\mathcal{O}\big(k^{-\frac{1}{4}}\big)$
GClip	$\mathcal{O}\big(k^{\frac{-(\alpha-1)}{\alpha}}\big)$	$\mathcal{O}\big(k^{-1}\big)$	$\mathcal{O}\big(k^{\frac{-(\alpha-1)}{3\alpha-2}}\big)$	$\mathcal{O}\big(k^{-\frac{1}{4}}\big)$
LowerBound	$\Omega\big(k^{\frac{-(\alpha-1)}{\alpha}}\big)$	$\Omega\big(k^{-1}\big)$	$\Omega\big(k^{\frac{-(\alpha-1)}{3\alpha-2}}\big)$	$\Omega\big(k^{-\frac{1}{4}}\big)$

that this holds for *any* fixed $\eta_k > 0$ even if allowed to depend on the statistics of the noise distribution (such as σ or α). □

The issue is that SGD is easily influenced by a single-stochastic gradient, which could be very large and incorrect. A simple strategy to circumvent this issue is to use a biased *clipped* stochastic gradient estimator. This allows us to circumvent the problem of unbounded variance and ensures optimal convergence rates even under heavy-tailed noise. Our results are summarized in Table 4.1, and all proofs are relegated to the Appendices.

4.4.1 Convergence of Clipped Methods

A simple clipping strategy is to globally clip the norm of the update to threshold τ_k:

$$x_{k+1} = x_k - \eta_k \min\Big\{\tfrac{\tau_k}{\|g_k\|}, 1\Big\} g_k\,, \;\; \tau_k \in \mathbb{R}_{\geq 0} \tag{GClip}$$

We refer to this strategy as GClip (Global Clip), as opposed to coordinate-wise clipping which we discuss later. We first state the rates for smooth non-convex functions.

Theorem 4.4.1 (Non-convex convergence). *Suppose that f is L-smooth and that the stochastic gradients satisfy Assumption 4.4.1 for $\alpha \in (1, 2]$. Let $\{x_k\}$ be the*

iterates of GClip with parameters $\eta = \min\{\frac{1}{4L}, \frac{1}{L(12\sigma)^{\alpha/(\alpha-1)}}, \left(\frac{f_0}{\sigma^2 K}\right)^{\frac{\alpha}{3\alpha-2}} / L^{\frac{2\alpha-2}{3\alpha-2}}\}$ *and* $\tau = \sigma(\eta L)^{-1/\alpha}$. *Then for* $F_0 := f(x_0) - f^*$,

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\min\{\|\nabla f(x_k)\|, \|\nabla f(x_k)\|^2\}] \leq 20 f_0 \max\{\tfrac{4L}{K}, \tfrac{L(12\sigma)^{\alpha/(\alpha-1)}}{K}, L^{\frac{2\alpha-2}{3\alpha-2}}\left(\tfrac{F_0}{K\sigma^2}\right)^{\frac{2(\alpha-1)}{3\alpha-2}}\}.$$

Remark 4.4.2. When $\|\nabla f(x_k)\| \leq \epsilon \ll 1$, then $\|\nabla f(x_k)\|^2 \ll \|\nabla f(x_k)\|$. Hence the dominant term on the left hand side of the rate above is $\|\nabla f(x_k)\|^2$. The right hand side is easily observed to be $\mathcal{O}(K^{-\frac{2(\alpha-1)}{3\alpha-2}})$. Together, this implies a convergence rate of $\mathbb{E}\|\nabla f(x)\| \leq \mathcal{O}(K^{-\frac{(\alpha-1)}{3\alpha-2}})$.

We prove improved rates of convergence for non-smooth *strongly-convex* functions in a bounded domain. Before stating the theorem, we list our assumptions and definitions below,

Here we describe some of the formal assumptions which were previously skipped. We define the standard notion of smoothness.

Assumption 4.4.2 (L**-smoothness).** f is L-smooth, i.e. there exist a positive constants L such that $\forall x, y,\ f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$.

We only need the smoothness assumption for non-convex functions.

For strongly-convex optimization, instead of bounding the noise, we assume that the stochastic oracle has bounded moment.

Assumption 4.4.3 (μ**-strong-convexity).** There exists positive real numbers $\alpha \in (1, 2]$ and $G > 0$ such that for all x, $\mathbb{E}[\|g(x)\|^\alpha] \leq G^\alpha$.

Note that the above assumption implies a uniform bound on gradient norm. Such bound is necessary for nonsmooth strongly convex problems, as one can no longer factor out the gradient norm using the smoothness assumption. See for example, Rakhlin et al. [2011].

Assumption 4.4.4 (μ**-strong-convexity).** f is μ-strongly convex, if there exist positive constants μ such that $\forall x, y$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2.$$

The strong convexity assumption and the bounded gradient assumption implicitly implies that the domain is bounded. However, the domain diameter is not used explicitly in the proof. The projected clipped gradient descent follows exactly the same argument and the fact that orthogonal projections contract distances.

$$x_{k+1} = x_k - \eta_k \min\Big\{\tfrac{\tau_k}{\|g_k\|}, 1\Big\} g_k\,,\ \ \tau_k \in \mathbb{R}_{\geq 0}. \tag{proj-GClip}$$

We are now ready to state the theorem.

Theorem 4.4.2 (**Strongly-convex convergence**). *Suppose that the stochastic gradients satisfy Assumption 4.4.3 for $\alpha \in (1, 2]$. Let $\{x_k\}$ be the iterates of projected GClip* (proj-GClip) *with clipping parameter $\tau_k = Gk^{\alpha-1}$ and steps-size $\eta_k = \frac{4}{\mu(k+1)}$. Define the output to be a k-weighted combination of the iterates: $\bar{x}_k = \sum_{j=1}^{k} j x_{j-1} / (\sum_{j=1}^{k} j)$. Then the output $\bar{x}_k$ satisfies:*

$$\mathbb{E}[f(\bar{x}_k)] - f(x^\star) \leq \tfrac{16G^2}{\mu(k+1)^{2(\alpha-1)/\alpha}}\,.$$

The rates of convergence for the strongly convex and non-convex cases in Theorem 4.4.2 and Theorem 4.4.1 exactly match those of the usual SGD rates ($\mathcal{O}(1/\sqrt{k})$ for convex and $\mathcal{O}(k^{-\frac{1}{4}})$ for non-convex) when $\alpha = 2$ and gracefully degrade for $\alpha \in (1, 2]$. As we will next show, both the strongly convex rates and non-convex rates of GClip are in fact optimal for every $\alpha \in (1, 2]$.

4.4.2 Theoretic lower bounds

We prove that the rates obtained with GClip are optimal up to constants. First, we show a strong lower-bound for the class of convex functions with stochastic gradients satisfying $\mathbb{E}[|g(x)|^\alpha] \leq 1$. This matches the upper bounds of Theorems 4.4.2 and 4.5.1 for strongly-convex functions, showing that the simple clipping mechanism of GClip is (up to constants) information theoretically optimal, providing a strong justification for its use.

Theorem 4.4.3. *For any $\alpha \in (1, 2]$ and any (possibly randomized) algorithm $\mathcal{A}$, there exists a problem f which is 1-strongly convex and 1-smooth ($\mu = 1$ and $L = 1$), and stochastic gradients which satisfy Assumptions 4.4.3 with $G \leq 1$ such that the output x_k of the algorithm $\mathcal{A}$ after processing k stochastic gradients has an error*

$$\mathbb{E}[f(x_k)] - f(x^\star) \geq \Omega\big(\tfrac{1}{k^{2(\alpha-1)/\alpha}}\big)\,.$$

Next, we examine non-convex functions.

Theorem 4.4.4. *Given any $\alpha \in (1, 2]$, smoothness constant L, and (possibly randomized) algorithm $\mathcal{A}$, there exists a constant c_1 and an L-smooth function f with stochastic gradients satisfying Assumption 4.4.1 for any given $\sigma \geq c_1\sqrt{(f(0) - f^*)L}$ such that the output x_k of the algorithm $\mathcal{A}$ after processing k stochastic gradients has an error*

$$\mathbb{E}[\|\nabla f(x_k)\|] \geq \Omega\big(\tfrac{1}{k^{(\alpha-1)/(3\alpha-2)}}\big)\,.$$

Theorem 4.4.4, proven in Appendix 4.7.7, extends the recent work of [Arjevani et al., 2019, Theorem 1] to heavy-tailed noise. Here, the lower-bound matches the upper-bound in Theorem 4.4.1 up to constants, proving its optimality.

4.5 Faster Optimization with Adaptive Coordinate-wise Clipping

The previous section showed that adaptive step sizes (which depend on the gradients) are essential for convergence under heavy-tailed noise, and also showed that GClip provides the optimal rates. There are of course other adaptive methods such as Adam that employ not only the current gradients but also *all* past gradients to adaptively set *coordinate-wise* step-sizes. In this section, we study why coordinate-wise clipping may yield even faster convergence than GClip, and show how to modify GClip to design an Adaptive Coordinate-wise Clipping algorithm (ACClip).

4.5.1 Coordinate-wise clipping

The first technique we use is applying coordinate-wise clipping instead of global clipping. We had previously assumed a global bound on the α-moment of the *norm* (or variance) of the stochastic gradient is bounded by σ. However, σ might be hiding some dimension dependence d. We show a more fine-grained model of the noise in order to tease out this dependence.

Assumption 4.5.1 (**Coordinate-wise α moment).** Denote $\{g_i(x)\}$ to be the coordinate-wise stochastic gradients for $i \in [d]$. We assume there exist constants $\{B_i\} \geq 0$ and $\alpha \in (1, 2]$ such that $\mathbb{E}[|*|\, g_i(x)^\alpha] \leq B_i^\alpha$.

For convenience, we denote $B = [B_1; B_2; \cdots; B_d] \in \mathbb{R}^d$, $\|B\|_a = (\sum B_i^a)^{1/a}$. Under this more refined assumption, we can show the following corollary:

Corollary 4.5.1 (**GClip under coordinate-wise noise).** *Suppose we run GClip under Assumption 4.5.1 to obtain the sequence $\{x_k\}$. If f is μ-strongly convex, with appropriate step-sizes and averaging, the output $\bar{x}_k$ satisfies*

$$\mathbb{E}[f(\bar{x}_k)] - f(x^\star) \leq \tfrac{16d\|B\|_\alpha^2}{\mu(k+1)^{2(\alpha-1)/\alpha}} .$$

Thus, the convergence of GClip can have a strong dependence on d, which for large-scale problems might be problematic. We show next that using coordinate-wise clipping removes this dependency:

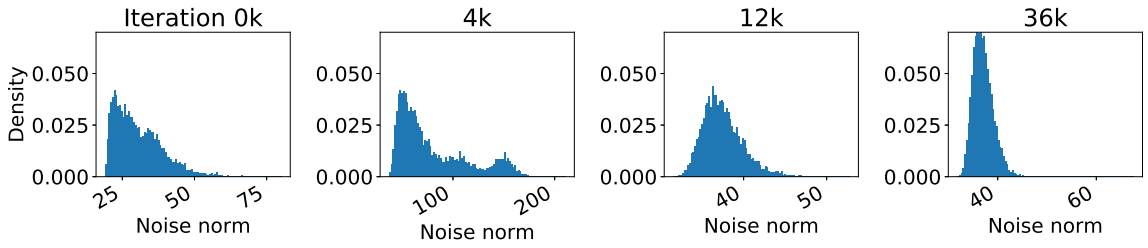
$$\mathrm{CClip}(\tau_k, m_k) = \min\big\{\tfrac{\tau_k}{|m_k|}, 1\big\} m_k\,,\ \tau_k \in \mathbb{R}^d_{\geq 0}\,. \tag{CClip}$$

Theorem 4.5.1 (**CClip under coordinate-wise noise).** *Suppose we run CClip under the Assumption of 4.5.1 with $\tau_k = Bk^{\alpha-1}$ to obtain the sequence $\{x_k\}$. Then, if f is μ-strongly convex, with appropriate step-sizes and averaging, the output $\bar{x}_k$ satisfies*

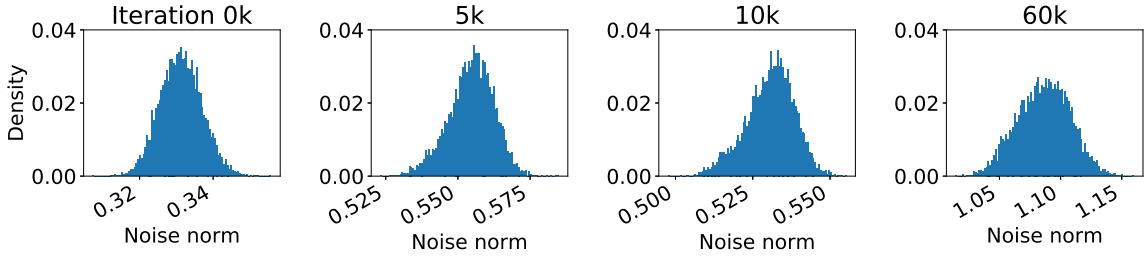
$$\mathbb{E}[f(\bar{x}_k)] - f(x^\star) \leq \tfrac{16\|B\|_2^2}{\mu(k+1)^{2(\alpha-1)/\alpha}} .$$

Note that $\|B\|_2 \leq \|B\|_\alpha$. CClip has a worst-case convergence independent of d under the coordinate-wise noise model. Similar comparison between GClip and CClip

can be done for non-convex conditions too, but we skip for conciseness. Though we only compare upper-bounds here, when the noise across coordinates is independent the upper bounds may be tight (see Lemma 4.7.4).



(a) Development of noise distribution during BERT training.



(b) Development of noise distribution during ResNet50 training on ImageNet.

Figure 4-2: The distribution of gradient noise is non-stationary during BERT training, while it remains almost unchanged for ResNet training on ImageNet.

4.5.2 Online moment estimation

We now present the second technique that is motivated by our observation in Figure 4-2. There, the distribution of gradient noise at the beginning of different epochs is shown during training for BERT with Wikipedia (top) as well as ResNet with ImageNet (bottom). The result highlights that the noise distribution is not only heavy-tailed, but also non-stationary during BERT training and becomes increasingly more concentrated. In contrast, for the ResNet model the noise distribution remains mostly unchanged.

Since the scale of the noise changes drastically during training for BERT model and our theoretical analysis suggest that we should clip proportional to the noise level, we propose to use an exponential moving average estimator to estimate the moment and clip the gradient accordingly (line 4,5 of Alg 1). This, combined with the momentum

Algorithm 1 ACClip.

$x, m_k \leftarrow x_0, 0$
for $k = 1, \cdot, T$ **do**
  $m_k \leftarrow \beta_1 m_{k-1} + (1-\beta_1) g_k$
  $\tau_k^\alpha \leftarrow \beta_2 \tau_{k-1}^\alpha + (1-\beta_2)|g_k|^\alpha$
  $\hat{g}_k \leftarrow \min\{\frac{\tau_k}{|m_k|+\epsilon}, 1\} m_k$
  $x_k \leftarrow x_{k-1} - \eta_k \hat{g}_k$
return x_K, where random variable K is supported on $\{1, \cdots, T\}$.

term leads to our proposed ACClip algorithm in Algorithm 1. On a high level, the algorithm applies clipping to the momentum term, where the clipping threshold is proportional to the estimated moment using an exponential moving average. From our experiment, we found the conservative choice of $\alpha = 1$ leads to the best performance.

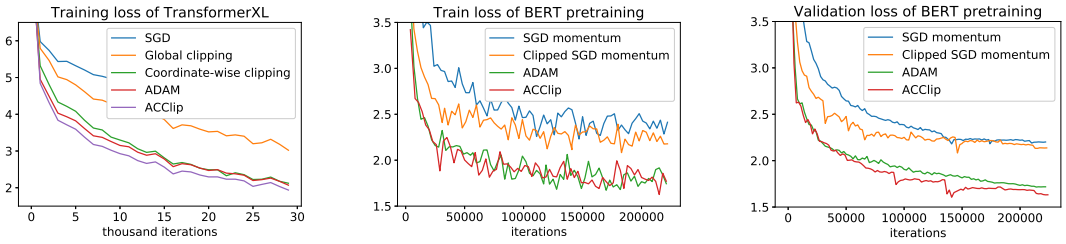
4.6 Experiments

In this section, we first verify the effect of coordinate-wise clipping and moment estimation introduced in Section 4.5. We then perform extensive evaluations of AC-Clip on BERT pre-training and fine-tuning tasks and demonstrate its advantage over Adam in Section 4.6.2. Finally, we start with a few more experiments on the noise distribution in neural network training.

4.6.1 From GClip to ACClip

In this section we instantiate the argument in Section 4.5 with a set of experiments. As seen in Figure 4-3b, global clipping improves the vanilla SGD algorithm but is still far from the ADAM baseline. We apply two techniques (coordinate-wise clipping and online moment estimation) onto the clipped SGD algorithm analyzed in Section 4.4. We use a set of experiments on Transformer-XL training to demonstrate the effect of each technique.

Experiment setup. We train a 6-layer Transformer-XL model [Dai et al., 2019a] on PTB dataset as a proof of concept. Our main experiments will be on BERT pretraining and finetuning described in Section 4.6.2. We adapt the author's github



(a) (b) (c)

Figure 4-3: (a) Performance of different algorithms for training a toy transformer-XL model described in Section 4.5. (b) Train and (c) validation loss for BERT$_{base}$ pretraining with the sequence length of 128. While there remains a large gap between non-adaptive methods and adaptive methods, clipped SGD momentum achieves faster convergence compared to standard SGD momentum. The proposed algorithm for adaptive coordinate-wise clipping (ACClip) achieves a lower loss than Adam.

repo[1], and replace the number of layers of the base model by 6. We then select the PTB data as input and set the maximum target length to be 128. The results are shown in Figure 4-3b.

Observations. From Figure 4-3b, we can tell that global clipping (orange curve) indeed speeds up vanilla SGD but is still much worse compared to the ADAM baseline provided by the code base. After replacing global clipping with coordinate-wise clipping, we see that the performance is already comparable to the ADAM baseline. Finally, after using the moment estimation to determine the clipping threshold, we are able to achieve faster convergence than ADAM.

4.6.2 Performance of ACClip for BERT experiments

We now evaluate the empirical performance of our proposed ACClip algorithm on BERT pre-training as well fine-tuning using the SQUAD v1.1 dataset. As a baseline, we use Adam optimizer and the same training setup as in the BERT paper [Devlin et al., 2018]. For ACClip, we set $\tau = 1$, *learning rate* = 1e-4, $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon =$ 1e-5 and *weight decay* = 1e-5. We compare both setups on BERT models of three different sizes, BERT$_{\text{base}}$ with 6 and 12 layers as well as BERT$_{\text{large}}$ with 24 layers.

[1]https://github.com/kimiyoung/transformer-xl/tree/master/pytorch

Figure 4-3b and 4-3c shows the loss for pretraining $\text{BERT}_{\text{base}}$ using SGD with momentum, GClip, Adam and ACClip. The learning rates and hyperparameters for each method have been extensively tuned to provide best performance on validation set. However, even after extensive tuning, there remains a large gap between (clipped) SGD momentum and adaptive methods. Furthermore, clipped SGD achieves faster convergence as well as lower final loss compared to standard SGD. Lastly, the proposed optimizer ACClip achieves a lower loss than the Adam. Table 4.2 further shows that ACClip achieves lower loss and higher masked-LM accuracy for all model sizes.

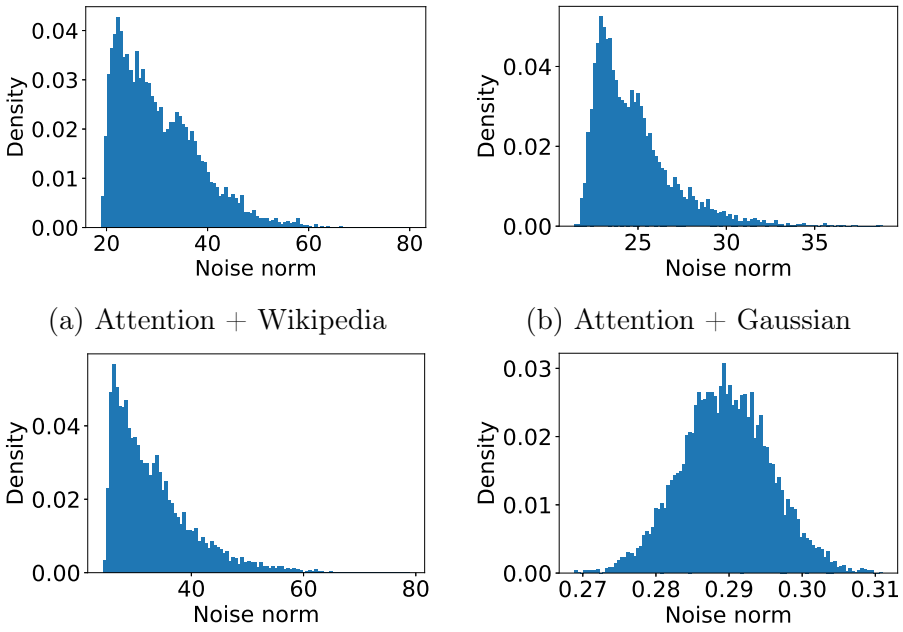
Next, we evaluate ACClip on the SQUAD v1.1 fine-tuning task. We again follow the procedure outlined in [Devlin et al., 2018] and present the results on the Dev set in Table 4.3. Both for F1 as well as for exact match, the proposed algorithm outperforms Adam on all model sizes. The experimental results on BERT pretraining and fine-tuning indicate the effectiveness of the proposed algorithm.

Table 4.2: **BERT pretraining: Adam vs ACClip**. Compared to Adam, the proposed ACClip algorithm achieves better evaluation loss and Masked LM accuracy for all model sizes.

	BERT Base 6 layers		**BERT Base 12 layers**		**BERT Large 24 layers**	
	Val. loss	Accuracy	Val. loss	Accuracy	Val. loss	Accuracy
Adam	1.907	63.45	1.718	66.44	1.432	70.56
ACClip	**1.877**	**63.85**	**1.615**	**67.16**	**1.413**	**70.97**

Table 4.3: **SQUAD v1.1 dev set: Adam vs ACClip**. The mean and standard deviation of F1 and exact match score for 5 runs. The first row contains results reported from the original BERT paper, which are obtained by picking the best ones out of 10 repeated experiments.

	BERT Base 6 layers		**BERT Base 12 layers**		**BERT Large 24 layers**	
	EM	F1	EM	F1	EM	F1
Adam (Devlin et al., 2018)			80.8	88.5	84.1	90.9
Adam	76.85 ± 0.34	84.79 ± 0.33	81.42 ± 0.16	88.61 ± 0.11	83.94 ± 0.19	90.87 ± 0.12
ACClip	**78.07 ± 0.24**	**85.87 ± 0.13**	**81.62 ± 0.18**	**88.82 ± 0.10**	**84.93 ± 0.29**	**91.40 ± 0.15**



(a) Attention + Wikipedia

(b) Attention + Gaussian

(c) Resnet + Wikipedia

(d) Resnet + Gaussian.

Figure 4-4: Distribution of gradient noise norm in Attention and ResNet models on two data sources: Wikipedia and synthetic Gaussian. The heavy-tailed noise pattern results from the interaction of both model architecture as well as data distribution.

4.6.3 Noise Patterns in BERT and ImageNet Training

In our initial analysis in Figure 4-1, we observe that training an attention model on Wikipedia leads to heavy-tailed noise whereas training a ResNet on ImageNet data leads to well-concentrated noise. Here, we aim to disentangle the effect that model architecture and training data have on the shape of gradient noise. To this end, we measure the distribution of the gradient noise norm in an Attention and a ResNet model on both Wikipedia and synthetic Gaussian data. We used $\text{BERT}_{\text{base}}$ as the Attention model, and the ResNet is constructed by removing the self-attention modules within the transformer blocks. Gaussian synthetic data is generated by replacing the token embedding layer with normalized Gaussian input. The resulting noise histograms are shown in Figure 4-4. The figure shows that the Attention model leads to heavy-tailed noise independently of input data. For the ResNet model, we observe that Gaussian input leads to Gaussian noise, whereas Wikipedia data leads to be heavy-tailed noise. We thus conclude that the heavy-tailed noise pattern results

from both the model architecture as well as the data distribution.

4.7 Appendix

We focus on (GClip) under stochastic gradients which satisfy Assumption 4.4.1.

Lemma 4.7.1. *For any $g(x)$ suppose that assumption 4.4.1 holds with $\alpha \in (1, 2]$. Then the estimator $\hat{g} := \min\{\frac{\tau_k}{\|g_k\|}, 1\}g_k$ from* (GClip) *with global clipping parameter $\tau \geq 0$ satisfies:*

$$\mathbb{E}\big[\|\hat{g}(x)\|^2\big] \leq G^{\alpha}\tau^{2-\alpha} \text{ and } \|\mathbb{E}[\hat{g}(x)] - \nabla f(x)\|^2 \leq G^{2\alpha}\tau^{-2(\alpha-1)}\,.$$

Proof. First, we bound the variance.

$$\mathbb{E}[\|\hat{g}(x)\|^2] = \mathbb{E}[\|\hat{g}(x)\|^{\alpha}\|\hat{g}(x)\|^{2-\alpha}].$$

By the fact that $\hat{g}(x) \leq \tau$, we get

$$\mathbb{E}[\|\hat{g}(x)\|^2] = \mathbb{E}[\|\hat{g}(x)\|^{\alpha}\tau^{2-\alpha}] \leq G^{\alpha}\tau^{2-\alpha}.$$

Next, we bound the bias,

$$\begin{aligned}
&\|\mathbb{E}[\hat{g}(x)] - \nabla f(x)\| = \|\mathbb{E}[\hat{g}(x) - g(x)]\| \\
&\leq \mathbb{E}[\|\hat{g}(x) - g(x)\|] = \mathbb{E}[\|\hat{g}(x) - g(x)\|\mathbb{1}_{\{|g(x)|\geq\tau\}}] \\
&\leq \mathbb{E}\big[\|g(x)\|\mathbb{1}_{\{|g(x)|\geq\tau\}}\big] \leq \mathbb{E}[\|g(x)\|^{\alpha}]\tau^{-(\alpha-1)}.
\end{aligned}$$

The last inequality follows by Markov inequality. □

As we increase the clipping parameter τ, note that the variance (the first term in Lemma 4.7.1) increases while the bias (which is the second term) decreases. This way, we can carefully trade-off the variance of our estimator against its bias, thereby ensuring convergence of the algorithm.

4.7.1 Non-convex Rates (Proof of Theorem 4.4.1)

The lemma in the previous section can be readily used in the nonsmooth strongly convex setting. However, we need a variant of Lemma 4.7.1 in the smooth case.

Lemma 4.7.2. *For any $g(x)$ suppose that assumption 4.4.1 holds with $\alpha \in (1, 2]$. If $\|\nabla f(x)\| \leq \tau/2$, then the estimator $\hat{g} := \min\{1, \tau/\|g_k\|\}g_k$ from* (GClip) *with global clipping parameter $\tau \geq 0$ satisfies:*

$$\mathbb{E}\big[\|\hat{g}(x)\|^2\big] \leq 2\|\nabla f(x)\|^2 + 4\sigma^\alpha \tau^{2-\alpha} \text{ and } \|\mathbb{E}[\hat{g}(x)] - \nabla f(x)\|^2 \leq 4\sigma^{2\alpha}\tau^{-2(\alpha-1)}\,.$$

Proof. First, we bound the variance.

$$\begin{aligned}
\mathbb{E}[\|\hat{g}(x)\|^2] &\leq \mathbb{E}[2\|\nabla f(x)\|^2 + 2\|\nabla f(x) - \hat{g}(x)\|^2] \\
&\leq \mathbb{E}[2\|\nabla f(x)\|^2 + 2\|\nabla f(x) - \hat{g}(x)\|^\alpha (2\tau)^{2-\alpha}] \\
&\leq 2\|\nabla f(x)\|^2 + 4\tau^{2-\alpha}\mathbb{E}[\|\nabla f(x) - g(x)\|^\alpha] \\
&\leq 2\|\nabla f(x)\|^2 + 4\tau^{2-\alpha}\sigma^\alpha.
\end{aligned}$$

The expectation is taken with respect to the randomness in noise. The second last inequality follows by the following fact: for any vector $v \in \mathbb{R}^d$ such that $\|v\| \geq \tau$, $\|\nabla f(x)\| \leq \tau/2$ implies that $\|\nabla f(x) - v\| \geq \|\nabla f(x) - \frac{\tau v}{\|v\|}\|$.

Next, we bound the bias,

$$\begin{aligned}
&\|\mathbb{E}[\hat{g}(x)] - \nabla f(x)\| = \|\mathbb{E}[\hat{g}(x) - g(x)]\| \\
&= \mathbb{E}[|\|g(x)\| - \tau| \mathbb{1}_{\{\|g(x)\| > \tau\}}] \\
&\leq \mathbb{E}[\|g(x) - \nabla f(x)\| \mathbb{1}_{\{\|g(x)\| > \tau\}}] \\
&\leq \mathbb{E}[\|g(x) - \nabla f(x)\| \mathbb{1}_{\{\|g(x) - \nabla f(x)\| > \tau/2\}}] \\
&\leq \mathbb{E}[\|g(x) - \nabla f(x)\|^\alpha](\tau/2)^{1-\alpha} \leq 2\sigma^\alpha \tau^{1-\alpha}.
\end{aligned}$$

The last inequality follows by Markov inequality. □

Next, we need a subprocedure at the end proof of Lemma 2 from Cutkosky and

Mehta [2020].

Lemma 4.7.3 (Lemma 2 in Cutkosky and Mehta [2020]). *For any vector $v \in \mathbb{R}^d$, $\langle v/\|v\|, \nabla f(x)\rangle \geq \frac{\|\nabla f(x)\|}{3} - \frac{8\|v-\nabla f(x)\|}{3}$.*

Finally, we are ready to show the proof.

At each iteration, we consider two cases, either $\|\nabla f(x_k)\| < \tau/2$ or $\|\nabla f(x_k)\| \geq \tau/2$.

Case 1: $\|\nabla f(x_k)\| < \tau/2$ For simplicity, we denote $\hat{g}_k = \min\{1, \tau/\|g_k\|\}g_k$ and the bias $b_k = \mathbb{E}[\hat{g}_k] - \nabla f(x_k)$. By Assumption 4.4.2, we have

$$\begin{aligned}
f(x_k) &\leq f(x_{k-1}) + \langle \nabla f(x_{k-1}), -\eta_k \hat{g}_{k-1}\rangle + \frac{\eta_{k-1}^2 L}{2}\|\hat{g}_{k-1}\|^2 \\
&\leq f(x_{k-1}) - \eta_{k-1}\|\nabla f(x_{k-1})\|^2 - \eta_{k-1}\langle \nabla f(x_{k-1}), b_{k-1}\rangle + \frac{\eta_{k-1}^2 L}{2}\|\hat{g}_{k-1}\|^2 \\
&\leq f(x_{k-1}) - \eta_{k-1}\|\nabla f(x_{k-1})\|^2 - \eta_{k-1}\langle \nabla f(x_{k-1}), b_{k-1}\rangle + \frac{\eta_{k-1}^2 L}{2}\|\hat{g}_{k-1}\|^2 \\
&\leq f(x_{k-1}) - \eta_{k-1}\|\nabla f(x_{k-1})\|^2 + \frac{\eta_{k-1}}{2}\|*\|\nabla f(x_{k-1})^2 + \frac{\eta_{k-1}}{2}\|b_{k-1}\|^2 + \frac{\eta_{k-1}^2 L}{2}\|\hat{g}_{k-1}\|^2 .
\end{aligned}$$

Here the last step used the AM-GM inequality. Then, taking expectation in both sides and using Lemma 4.7.2 gives

$$\begin{aligned}
\mathbb{E}[f(x_k)|x_{k-1}] &\leq f(x_{k-1}) - (\frac{\eta_{k-1}}{2} - \eta L)\|*\|\nabla f(x_{k-1})^2 + 2\eta_{k-1}^2 L\sigma^\alpha \tau^{2-\alpha} + \frac{2\eta_k \sigma^{2\alpha}}{\tau^{2\alpha-2}} \\
&\leq f(x_{k-1}) - \frac{\eta_{k-1}}{4}\|*\|\nabla f(x_{k-1})^2 + 2\eta_{k-1}^2 L\sigma^\alpha \tau^{2-\alpha} + \frac{2\eta_{k-1} \sigma^{2\alpha}}{\tau^{2\alpha-2}} .
\end{aligned}$$

In the last step we used $\{\eta_k = \eta \leq \frac{1}{4L}\}$.

Case 2: $\|\nabla f(x_k)\| > \tau/2$ Recall $\hat{g}_k = \min\{1, \tau/\|g_k\|\}g_k$. Denote $p = \mathbb{P}\{\|g_k\| \leq \tau\}$. Hence we know that

$$\begin{aligned}\mathbb{E}[\langle \nabla f(x_k), \hat{g}_k \rangle] &= \mathbb{E}[\frac{\|\nabla f(x_k)\|^2}{\tau} \mathbb{1}_{\{\|g_k\| \leq \tau\}}] + \mathbb{E}[\langle g_k/\|g_k\|, \nabla f(x_k)\rangle \mathbb{1}_{\{\|g_k\| \geq \tau\}}] \\ &\geq \frac{p\|\nabla f(x_k)\|^2}{\tau} + \mathbb{E}[\frac{\|\nabla f(x)\|}{3} \mathbb{1}_{\{\|g_k\| \geq \tau\}}] + \mathbb{E}[\frac{8\|g_k - \nabla f(x)\|}{3}] \\ &\geq \frac{p\|\nabla f(x_k)\|^2}{\tau} + (1-p)\frac{\|\nabla f(x)\|}{3} + \frac{8\sigma}{3} \\ &\geq \frac{\|\nabla f(x_k)\|}{3} + \frac{8\sigma}{3} \geq \frac{\|\nabla f(x_k)\|}{6}.\end{aligned}$$

The second inequality used Lemma 4.7.3. The third inequality and fourth inequality used that $\|\nabla f(x)\| \geq \tau/2 \geq 8\sigma$. By Assumption 4.4.2, we have

$$\begin{aligned}\mathbb{E}[f(x_k)] &\leq f(x_{k-1}) + \mathbb{E}[\langle \nabla f(x_{k-1}), -\eta_k \hat{g}_k \rangle] + \frac{\eta_k^2 L}{2}\tau^2 \\ &\leq f(x_{k-1}) - \eta_k \|\nabla f(x_{k-1})\|/6 + \eta_k^2 L\tau \|\nabla f(x_{k-1})\| \leq f(x_{k-1}) - \eta_k \|\nabla f(x_{k-1})\|/12.\end{aligned}$$

The last inequality follows by $\frac{1}{L(12\sigma)^{\alpha/(\alpha-1)}}, \tau = \frac{\sigma}{(L\eta)^{\frac{1}{\alpha}}}$ implies that $\eta L\tau \leq \frac{1}{12}$.

Combining the two cases, we have the inequality below,

$$\mathbb{E}[f(x_k)|x_{k-1}] \leq f(x_{k-1}) - \frac{\eta}{12}\min\{\|*\|\nabla f(x_{k-1})^2, \|*\|\nabla f(x_{k-1})\} + 2\eta^2 L\sigma^\alpha \tau^{2-\alpha} + \frac{2\eta\sigma^{2\alpha}}{\tau^{2\alpha-2}}.$$

Rearrange and sum the terms above for some fixed step-size and threshold $\{\tau_k = \tau\}$ to get

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\big[\min\{\|\nabla f(x_{k-1})\|^2, \|\nabla f(x_{k-1})\|\}\big] &\leq \frac{4}{\eta K}(f(x_0) - \mathbb{E}[f(x_K)]) + 8\eta L\sigma^\alpha\tau^{2-\alpha} + 8\frac{\sigma^{2\alpha}}{\tau^{2\alpha-2}} \\ &\leq \underbrace{\frac{4}{\eta K}(f(x_0) - f^\star)}_{T_1} + \underbrace{8\eta L\sigma^\alpha\tau^{2-\alpha} + \frac{8\sigma^{2\alpha}}{\tau^{2\alpha-2}}}_{T_2}.\end{aligned}$$

Since we use a threshold $\tau = \frac{\sigma}{(L\eta)^{\frac{1}{\alpha}}}$, we can simplify T_2 as

$$\eta L\sigma^\alpha\tau^{2-\alpha} + \frac{\sigma^{2\alpha}}{\tau^{2\alpha-2}} = \frac{2\sigma^{2\alpha}}{\tau^{2\alpha-2}} = 2\sigma^2(\eta L)^{\frac{2\alpha-2}{\alpha}}.$$

Denote $f_0 = f(x_0) - f^\star$ to ease notation. Then, adding T_2 back to T_1 and using a step-size $\eta \leq \left[\left(\frac{f_0}{G^2 K}\right)^{\frac{\alpha}{3\alpha-2}} / L^{\frac{2\alpha-2}{3\alpha-2}}\right]$ we get

$$T_1 + T_2 \leq \frac{4f_0}{\eta K} + 16G^2(\eta L)^{\frac{2\alpha-2}{\alpha}} = \frac{20f_0}{\eta K} \leq \frac{20f_0}{K} \max\{4L, L(12\sigma)^{\alpha/(\alpha-1)}, L^{\frac{2\alpha-2}{3\alpha-2}} / \left(\frac{f_0}{G^2 K}\right)^{\frac{\alpha}{3\alpha-2}}\}.$$

This proves the statement of the theorem. □

4.7.2 Strongly-Convex Rates (Proof of Theorem 4.4.2)

For simplicity, we denote $\hat{g}_k = \min\left\{\frac{\tau_k}{\|g_k\|}, 1\right\} g_k$ and the bias $b_k = \mathbb{E}[\hat{g}_k] - \nabla f(x_k)$. Then we can iteratively bound the distance as follows,

$$\begin{aligned}
\|x_k - x^*\|^2 &= \|x_{k-1} - \eta_k \hat{g}_{k-1} - x^*\|^2 \\
&= \|x_{k-1} - x^*\|^2 - 2\eta_k \langle x_{k-1} - x^*, \nabla f(x_{k-1})\rangle \\
&- 2\eta_k \langle x_{k-1} - x^*, b_{k-1}\rangle + \eta_k^2 \|\hat{g}_{k-1}\|^2 \\
&\leq (1 - \mu\eta_k)\|x_{k-1} - x^*\|^2 - 2\eta_k (f(x_{k-1}) - f^*)) \\
&+ 2\eta_k (\frac{\mu}{4}\|x_{k-1} - x^*\|^2 + \frac{4}{\mu}\|*\|{b_k}^2) + \eta_k^2 \|\hat{g}_{k-1}\|^2.
\end{aligned}$$

Rearrange and we get

$$f(x_{k-1}) - f^* \leq \frac{\eta_k^{-1} - \mu/2}{2}\|x_{k-1} - x^*\|^2 - \frac{\eta_k^{-1}}{2}\|x_k - x^*\|^2 + \frac{4}{\mu}\|b_k\|^2 + \frac{\eta_k}{2}\|\hat{g}_{k-1}\|^2.$$

After taking expectation and apply the inequality from Lemma 4.7.1, we get

$$\begin{aligned}
\mathbb{E}[f(x_{k-1})] - f^* \leq \mathbb{E}\left[\frac{\eta_k^{-1} - \mu/2}{2}\|x_{k-1} - x^*\|^2 - \frac{\eta_k^{-1}}{2}\|x_k - x^*\|^2\right] \\
+ 4G^{2\alpha}\tau^{2-2\alpha}\mu^{-1} + \eta_k G^\alpha \tau^{2-\alpha}/2.
\end{aligned}$$

Then take $\eta_k = \frac{4}{\mu(k+1)}, \tau_k = Gk^{\frac{1}{\alpha}}$ and multiply both side by k, we get

$$k\mathbb{E}[f(x_{k-1})] - f^* \leq \frac{\mu}{8}\mathbb{E}\big[k(k-1)\|x_{k-1} - x^*\|^2 - k(k+1)\|x_k - x^*\|^2\big] + 8G^2 k^{\frac{2-\alpha}{\alpha}}\mu^{-1}.$$

Notice that $\sum_{k=1}^{K} k^{\frac{2-\alpha}{\alpha}} \leq \int_0^{K+1} k^{\frac{2-\alpha}{\alpha}} dk \leq (K+1)^{2/\alpha}$. Sum over k and we get

$$\sum_{k=1}^{K} k\mathbb{E}[f(x_{k-1})] - f^* \leq \frac{\mu}{8}\mathbb{E}\big[-T(T+1)\|x_T - x^*\|^2\big] + 8G^2(K+1)^{\frac{2}{\alpha}}\mu^{-1}.$$

Devide both side by $\frac{K(K+1)}{2}$ and we get

$$\frac{2}{K(K+1)}\sum_{k=1}^{K} k\mathbb{E}[f(x_{k-1})] - f^* \leq 8G^2K^{-1}(K+1)^{\frac{2-\alpha}{\alpha}}\mu^{-1}.$$

Notice that for $K \geq 1$, $K^{-1} \leq 2(K+1)^{-1}$. We have

$$\frac{2}{K(K+1)}\sum_{k=1}^{K} k\mathbb{E}[f(x_{k-1})] - f^* \leq 16G^2(K+1)^{\frac{2-2\alpha}{\alpha}}\mu^{-1}.$$

The theorem then follows by Jensen's inequality. □

4.7.3 Effect of coordinate-wise moment bound

We now examine how the rates would change if we replace Assumption 4.4.3 with Assumption 4.5.1.

4.7.4 Convergence of GClip (proof of Corollary 4.5.1)

We now look at(GClip) under assumption 4.5.1.

The proof of both the convex and non-convex rates following directly from the following Lemma.

Lemma 4.7.4. *For any $g(x)$ suppose that assumption 4.5.1 with $\alpha \in (1, 2]$. Then suppose we have a constant upper-bound*

$$\mathbb{E}[\|*\|g(x)^\alpha] \leq D\,.$$

Then D satisfies

$$d^{\frac{\alpha}{2}-1}\|B\|_\alpha^\alpha \leq D \leq d^{\alpha/2}\|B\|_\alpha^\alpha.$$

Proof. Note that the function $(\cdot)^{\alpha/2}$ is concave for $\alpha \in (1, 2]$. Using Jensen's inequality we can rewrite as:

$$D \geq \mathbb{E}[\|*\|g(x)^\alpha] = d^{\alpha/2}\mathbb{E}\left[\left(\frac{1}{d}\sum_{i=1}^{d}|g(x)^{(i)}|^2\right)^{\alpha/2}\right] \geq d^{\alpha/2-1}\mathbb{E}\left[\sum_{i=1}^{d}|g(x)^{(i)}|^\alpha\right].$$

Since the right hand-side can be as large as $d^{\frac{\alpha}{2}-1}\|B\|_\alpha^\alpha$, we have our first inequality. On the other hand, we also have an upper bound below:

$$\begin{aligned}\mathbb{E}[\|*\|g(x)^\alpha] &= \mathbb{E}\left[\left(\sum_{i=1}^{d}|g(x)^{(i)}|^2\right)^{\alpha/2}\right] \leq \mathbb{E}\left[\left(d(\max_{i=1}^{d} g(x)^{(i)})^2\right)^{\alpha/2}\right] \\ &\leq \mathbb{E}\left[d^{\alpha/2}(\max_{i=1}^{d} g(x)^{(i)})^\alpha\right] \leq \mathbb{E}\left[d^{\alpha/2}\sum_{i=1}^{d}(g(x)^{(i)})^\alpha\right] \leq d^{\alpha/2}\sum_{i=1}^{d}B_i^\alpha\end{aligned}$$

where $\|B\|_\alpha^\alpha = \sum_{i=1}^{d} B_i^\alpha$. Thus, we have shown that

$$d^{\frac{\alpha}{2}-1}\|B\|_\alpha^\alpha \leq \mathbb{E}[\|*\|g(x)^\alpha] \leq d^{\alpha/2}\|B\|_\alpha^\alpha\,.$$

We know that Jensen's inequality is tight when all the co-ordinates have equal values. This means that if the noise across the coordinates is linearly correlated the lower bound is tighter, whereas the upper bound is tighter if the coordinates depend upon each other in a more complicated manner or are independent of each other. □

Substituting this bound on G in Theorems 4.4.2 and 4.4.1 gives us our corollaries.

4.7.5 Convergence of CClip (Proof of Theorem 4.5.1)

The proof relies on the key lemma which captures the bias-variance trade off under the new noise-assumption and coordinate-wise clipping.

Lemma 4.7.5. *For any $g(x)$ suppose that assumption 4.5.1 with $\alpha \in (1, 2]$ holds. Denote g_i to be i_{th} component of $g(x)$, $\nabla f(x)_i$ to be i_{th} component of $\nabla f(x)$. Then the estimator $\hat{g}(x) = [\hat{g}_1; \cdots; \hat{g}_d]$ from* (CClip) *with clipping parameter $\tau = [\tau_1; \tau_2; \cdots; \tau_d]$ satisfies:*

$$\mathbb{E}\left[\|\hat{g}_i\|^2\right] \leq B_i^{\alpha}\tau_i^{2-\alpha} \text{ and } \|\mathbb{E}[\hat{g}_i] - \nabla f(x)_i\|^2 \leq B_i^{2\alpha}\tau_i^{-2(\alpha-1)}.$$

Proof. Apply Lemma 4.7.1 to the one dimensional case in each coordinate. □

Proof of Theorem 4.5.1. Theorem 4.4.2 For simplicity, we denote $\hat{g}_k = \eta_k \hat{g}(x_k)$ and the bias $b_k = \mathbb{E}[\hat{g}_k] - \nabla f(x_k)$.

$$\begin{aligned}
\|x_k - x^*\|^2 &= \|x_{k-1} - \eta_k \hat{g}_{k-1} - x^*\|^2 \\
&= \|x_{k-1} - x^*\|^2 - 2\eta_k \langle x_{k-1} - x^*, \nabla f(x_{k-1})\rangle \\
&- 2\eta_k \langle x_{k-1} - x^*, b_{k-1}\rangle + \eta_k^2 \|\hat{g}_{k-1}\|^2 \\
&\leq (1 - \mu\eta_k)\|x_{k-1} - x^*\|^2 - 2\eta_k(f(x_{k-1}) - f^*)) \\
&+ 2\eta_k(\frac{\mu}{4}\|x_{k-1} - x^*\|^2 + \frac{4}{\mu}\|*\|{b_k}^2) + \eta_k^2\|\hat{g}_{k-1}\|^2.
\end{aligned}$$

Rearrange and we get

$$f(x_{k-1}) - f^* \leq \frac{\eta_k^{-1} - \mu/2}{2}\|x_{k-1} - x^*\|^2 - \frac{\eta_k^{-1}}{2}\|x_k - x^*\|^2 + \frac{4}{\mu}\|b_k\|^2 + \frac{\eta_k}{2}\|\hat{g}_{k-1}\|^2.$$

After taking expectation and apply the inequality from Lemma 4.7.1, we get

$$\begin{aligned}
\mathbb{E}[f(x_{k-1})] - f^* \leq \mathbb{E}\left[\frac{\eta_k^{-1} - \mu/2}{2}\|x_{k-1} - x^*\|^2 - \frac{\eta_k^{-1}}{2}\|x_k - x^*\|^2\right] \\
+ \textstyle\sum_{i=1}^{d} 4B_i^{2\alpha}\tau_i^{2-2\alpha}\mu^{-1} + \eta_k G^{\alpha}\tau_i^{2-\alpha}/2.
\end{aligned}$$

Then take $\eta_k = \frac{4}{\mu(k+1)}, \tau_i = B_i k^{\frac{1}{\alpha}}$ and multiply both side by k, we get

$$k\mathbb{E}[f(x_{k-1})] - f^* \leq \frac{\mu}{8}\mathbb{E}\big[k(k-1)\|x_{k-1} - x^*\|^2 - k(k+1)\|x_k - x^*\|^2\big] + 8\sum_{i=1}^d B_i^2 k^{\frac{2-\alpha}{\alpha}}\mu^{-1}.$$

Notice that $\sum_{k=1}^K k^{\frac{2-\alpha}{\alpha}} \leq \int_0^{K+1} k^{\frac{2-\alpha}{\alpha}} dk \leq (K+1)^{2/\alpha}$. Sum over k and we get

$$\sum_{k=1}^K k\mathbb{E}[f(x_{k-1})] - f^* \leq \frac{\mu}{8}\mathbb{E}\big[-T(T+1)\|x_T - x^*\|^2\big] + 8\sum_{i=1}^d B_i^2 k^{\frac{2-\alpha}{\alpha}}\mu^{-1}.$$

Devide both side by $\frac{K(K+1)}{2}$ and we get

$$\frac{2}{K(K+1)}\sum_{k=1}^K k\mathbb{E}[f(x_{k-1})] - f^* \leq 8\sum_{i=1}^d B_i^2 k^{\frac{2-\alpha}{\alpha}}\mu^{-1}.$$

Notice that for $K \geq 1$, $K^{-1} \leq 2(K+1)^{-1}$. We have

$$\frac{2}{K(K+1)}\sum_{k=1}^K k\mathbb{E}[f(x_{k-1})] - f^* \leq 16\sum_{i=1}^d B_i^2 k^{\frac{2-\alpha}{\alpha}}\mu^{-1}.$$

The theorem then follows by Jensen's inequality. □

4.7.6 Lower Bound (Proof of Theorem 4.4.3)

We consider the following simple one-dimensional function class parameterized by b:

$$\min_{x\in[0,1/2]} \left\{ f_b(x) = \tfrac{1}{2}(x-b)^2 \right\}, \text{ for } b \in [0, 1/2]\,. \tag{4.2}$$

Also suppose that for $\alpha \in (1, 2]$ and $b \in [0, 1/2]$ the stochastic gradients are of the form:

$$g(x) \sim \nabla f_b(x) + \chi_b\,, \mathbb{E}[g(x)] = \nabla f_b(x)\,, \text{ and } \mathbb{E}[|g(x)|^\alpha] \leq 1\,. \tag{4.3}$$

Note that the function class (4.2) has $\mu = 1$ and optimum value $f_b(b) = 0$, and the α-moment of the noise in (4.3) satisfies $G = B \leq 1$. Thus, we want to prove the following:

Theorem 4.7.6. *For any $\alpha \in (1, 2]$ there exists a distribution χ_b such that the stochastic gradients satisfy* (4.3)*. Further, for any (possibly randomized) algorithm $\mathcal{A}$, define $\mathcal{A}_k(f_b + \chi_b)$ to be the output of the algorithm $\mathcal{A}$ after k queries to the stochastic gradient $g(x)$. Then:*

$$\max_{b \in [0,1/2]} \mathbb{E}[f_b(\mathcal{A}_k(f_b + \chi_b))] \geq \Omega\left(\frac{1}{k^{2(\alpha-1)/\alpha}}\right).$$

Our lower bound construction is inspired by Theorem 2 of **?**. Let $\mathcal{A}_k(f_b+\chi_b)$ denote the output of any possibly randomized algorithm $\mathcal{A}$ after processing k stochastic gradients of the function f_b (with noise drawn i.i.d. from distribution χ_b). Similarly, let $\mathcal{D}_k(f_b + \chi_b)$ denote the output of a *deterministic* algorithm after processing the k stochastic gradients. Then from Yao's minimax principle we know that for any fixed distribution $\mathcal{B}$ over $[0, 1/2]$,

$$\min_{\mathcal{A}} \max_{b \in [0,1/2]} \mathbb{E}_{\mathcal{A}}[\mathbb{E}_{\chi_b} f_b(\mathcal{A}_k(f_b + \chi_b))] \geq \min_{\mathcal{D}} \mathbb{E}_{b \sim \mathcal{B}}[\mathbb{E}_{\chi_b} f_b(\mathcal{D}_k(f_b + \chi_b))].$$

Here we denote $\mathbb{E}_{\mathcal{A}}$ to be expectation over the randomness of the algorithm $\mathcal{A}$ and $\mathbb{E}_{\chi_b}$ to be over the stochasticity of the the noise distribution χ_b. Hence, we only have to analyze deterministic algorithms to establish the lower-bound. Further, since $\mathcal{D}_k$ is deterministic, for any *bijective* transformation h which transforms the stochastic gradients, there exists a deterministic algorithm $\tilde{\mathcal{D}}$ such that $\tilde{\mathcal{D}}_k(h(f_b+\chi_b)) = \mathcal{D}_k(f_b+\chi_b)$. This implies that for any bijective transformation $h(\cdot)$ of the gradients:

$$\min_{\mathcal{D}} \mathbb{E}_{b \sim \mathcal{B}}[\mathbb{E}_{\chi_b} f_b(\mathcal{D}_k(f_b + \chi_b))] = \min_{\mathcal{D}} \mathbb{E}_{b \sim \mathcal{B}}[\mathbb{E}_{\chi_b} f_b(\mathcal{D}_k(h(f_b + \chi_b)))].$$

In this rest of the proof, we will try obtain a lower bound for the right hand side above.

We now describe our construction of the three quantities to be defined: the prob-

lem distribution $\mathcal{B}$, the noise distribution χ_b, and the bijective mapping $h(\cdot)$. All of our definitions are parameterized by $\alpha \in (1, 2]$ (which is given as input) and by $\epsilon \in (0, 1/8]$ (which represents the desired target accuracy). We will pick ϵ to be a fixed constant which depends on the problem parameters (e.g. k) and should be thought of as being small.

- Problem distribution: $\mathcal{B}$ picks $b_0 = 2\epsilon$ or $b_1 = \epsilon$ at random i.e. $\nu \in \{0, 1\}$ is chosen by an unbiased coin toss and then we pick

$$b_\nu = (2 - \nu)\epsilon\,. \tag{4.4}$$

- Noise distribution: Define a constant $\gamma = (4\epsilon)^{1/(\alpha-1)}$ and $p_\nu = (\gamma^\alpha - 2\nu\gamma\epsilon)$. Simple computations verify that $\gamma \in (0, 1/2]$ and that

$$p_\nu = (4\epsilon)^{\frac{\alpha}{\alpha-1}} - 2\nu(4\epsilon^\alpha)^{\frac{1}{\alpha-1}} = (4 - 2\nu)(4\epsilon^\alpha)^{\frac{1}{\alpha-1}} \in (0, 1)\,.$$

  Then, for a given $\nu \in \{0, 1\}$ the stochastic gradient $g(x)$ is defined as

$$g(x) = \begin{cases} x - \frac{1}{2\gamma} & \text{with prob. } p_\nu\,, \\ x & \text{with prob. } 1 - p_\nu\,. \end{cases} \tag{4.5}$$

  To see that we have the correct gradient in expectation verify that

$$\mathbb{E}[g(x)] = x - \frac{p_\nu}{2\gamma} = x - \frac{\gamma^{\alpha-1}}{2} + \nu\epsilon = x - (2 - \nu)\epsilon = x - b_\nu = \nabla f_{b_\nu}(x)\,.$$

  Next to bound the α moment of $g(x)$ we see that

$$\mathbb{E}[|g(x)|^\alpha] \le \gamma^\alpha\left(x - \frac{1}{2\gamma}\right)^\alpha + x^\alpha \le \frac{1}{2} + \frac{1}{2} = 1\,.$$

  The above inequality used the bounds that $\alpha \ge 1$, $x \in [0, 1/2]$, and $\gamma \in (0, 1/2]$. Thus $g(x)$ defined in (4.5) satisfies condition (4.3).

- Bijective mapping: Note that here the only unknown variable is ν which only

affects p_ν. Thus the mapping is bijective as long as the *frequencies* of the events are preserved. Hence given a stochastic gradient $g(x_i)$ the mapping we use is:

$$h(g(x_i)) = \begin{cases} 1 & \text{if } g(x_i) = x_i - \frac{1}{2\gamma}\,, \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Given the definitions above, the output of algorithm $\mathcal{D}_k$ is thus simply a function of k i.i.d. samples drawn from the Bernoulli distribution with parameter p_ν (which is denoted by $\text{Bern}(p_\nu)$). We now show how achieving a small optimization error implies being able to guess the value of ν.

Lemma 4.7.7. *Suppose we are given problem and noise distributions defined as in* (4.4) *and* (4.5)*, and an bijective mapping* $h(\cdot)$ *as in* (4.6)*. Further suppose that there is a deterministic algorithm* $\mathcal{D}_k$ *whose output after processing* k *stochastic gradients satisfies*

$$\mathbb{E}_{b\sim\mathcal{B}}[\mathbb{E}_{\chi_b} f_b(\mathcal{D}_k(h(f_b + \chi_b)))] < \epsilon^2/64\,.$$

Then, there exists a deterministic function $\tilde{\mathcal{D}}_k$ *which given* k *independent samples of* $\text{Bern}(p_\nu)$ *outputs* $\nu' = \tilde{\mathcal{D}}_k(\text{Bern}(p_\nu)) \in \{0, 1\}$ *such that*

$$\mathbb{P}\Big[\tilde{\mathcal{D}}_k(\text{Bern}(p_\nu)) = \nu\Big] \geq \frac{3}{4}\,.$$

Proof. Suppose that we are given access to k samples of $\text{Bern}(p_\nu)$. Use these k samples as the input $h(f_b + \chi_b))$ to the procedure $\mathcal{D}_k$ (this is valid as previously discussed), and let the output of $\mathcal{D}_k$ be $x_k^{(\nu)}$. The assumption in the lemma states that

$$\mathbb{E}_\nu\left[\mathbb{E}_{\chi_b}\left|x_k^{(\nu)} - b_\nu\right|^2\right] < \frac{\epsilon^2}{32}, \text{ which implies that } \mathbb{E}_{\chi_b}\left|x_k^{(\nu)} - b_\nu\right|^2 < \frac{\epsilon^2}{16} \text{ almost surely.}$$

Then, using Markov's inequality (and then taking square-roots on both sides) gives

$$\mathbb{P}\Big[\Big|x_k^{(\nu)} - b_\nu\Big| \geq \frac{\epsilon}{2}\Big] \leq \frac{1}{4}\,.$$

Consider a simple procedure $\tilde{\mathcal{D}}_k$ which outputs $\nu' = 0$ if $x_k^{(\nu)} \geq \frac{3\epsilon}{2}$, and $\nu' = 1$ otherwise. Recall that $|b_0 - b_1| = \epsilon$ with $b_0 = 2\epsilon$ and $b_1 = \epsilon$. With probability $\frac{3}{4}$, $\left|x_k^{(\nu)} - b_\nu\right| < \frac{\epsilon}{2}$ and hence the output ν' is correct. □

Lemma 4.7.7 shows that if the optimization error of $\mathcal{D}_k$ is small, there exists a procedure $\tilde{\mathcal{D}}_k$ which distinguishes between the Bernoulli distributions with parameters p_0 and p_1 using k samples. To argue that the optimization error is large, one simply has to argue that a large number of samples are required to distinguish between $\mathrm{Bern}(p_0)$ and $\mathrm{Bern}(p_1)$.

Lemma 4.7.8. *For any deterministic procedure $\tilde{\mathcal{D}}_k(\mathrm{Bern}(p_\nu))$ which processes k samples of $\mathrm{Bern}(p_\nu)$ and outputs ν'*

$$\mathbb{P}[\nu' = \nu] \leq \frac{1}{2} + \sqrt{k(4\epsilon)^{\frac{\alpha}{\alpha-1}}}\,.$$

Proof. Here it would be convenient to make the dependence on the samples explicit. Denote $\binom{(\nu)}{k} = \left(s_1^{(\nu)}, \ldots, s_k^{(\nu)}\right) \in \{0,1\}^k$ to be the k samples drawn from $\mathrm{Bern}(p_\nu)$ and denote the output as $\nu' = \tilde{\mathcal{D}}(\binom{(\nu)}{k})$. With some slight abuse of notation where we use the same symbols to denote the realization and their distributions, we have:

$$\mathbb{P}\left[\tilde{\mathcal{D}}(\tbinom{(\nu)}{k}) = \nu\right] = \frac{1}{2}\mathbb{P}\left[\tilde{\mathcal{D}}(\tbinom{(1)}{k}) = 1\right] + \frac{1}{2}\mathbb{P}\left[\tilde{\mathcal{D}}(\tbinom{(0)}{k}) = 0\right] = \frac{1}{2} + \frac{1}{2}\mathbb{E}\left[\tilde{\mathcal{D}}(\tbinom{(1)}{k}) - \tilde{\mathcal{D}}(\tbinom{(0)}{k})\right].$$

Next using Pinsker's inequality we can upper bound the right hand side as:

$$\mathbb{E}\left[\tilde{\mathcal{D}}(\tbinom{(1)}{k}) - \tilde{\mathcal{D}}(\tbinom{(0)}{k})\right] \leq |*|\,\tilde{\mathcal{D}}(\tbinom{(1)}{k}) - \tilde{\mathcal{D}}(\tbinom{(0)}{k})_{TV} \leq \sqrt{\frac{1}{2}\,\mathrm{KL}\left(\tilde{\mathcal{D}}\left(\tbinom{(1)}{k}\right), \tilde{\mathcal{D}}\left(\tbinom{(0)}{k}\right)\right)}\,,$$

where $|\cdot|_{TV}$ denotes the total-variation distance and $\mathrm{KL}(\cdot,\cdot)$ denotes the KL-divergence. Recall two properties of KL-divergence: i) for a product measures defined over the same measurable space $(p_1, \ldots, p_k)$ and $(q_1, \ldots, q_k)$,

$$\mathrm{KL}((p_1, \ldots, p_k), (q_1, \ldots, q_k)) = \sum_{i=1}^{k} \mathrm{KL}(p_i, q_i)\,,$$

and ii) for any deterministic function $\tilde{\mathcal{D}}$,

$$\mathrm{KL}(p, q) \geq \mathrm{KL}(\tilde{\mathcal{D}}(p), \tilde{\mathcal{D}}(q)) \,.$$

Thus, we can simplify as

$$\begin{aligned}
\mathbb{P}\Big[\tilde{\mathcal{D}}(\tbinom{(\nu)}{k}) = \nu\Big] &\leq \frac{1}{2} + \sqrt{\frac{k}{8}\,\mathrm{KL}(\mathrm{Bern}(p_1), \mathrm{Bern}(p_0))} \\
&\leq \frac{1}{2} + \sqrt{\frac{k}{8}\frac{(p_0 - p_1)^2}{p_0(1-p_0)}} \\
&\leq \frac{1}{2} + \sqrt{\frac{k(\gamma\epsilon)^2}{4\gamma^\alpha}} \\
&= \frac{1}{2} + \sqrt{k(4^{(2-1/\alpha)}\epsilon)^{\frac{\alpha}{\alpha-1}}} \,.
\end{aligned}$$

Recalling that $\alpha \in (1, 2]$ gives us the statement of the lemma. □

If we pick ϵ to be

$$\epsilon = \frac{1}{16k^{(\alpha-1)/\alpha}} \,,$$

we have that

$$\frac{1}{2} + \sqrt{k(4\epsilon)^{\frac{\alpha}{\alpha-1}}} < \frac{3}{4} \,.$$

Given Lemmas 4.7.7 and 4.7.8, this implies that for the above choice of ϵ,

$$\mathbb{E}_{b\sim\mathcal{B}}[\mathbb{E}_{\chi_b} f_b(\mathcal{D}_k(h(f_b + \chi_b)))] \geq \epsilon^2/64 = \frac{1}{2^{14}k^{2(\alpha-1)/\alpha}} \,.$$

This finishes the proof of the theorem. Note that the readability of the proof was prioritized over optimality and it is possible to obtain significantly better constants. □

4.7.7 Non-convex Lower Bound (Proof of Theorem 4.4.4)

The proof is based on the proof of Theorem 1 in Arjevani et al. [2019]. The only difference is that we assume bounded $\alpha-$moment of the stochastic oracle instead

of bounded variance as in the original proof. We refer readers to Arjevani et al. [2019] for more backgrounds and intuitions. For convenience, we study the stochastic setting ($K = 1$ in Arjevani et al. [2019]) instead of batched setting. We denote a $d-$dimensional vector x as, $x = [x_{(1)}; ...; x_{(d)}]$. Let support$(x)$ denote the set of coordinates where x is nonzero, i.e.

$$\text{support}(x) = \{i \in [d] | x_{(i)} \neq 0\} \subseteq [d].$$

Denote $\text{prog}_\beta(x)$ as the highest index whose entry is $\beta-$far from zero.

$$\text{prog}_\beta(x) = \max\{i \in [d] | |x_{(i)}| > \beta\} \in [d].$$

Note that the function $\text{prog}_\beta(\,\cdot\,)$ is decreasing in β. The function we use to prove the theorem is the same as in **??**. We denote

$$f_d(x) = -\Psi(1)\Phi(x_{(1)}) + \sum_{i=2}^{d} \big(\Psi(-x_{(i-1)})\Phi(-x_{(i)}) - \Psi(x_{(i-1)})\Phi(x_{(i)})\big), \quad \text{where}$$

$$\Psi(x) = \begin{cases} 0, & x \leq 1/2 \\ \exp(1 - \frac{1}{(2x-1)^2}), & x > 1/2 \end{cases}, \Phi(x) = \sqrt{e} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

The above function satisfies the following important properties,

Lemma 4.7.9 (Lemma 2 in Arjevani et al. [2019])**.** *The function f_d satisfies the following properties,*

1. $f_d(0) - \inf_x f_d(x) \leq 12d$.
2. f_d *is* L_0*-smooth, where* $L_0 = 152$.
3. *For all* x, $\|\nabla f_d(x)\|_\infty \leq 23$.
4. *For all* x, $prog_0(\nabla f_d(x)) \leq prog_{\frac{1}{2}}(x) + 1$
5. *For all* x, *if* $prog_1(x) < d$, *then* $\|\nabla f_d(x)\|^2 \geq 1$.

We also define the stochastic oracle $g_d(x)$ as below

$$g_d(x)_{(i)} = \left(1 + \mathbb{1}\Big\{i = \mathrm{prog}_{\frac{1}{4}}(x) + 1\Big\}\left(\frac{z}{p} - 1\right)\right)\frac{\partial}{\partial x_{(i)}} f_d(x)$$

where $z \sim \text{Bernoulli}(p)$. The stochasticity of $g_d(x)$ is only in the $(\mathrm{prog}_{\frac{1}{4}}(x) + 1)$th coordinate. It is easy to see that $g_d(x)$ is a probability-p zero chain as in [**?**, Definition 2] i.e. it satisfies

$$\mathbb{P}\Big(\exists x,\ \text{s.t. } \mathrm{prog}_0(g_d(x)) = \mathrm{prog}_{\frac{1}{4}}(x) + 1\Big) \le p,$$
$$\mathbb{P}\Big(\exists x,\ \text{s.t. } \mathrm{prog}_0(g_d(x)) > \mathrm{prog}_{\frac{1}{4}}(x) + 1\Big) = 0.$$

The second claim is because $\mathrm{prog}_\beta(\,\cdot\,)$ is decreasing in β and

$$\mathrm{prog}_{\frac{1}{4}}(\nabla f_d(x)) \le \mathrm{prog}_0(\nabla f_d(x)) \le \mathrm{prog}_{\frac{1}{2}}(x) + 1 \le \mathrm{prog}_{\frac{1}{4}}(x) + 1\,.$$

The first claim is because if $z = 0$, then we explicitly set the $(\mathrm{prog}_{\frac{1}{4}}(x) + 1)$th coordinate to 0. The stochastic gradient additionally has bounded α-moment as we next show.

Lemma 4.7.10. *The stochastic oracle above is an unbiased estimator of the true gradient, and for any $\alpha \in (1, 2]$*

$$\mathbb{E}[\|g_d(x)\|^\alpha] \le 2\|\nabla f_d(x)\|^\alpha + 23^\alpha \frac{2}{p^{\alpha-1}}.$$

Proof. The unbiased-ness is easy to verify. For the bounded α-moment, observe that only the $(\mathrm{prog}_{\frac{1}{4}} + 1)$-th coordinate is noisy and differs by a factor of $(\frac{z}{p} - 1)$. Hence,

we have

$$\begin{aligned}\mathbb{E}[\|g_d(x)\|^\alpha] &\le 2\|\nabla f_d(x)\|^\alpha + 2\mathbb{E}[\|g_d(x) - \nabla f_d(x)\|^\alpha] \\ &\le 2\|\nabla f_d(x)\|^\alpha + \|\nabla f_d(x)\|_\infty^\alpha \mathbb{E}\left[|\frac{z}{p} - 1|^\alpha\right] \\ &\le 2\|\nabla f_d(x)\|^\alpha + \|\nabla f_d(x)\|_\infty^\alpha \frac{p(1-p)^\alpha + (1-p)p^\alpha}{p^\alpha} \\ &\le 2\|\nabla f_d(x)\|^\alpha + 2 3^\alpha \frac{2}{p^{\alpha-1}}\end{aligned}$$

The first inequality followed from Jensen's inequality and the convexity of $\|\cdot\|^\alpha$ for $\alpha \in (1, 2]$:

$$\|u+v\|^\alpha \le 4\|\tfrac{u+v}{2}\|^\alpha \le 2(\|u\|^\alpha + \|v\|^\alpha) \text{ for any } u, v\,.$$

□

Now we are ready to prove Theorem 4.4.4. Given accuracy parameter ϵ, suboptimality $\Delta = f(0) - f^*$, smoothness constant L, and bounded $\alpha-$moment G^α, we define

$$f(x) = \frac{L\lambda^2}{152} f_d(\frac{x}{\lambda}),$$

where $\lambda = \frac{304\epsilon}{L}$ and $d = \lfloor \frac{\Delta L}{7296\epsilon^2} \rfloor$. Then,

$$g(x) = \frac{L\lambda}{152} g_d(x/\lambda) = 2\epsilon g_d(x/\lambda).$$

Using Lemma 4.7.10, we have

$$\mathbb{E}[\|g(x)\|^\alpha] \le 8\epsilon^\alpha \|\nabla f_d(x)\|^\alpha + \frac{5000\epsilon^\alpha}{p^{\alpha-1}}$$

When $G \ge 4\sqrt{\Delta L}$, we can set $p = \frac{(5000\epsilon)^{\frac{\alpha}{\alpha-1}}}{(G-4\sqrt{\Delta L})^{\frac{\alpha}{\alpha-1}}}$ and get $\mathbb{E}[\|g(x)\|^\alpha] \le G^\alpha$.

Let x_k be the output of any *zero-respecting* algorithm $\mathcal{A}$. By [**?**, Lemma 1], we know that with probability at least $1/2$, $\text{prog}_1(x_k) \le \text{prog}_0(x_k) < d$ for all $k \le \frac{(d-1)}{2p}$.

Now applying Lemma 4.7.9.5, we have that for all $k \leq \frac{(d-1)}{2p}$:

$$\mathbb{E}[\|\nabla f(x_k)\|] \geq \frac{1}{2}\frac{L\lambda}{152}\mathbb{E}[\nVdash\|\nabla f_d(x_k/\lambda)\| \mid \{\text{prog}_1(x_k) < d\}] \geq \epsilon\,.$$

Therefore, $\mathbb{E}\|\nabla f(x_k)\| \geq \epsilon$, for all $k \leq \frac{(d-1)}{2p} = \frac{(G-4\sqrt{\Delta L})^{\frac{\alpha}{\alpha-1}}\Delta L}{7296\times 5000^{\frac{\alpha}{\alpha-1}}\epsilon^{2+\frac{\alpha}{\alpha-1}}} = c(\alpha)(G - 4\sqrt{\Delta L})^{\frac{\alpha}{\alpha-1}}\Delta L\epsilon^{-\frac{3\alpha-2}{\alpha-1}}$. By eliminating ϵ, we can rewrite this in terms of k. Finally, the techniques from [**?**, Theorem 3] show how to lift lower-bounds for *zero-respecting* algorithms to any randomized method.

Chapter 5

Non-differentiable, nonconvex Optimization

The definition of gradient requires the function of interest to be differentiable. However, as we discussed in Chapter 2,"gradient" methods can be generalized to subgradient methods in nondifferentiable convex problems and still achieve reasonable convergence rates. One natural question following this observation could be: *can convergence analysis also be done for nonconvex nondifferentiable functions?*

Studying this problem is also interesting from deep learning perspective, as many neural networks adopt ReLU as the activation function, and hence the training objective is nondifferentiable. Motivated by this, we study the effect of non-differentiablility on complexity analysis. We provide the first non-asymptotic analysis [Zhang et al., 2020b] 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. We will also discuss how this analysis could relate to the gap between optimization analysis and deep learning experiments.

5.1 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 5.1 based on the references [Nesterov, 2018, Carmon et al., 2017, Arjevani et al., 2019].

Table 5.1: When the problem is nonconvex and nonsmooth, finding a ϵ-stationary point is intractable, see Theorem 5.5.2. 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	$\mathcal{O}(\epsilon^{-2})$	$\tilde{\mathcal{O}}(\epsilon^{-4}\delta^{-1})$

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 and 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.

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. As we will see later.

5.1.1 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 and 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 and Ruan [2018], Drusvyatskiy and Paquette [2019], Davis and Drusvyatskiy [2019] consider the composition structure $f \circ g$ of convex and smooth functions; Bolte et al. [2018], Zhang et al. [2018], Beck and 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., $-|x|$.

5.2 Preliminaries

In this section, we set up the notion of generalized directional derivatives that will play a central role in our analysis. Throughout the chapter, we assume that the nonsmooth function f is L-Lipschitz continuous (more precise assumptions on the function class are outlined in Section 5.2.3).

5.2.1 Generalized gradients

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

Definition 5.2.1. 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 5.2.2. The ***generalized gradient*** of f is defined as

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

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

Proposition 5.2.1 (Properties of generalized gradients)**.**

1. *$\partial f(x)$ is a nonempty, convex compact set. For all vectors $g \in \partial f(x)$, we have $\|g\| \leq L$.*
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

$$\partial f(x) = \operatorname{conv}\big(\{g | g = \lim_{k\to\infty} \nabla f(x_k),\ x_k \to x\}\big).$$

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 $f(y) - f(z) = \langle g, y - z\rangle$.*

5.2.2 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 5.2.3. 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:

$$f'(x; d) = \lim_{t\to 0^+} \tfrac{1}{t}(f(\varphi(t)) - f(x)). \tag{5.1}$$

In the rest of the chapter, 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 5.2.1 (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 5.2.2. *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 5.2.3. *Assume that the directional derivative exists. For any x, d, there exists $g \in \partial f(x)$ s.t. $\langle g, d\rangle = f'(x; d)$.*

5.2.3 Nonsmooth function class of interest

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

$$\begin{aligned}\mathcal{F}(\Delta, L) := \{f | f \text{ is } L\text{-Lipschitz};& \\ & f \text{ is directionally differentiable}; \\ & f(x_0) - \inf_x f(x) \leq \Delta\},\end{aligned} \tag{5.2}$$

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

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

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

5.3 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.

5.3.1 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

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

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 5.3.1. *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 5.3.1 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 5.3.1. 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 5.3.1 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 5.3.1. *The following statements hold:*

1. *ϵ-stationarity implies (δ, ϵ)-stationarity for any $\delta \geq 0$.*
2. *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 5.5.2 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 5.3.2. *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 5.3.2 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.

5.3.2 Gradient Oracles

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

Assumption 5.3.1. 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

1. In the **deterministic** setting, the oracle returns

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

2. 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 5.2.1. This property justifies our choice of oracles.

5.3.3 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

$$x_{k+1} = A^{(k)}(x_1, g_1, f_1, x_2, g_2, f_2, ..., x_k, g_k, f_k), \tag{5.3}$$

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

$$\begin{aligned} &T_{\delta,\epsilon}(\{x_t\}_{t\in\mathbb{N}}, f) := \\ &\quad \inf\big\{t \in \mathbb{N} \;\mid\; \text{Prob}\{d(0, \partial f(x+\delta B)) \geq \epsilon \\ &\qquad \text{for all } k \leq t\} \leq \tfrac{1}{3}\big\}. \end{aligned} \tag{5.4}$$

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

$$\mathcal{N}(\mathcal{A}, \mathcal{F}, \epsilon, \delta) := \inf_{A\in\mathcal{A}} \sup_{f\in\mathcal{F}} T_{\delta,\epsilon}(A[f, x_0], f). \tag{5.5}$$

At a high level, (5.5) 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 $\mathcal{F}$.

5.4 Deterministic Setting

For optimizing L-smooth functions, a crucial inequality is

$$f\big(x - \tfrac{1}{L}\nabla f(x)\big) - f(x) \leq -\tfrac{1}{2L}\|\nabla f(x)\|^2. \tag{5.6}$$

In other words, either the gradient is small or the function value decreases sufficiently along the negative gradient. However, when the objective function is non-smooth, 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 (5.6), 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 (5.6) 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.

Theorem 5.4.1. *In the deterministic setting and with Assumption 5.3.1(a), the Ingd*

Algorithm 2 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: Return x_t such that $\|m_{t,K}\| \leq \epsilon$

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 5.7.4.

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 5.4.1. 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. 5.3.1), 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.

5.5 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.

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 AdaGrad or ADAM to prevent instability. Here, we show that the constant q allows us to control the variance of

Algorithm 3 Stochastic Ingd (x_1, p, q, β, T, K)

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

$x_{t+1} - x_t$. 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 5.5.1. *In the stochastic setting, with Assumption 5.3.1(b), the Stochastic-*

Ingd algorithm (Algorithm 3) 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* $\tilde{\mathcal{O}}\left(\frac{G^3\Delta}{\epsilon^4\delta}\right)$.

For readability, the constants in Theorem 5.5.1 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 5.3.1.

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 5.7.5.

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 5.5.2 (Lower bound on δ dependence)**.** *Let $\mathcal{A}$ denote the class of algorithms defined in Section 5.3.2 and $\mathcal{F}(\Delta, L)$ denote the class of functions defined in Equation (5.2). 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 5.7.6.

This theorem also implies the negative result for finite time analyses that we showed in Theorem 5.3.1. 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.

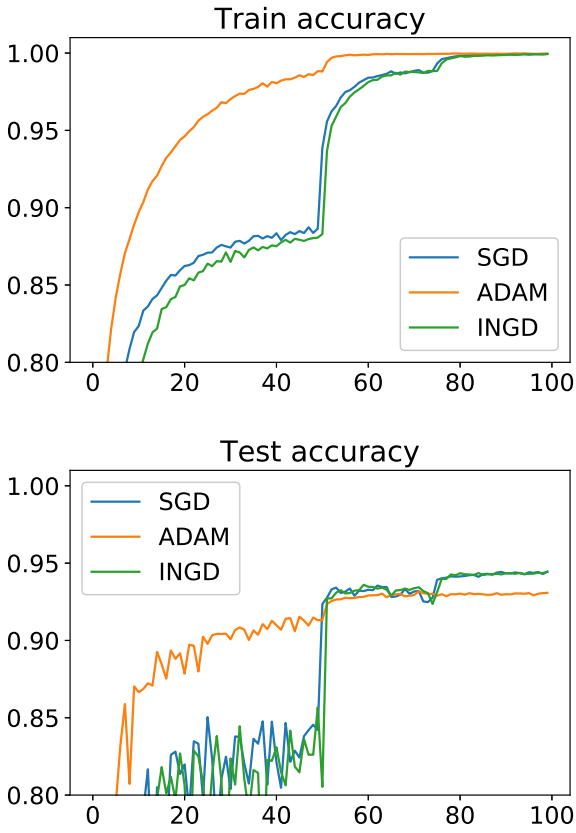


Figure 5-1: Learning curve of SGD, ADAM and Ingd on training ResNet 20 on CIFAR10.

5.6 Experiments

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 and 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 and Ba [2014]. 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 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 5-1. 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.

5.7 Proofs

5.7.1 Proof of Lemma 5.2.2

Proof. Let $g(t) = f(x + t(y - x))$ for $t \in [0, 1]$, then g is $L\|y - x\|$-Lipschitz implying that g is absolutely continuous. Thus from the fundamental theorem of calculus (Lebesgue), g has a derivative g' almost everywhere, and the derivative is Lebesgue integrable such that

$$g(t) = g(0) + \int_0^t g'(s)ds.$$

Moreover, if g is differentiable at t, then

$$g'(t) = \lim_{\delta t \to 0} \frac{g(t + \delta t) - g(t)}{\delta t} = \lim_{\delta t \to 0} \frac{f(x + (t + \delta t)(y - x)) - f(x + t(y - x))}{\delta t}$$

$$= f'(x + t(y - x), y - x).$$

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

Since this equality holds almost everywhere, we have

$$f(y) - f(x) = g(1) - g(0) = \int_0^1 g'(t)dt = \int_0^1 f'(x + t(y - x), y - x)dt.$$

□

5.7.2 Proof of Lemma 5.2.3

Proof. For any $\varphi(t) = x + td$ as given in Definition 5.2.3, let $t_k \to 0$. Denote $x_k = \varphi(t_k), \delta_k = \|x_k - x\| \to 0$. By Proposition 1.6, we know that there exists $g_{k,j} \in \cup_{y \in x + \delta_k B} \partial f(y)$ such that

$$f(x_k) - f(x) = \langle g_{k,j}, x_k - x \rangle.$$

By the existence of directional derivative, we know that

$$\lim_{k \to \infty} \langle g_{k,j}, d \rangle = \lim_{k \to \infty} \frac{\langle g_{k,j}, t_k d \rangle}{t_k} = f'(x, d),$$

and $g_{k,j}$ is in a bounded set with norm less than L. The Lemma follows by the fact that any accumulation point of $g_{k,j}$ is in $\partial f(x)$ due to upper-semicontinuity of $\partial f(x)$. □

5.7.3 Proof of Lemmas in Algorithm Complexity

Proof of Theorem 5.3.1

Our proof strategy is similar to Theorem 1.1.2 in Nesterov [2013], where we use the resisting strategy to prove lower bound. Given a one dimensional function f, let $x_k, k \in [1, K]$ be the sequence of points queried in ascending order instead of query order. We assume without loss of generality that the initial point is queried and is an element of $\{x_k\}_{k=0}^K$ (otherwise, query the initial point first before proceeding with the algorithm).

Then we define the resisting strategy: always return

$$f(x) = 0, \text{and} \quad \nabla f(x) = L,$$

If we can prove that for any set of points $x_k, k \in [1, K]$, there exists two functions such that they satisfy the resisting strategy $f(x_k) = 0, \text{and} \quad \nabla f(x_k) = L, k \in [1, K]$, and that the two functions do not share any common stationary points, then we know no randomized/deterministic can return an $\epsilon-$stationary points with probability more than $1/2$ for both functions simultaneously. In other word, no algorithm that query K points can distinguish these two functions. Hence we proved the theorem following the definition of complexity in (5.5) with $\delta = 0$.

All we need to do is to show that such two functions exist in the Lemma below.

Lemma 5.7.1. *Given a finite sequence of real numbers $\{x_k\}_{k\in[1,K]} \in \mathbb{R}$, there is a family of functions $f_\theta \in \mathcal{F}(\Delta, L)$ such that for any $k \in [1, K]$,*

$$f_\theta(x_k) = 0 \quad \textit{and} \quad \nabla f_\theta(x_k) = L$$

and for ϵ sufficiently small, the set of ϵ-stationary points of f_θ are all disjoint, i.e $\{\epsilon$-stationary points of $f_{\theta_1}\} \cap \{\epsilon$-stationary points of $f_{\theta_2}\} = \emptyset$ for any $\theta_1 \neq \theta_2$.

Proof. Up to a permutation of the indices, we could reorder the sequence in the increasing order. WLOG, we assume x_k is increasing. Let $\delta = \min\{\min_{x_i \neq x_j}\{|x_i - x_j|\}, \frac{\Delta}{L}\}$. For any $0 < \theta < 1/2$, we define f_θ by

$$\begin{aligned}
&f_\theta(x) = -L(x - x_1 + 2\theta\delta) \quad \text{for} \quad x \in (-\infty, x_1 - \theta\delta] \\
&f_\theta(x) = L(x - x_k) \quad \text{for} \quad x \in \left[x_k - \theta\delta, \frac{x_k + x_{k+1}}{2} - \theta\delta\right] \\
&f_\theta(x) = -L(x - x_{k+1} + 2\theta\delta) \quad \text{for} \quad x \in \left[\frac{x_k + x_{k+1}}{2} - \theta\delta, x_{k+1} - \theta\delta\right] \\
&f_\theta(x) = L(x - x_K) \quad x \in [x_K + \theta\delta, +\infty).
\end{aligned}$$

It is clear that f_θ is directional differentiable at all point and $\nabla f_\theta(x_k) = L$. Moreover,

the minimum $f_\theta^* = -L\theta\delta \geq -\Delta$. This implies that $f_\theta \in \mathcal{F}(\Delta, L)$. Note that $\nabla f_\theta = L$ or $-L$ except at the local extremum. Therefore, for any $\epsilon < L$ the set of ϵ-stationary points of f_θ are exactly

$$\{\epsilon\text{-stationary points of } f_\theta\} = \{x_k - \theta\delta \mid k \in [1, K]\} \cup \left\{ \frac{x_k + x_{k+1}}{2} - \theta\delta \mid k \in [1, K-1] \right\},$$

which is clearly distinct for different choice of θ. □

Proof of Proposition 5.3.1

Proof. When x is $(\frac{\epsilon}{3L}, \frac{\epsilon}{3})$ stationary, we have $d(0, \partial f(x + \frac{\epsilon}{3L}B)) \leq \frac{\epsilon}{3}$. By definition, we could find $g \in \text{conv}(\cup_{y \in x + \frac{\epsilon}{3L}B} \nabla f(y))$ such that $\|g\| \leq 2\epsilon/3$. This means, there exists $x_1, \cdots, x_k \in x + \frac{\epsilon}{3L}B$, and $\alpha_1, \cdots, \alpha_k \in [0, 1]$ such that $\alpha_1 + \cdots + \alpha_k = 1$ and

$$g = \sum_{i=1}^{k} \alpha_i \nabla f(x_i)$$

Therefore

$$\begin{aligned}
\|\nabla f(x)\| &\leq \|g\| + \|\nabla f(x) - g\| \\
&\leq \frac{2\epsilon}{3} + \sum_{i=1}^{k} \alpha_i \|\nabla f(x) - \nabla f(x_k)\| \\
&\leq \frac{2\epsilon}{3} + \sum_{i=1}^{k} \alpha_i L \|x - x_k\| \\
&\leq \frac{2\epsilon}{3} + \sum_{i=1}^{k} \alpha_i L \frac{\epsilon}{3L} = \epsilon.
\end{aligned}$$

Therefore, x is an ϵ-stationary point in the standard sense. □

Proof of Lemma 5.3.2

Proof. First, we show that the limit exists. By Lipschitzness and Jensen inequality, we know that $\partial f(x + \delta_{k+1}B)$ lies in a bounded ball with radius L. For any sequence of $\{\delta_k\}$ with $\delta_k \downarrow 0$, we know that $\partial f(x + \delta_{k+1}B) \subseteq \partial f(x + \delta_k B)$. Therefore, the limit

exists by the monotone convergence theorem.

Next, we show that $\lim_{\delta \downarrow 0} \partial f(x + \delta B) = \partial f(x)$. For one direction, we show that $\partial f(x) \subseteq \lim_{\delta \downarrow 0} \partial f(x + \delta B)$. This follows by Proposition 1.5 and the fact that

$$\cup_{y \in x + \delta B} \partial f(y) \subseteq \text{conv}(\cup_{y \in x + \delta B} \partial f(y)) = \partial f(x + \delta B).$$

Next, we show the other direction $\lim_{\delta \downarrow 0} \partial f(x + \delta B) \subseteq \partial f(x)$. By upper semicontinuity, we know that for any $\epsilon > 0$, there exists $\delta > 0$ such that

$$\cup_{y \in x + \delta B} \partial f(y) \subseteq \partial f(x) + \epsilon B.$$

Then by convexity of $\partial f(x)$ and ϵB, we know that their Minkowski sum $\partial f(x) + \epsilon B$ is convex. Therefore, we conclude that for any $\epsilon > 0$, there exists $\delta > 0$ such that

$$\partial f(x + \delta B) = \text{conv}(\cup_{y \in x + \delta B} \partial f(y)) \subseteq \partial f(x) + \epsilon B.$$

□

5.7.4 Proof of Theorem 5.4.1

Before we prove the theorem, we first analyze how many times the algorithm iterates in the while loop.

Lemma 5.7.2. *Let $K = \frac{48L^2}{\epsilon^2}$. Given $t \in [1, T]$,*

$$\mathbb{E}[\|m_{t,K}\|^2] \leq \frac{\epsilon^2}{16},$$

where for convenience of analysis, we define $m_{t,k} = 0$ for all $k > k_0$ if the k-loop breaks at (t, k_0). Consequently, for any $\gamma < 1$, with probability $1 - \gamma$, there are at most $\log(1/\gamma)$ restarts of the while loop at the t-th iteration.

Proof. Let $\mathfrak{F}_{t,k} = \sigma(y_{t,1}, \cdots, y_{t,k+1})$, then $x_{t,k}, m_{t,k} \in \mathfrak{F}_{t,k}$. We denote $D_{t,k}$ as the event

that k-loop does not break at $x_{t,k}$, i.e. $\|m_{t,k}\| > \epsilon$ and $f(x_{t,k}) - f(x_t) > -\frac{\delta\|m_{t,k}\|}{4}$. It is clear that $D_{t,k} \in \mathfrak{F}_{t,k}$.

Let $\gamma(\lambda) = (1-\lambda)x_t + \lambda x_{t,k},\ \lambda \in [0,1]$. Note that $\gamma'(\lambda) = x_{t,k} - x_t = -\delta\frac{m_{t,k}}{\|m_{t,k}\|}$. Since $y_{t,k+1}$ is uniformly sampled from line segment $[x_t, x_{t,k}]$, we know

$$\mathbb{E}[\langle g_{t,k+1}, x_{t,k} - x_t\rangle|\mathfrak{F}_{t,k}] = \int_0^1 f'(\gamma(t), x_{t,k} - x_t)dt = f(x_{t,k}) - f(x_t),$$

where the second equality comes from directional differentiability. Since $x_{k+1} - x_k = -\delta\frac{m_{t,k}}{\|m_{t,k}\|}$, we know that

$$\mathbb{E}[\langle g_{t,k+1}, m_{t,k}\rangle|\mathfrak{F}_{t,k}] = -\frac{\|m_{t,k}\|}{\delta}(f(x_{t,k}) - f(x_t)). \tag{5.7}$$

By construction $m_{t,k+1} = \beta m_{t,k} + (1-\beta)g_{t,k+1}$ under $D_{t,k} \cap \cdots \cap D_{t,1}$, and $m_{t,k+1} = 0$ otherwise. Therefore,

$$\begin{aligned}
&\mathbb{E}[\|m_{t,k+1}\|^2|\mathfrak{F}_{t,k}] \\
=&\mathbb{E}[\|\beta m_{t,k} + (1-\beta)g_{t,k+1}\|^2 \mathbb{1}_{\{D_{t,k}\cap\cdots\cap D_{t,1}\}}|\mathfrak{F}_{t,k}] \\
\leq& \left(\beta^2\|m_{t,k}\|^2 + (1-\beta)^2L^2 + 2\beta(1-\beta)\mathbb{E}[\langle g_{t,k+1}, m_{t,k}\rangle|\mathfrak{F}_{t,k}]\right)\mathbb{1}_{\{D_{t,k}\cap\cdots\cap D_{t,1}\}} \\
\leq&\beta^2\|m_{t,k}\|^2 + (1-\beta)^2L^2 - 2\beta(1-\beta)\frac{\|m_{t,k}\|}{\delta}(f(x_{t,k}) - f(x_t))\mathbb{1}_{\{D_{t,k}\cap\cdots\cap D_{t,1}\}} \\
\leq&\beta^2\|m_{t,k}\|^2 + (1-\beta)^2L^2 + 2\beta(1-\beta)\frac{\|m_{t,k}\|^2}{4},
\end{aligned}$$

where in the third line, we use the fact $\beta, D_{t,k}\cap\cdots\cap D_{t,1} \in \mathfrak{F}_{t,k}$; in the fourth line we use the fact under $D_{t,k}$, $f(x_{t,k}) - f(x_t) \geq -\frac{\delta\|m_{t,k}\|}{4}$. The last equation is a quadratic function with respect to β, which could be rewritten as

$$h(\beta) = \beta^2(\frac{\|m_{t,k}\|^2}{2} + L^2) - 2\beta(L^2 - \frac{\|m_{t,k}\|^2}{4}) + L^2.$$

It achieves the minimum at $\beta = \frac{4L^2 - \|m_{t,k}\|^2}{4L^2 + 2\|m_{t,k}\|^2}$, which belongs to $\mathfrak{F}_{t,k}$. Since $\|m_{t,k}\| \leq L$, we have

$$h^* = \frac{L^2}{L^2 + \frac{\|m_{t,k}\|^2}{2}}\|m_{t,k}\|^2 \leq \left(1 - \frac{\|m_{t,k}\|^2}{3L^2}\right)\|m_{t,k}\|^2.$$

Therefore,

$$\begin{aligned}
&\mathbb{E}[\|m_{t,k+1}\|^2] \\
=&\mathbb{E}[\mathbb{E}[\|m_{t,k+1}\|^2|\mathfrak{F}_{t,k}]] \\
\leq&\mathbb{E}\left[\left(1-\frac{\|m_{t,k}\|^2}{3L^2}\right)\|m_{t,k}\|^2\right] \\
\leq&\left(1-\frac{\mathbb{E}[\|m_{t,k}\|^2]}{3L^2}\right)\mathbb{E}[\|m_{t,k}\|^2],
\end{aligned}$$

where the last inequality follows from Jensen's inequality under the fact that the function $x \to (1-x/3L^2)x$ is concave. Now consider the sequence $v_k = \mathbb{E}[\|m_{t,k}\|^2]/L^2 \in [0,1]$, we get

$$v_{k+1} \leq v_k - v_k^2/3 \quad \Longrightarrow \quad \frac{1}{v_{k+1}} \geq \frac{1}{v_k - v_k^2/3} \geq \frac{1}{v_k} + \frac{1}{3}.$$

Knowing that $v_1 \leq 1$, we therefore have

$$v_k \leq \frac{3}{k+2}.$$

When $K > \frac{48L^2}{\epsilon^2}$, we have $\mathbb{E}[\|m_{t,K}\|^2] \leq \frac{\epsilon^2}{16}$. Therefore, by Markov inequality, $\mathcal{P}\{\|m_{t,K}\| \geq \epsilon\} \leq 1/4$. In other words, the while-loop restarts with probability at most $1/4$. Therefore, with probability $1-\gamma$, there are at most $\log(1/\gamma)$ restarts. □

Now we are ready to prove the main theorem.

Proof of Theorem 5.4.1. We notice that $m_{t,k}$ is always a convex combinations of generalized gradients within the δ ball of x_k, i.e.

$$m_{t,k} \in \partial f(x_t + \delta B) = \text{conv}(\cup_{y \in x_t + \delta B} \partial f(y)).$$

Therefore, if at any t, k, $\|m_{t,k}\| \leq \epsilon$, then the corresponding x_t is a (δ, ϵ) approximate stationary point. To show that our algorithm always find a $\|m_{t,k}\| \leq \epsilon$, we need to control the number of times the descent condition is satisfied, which breaks the while-loop without satisfying $\|m_{t,k}\| \leq \epsilon$. Indeed, when the descent condition holds,

we have

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

where we use the fact $\|m_{t,k}\| > \epsilon$, otherwise, the algorithm already terminates. Consequently, there are at most $\frac{4\Delta}{\delta\epsilon} - 1 = T - 1$ iterations that the descent condition holds. As a result, for at least one t , the while-loop ends providing a (δ, ϵ) approximate stationary point.

By Lemma 5.7.2, we know that with probability $1 - \frac{\gamma\delta\epsilon}{4\Delta}$, the t-th iteration terminates in $\log(\frac{4\Delta}{\gamma\delta\epsilon})$ restarts. Consequently, with probability $1-\gamma$, the algorithm returns a (δ, ϵ) approximate stationary point using

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

□

5.7.5 Proof of Theorem 5.5.1

Stochastic INGD has convergence guarantee as stated in the next theorem.

Theorem 5.7.3. *Under the stochastic Assumption 5.3.1, the Stochastic INGD algorithm in Algorithm 3 with parameters* $\beta = 1 - \frac{\epsilon^2}{64G^2}$, $p = \frac{64G^2 \ln(16G/\epsilon)}{\delta\epsilon^2}$, $q = 4Gp$, $T = \frac{2^{16}G^3\Delta \ ln(16G/\epsilon)}{\epsilon^4\delta} \max\{1, \frac{G\delta}{8\Delta}\}$, $K = p\delta$ *has algorithm complexity upper bounded by*

$$\frac{2^{16}G^3\Delta \ ln(16G/\epsilon)}{\epsilon^4\delta} \max\{1, \frac{G\delta}{8\Delta}\} = \tilde{O}\left(\frac{G^3\Delta}{\epsilon^4\delta}\right).$$

Proof. First, we are going to show that

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|m_t\|] \leq \epsilon/4. \tag{5.8}$$

From construction of the descent direction, we have

$$\|m_{t+1}\|^2 = (1-\beta)^2\|g(y_{t+1})\|^2 + 2\beta(1-\beta)\langle g(y_{t+1}), m_t\rangle + \beta^2\|m_t\|^2. \tag{5.9}$$

Multiply both side by η_t and sum over t, we get

$$0 = (1-\beta)^2 \underbrace{\sum_{t=1}^{T} \eta_t \|g(y_{t+1})\|^2}_{\text{i}} + 2\beta(1-\beta) \underbrace{\sum_{t=1}^{T} \langle g(y_{t+1}), \eta_t m_t \rangle}_{\text{ii}} + \underbrace{\sum_{t=1}^{T} \eta_t (-\|m_{t+1}\|^2 + \beta^2 \|m_t\|^2)}_{\text{iii}}. \tag{5.10}$$

We remark that at each iteration, we have two randomized/stochastic procedure: first we draw y_{t+1} randomly between the segment $[x_t, x_{t+1}]$, second we draw a stochastic gradient at y_{t+1}. For convenience of analysis, we denote $\mathcal{G}_t$ as the sigma field generated by $g(y_t)$, and $\mathcal{Y}_t$ as the sigma field generated by y_t. Clearly, $\mathcal{G}_t \subset \mathcal{Y}_{t+1} \subset \mathcal{G}_{t+1}$. By definition η_t is determined by m_t, which is further determined by g_t. Hence, the vectors m_t, η_t and x_{t+1} are $\mathcal{G}_t$-measurable.

Now we analyze each term one by one.

Term i: This term can be easily bound by

$$\mathbb{E}[\eta_t \|g(y_{t+1})\|^2] \le \frac{1}{q}\mathbb{E}[\|g(y_{t+1})\|^2] = \frac{1}{q}\mathbb{E}[\mathbb{E}[\|g(y_{t+1})\|^2 | \mathbf{Y}_{t+1}]] \le \frac{G^2}{q}. \tag{5.11}$$

Term ii: Note that $\eta_t m_t = x_t - x_{t+1}$, we have

$$\begin{aligned}
\mathbb{E}[\langle g(y_{t+1}), \eta_t m_t \rangle \mid \mathcal{G}_t] &= \mathbb{E}[\mathbb{E}[\langle g(y_{t+1}), x_t - x_{t+1} \rangle \mid \mathcal{Y}_{t+1}] | \ \mathcal{G}_t] \\
&= \mathbb{E}[f'(y_{t+1}; x_t - x_{t+1}) | \ \mathcal{G}_t] \\
&= \int_{[0,1]} f'(x_{t+1} + \lambda(x_t - x_{t+1}); x_t - x_{t+1}) d\lambda \\
&= f(x_t) - f(x_{t+1}),
\end{aligned}$$

where the second line we use the property of the oracle given in Assumption 1(b). Thus by taking the expectation, we have

$$\sum_{t=1}^{T} \mathbb{E}[\langle g(y_{t+1}), \eta_t m_t \rangle] = \mathbb{E}[f(x_1) - f(x_{T+1})] \le \Delta.$$

Term iii: we would like to develop a telescopic sum for the third term, however

this is non-trivial since the stepsize η_t is adaptive. Extensive algebraic manipulation is involved.

$$\begin{aligned}
&\sum_{t=1}^{T} \eta_t(-\|m_{t+1}\|^2 + \beta^2\|m_t\|^2) \\
=&\sum_{t=1}^{T} \frac{-\|m_{t+1}\|^2}{p\|m_t\| + q} + \beta^2 \sum_{t=1}^{T} \frac{\|m_t\|^2}{p\|m_t\| + q} \\
=&\sum_{t=1}^{T} \left(\frac{-\|m_{t+1}\|^2}{p\|m_t\| + q} + \frac{\|m_{t+1}\|^2}{p\|m_{t+1}\| + q} \right) - \sum_{t=1}^{T} \frac{\|m_{t+1}\|^2}{p\|m_{t+1}\| + q} + \beta^2 \sum_{t=1}^{T} \frac{\|m_t\|^2}{p\|m_t\| + q} \\
=&\sum_{t=1}^{T} \frac{p\|m_{t+1}\|^2(\|m_t\| - \|m_{t+1}\|)}{(p\|m_t\| + q)(p\|m_{t+1}\| + q)} + \beta^2 \frac{\|m_1\|^2}{p\|m_1\| + q} + (\beta^2 - 1) \sum_{t=2}^{T+1} \frac{\|m_t\|^2}{p\|m_t\| + q}. \quad (5.12)
\end{aligned}$$

The first equality follows by $\eta_t = \frac{1}{p\|m_t\|+q}$. The second equality subtract and add the same terms $\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}$. The last equality regroups the terms. We now prove the first term in (5.12) admits the following upper bound:

$$\frac{p\|m_{t+1}\|^2(\|m_t\| - \|m_{t+1}\|)}{(p\|m_t\| + q)(p\|m_{t+1}\| + q)} \leq (1-\beta)\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\| + q} + \frac{(1-\beta)p\|g(y_{t+1})\|}{q} \frac{\|m_t\|^2}{p\|m_t\| + q}. \quad (5.13)$$

Note that if $\|m_{t+1}\| \geq \|m_t\|$ then the inequality trivially holds. Thus, we only need to consider the case when $\|m_{t+1}\| \leq \|m_t\|$. By triangle inequality,

$$\begin{aligned}
\|m_t\| - \|m_{t+1}\| \leq \|m_t - m_{t+1}\| &= (1-\beta)\|m_t - g(y_{t+1})\| \\
&\leq (1-\beta)(\|m_t\| + \|g(y_{t+1})\|).
\end{aligned}$$

Therefore, substitue the above inequality into lefthand side of (5.13) and regroup the fractions,

$$\begin{aligned}\frac{p\|m_{t+1}\|^2(\|m_t\|-\|m_{t+1}\|)}{(p\|m_t\|+q)(p\|m_{t+1}\|+q)} &\leq \frac{p\|m_{t+1}\|^2(1-\beta)(\|m_t\|+\|g(y_{t+1})\|)}{(p\|m_t\|+q)(p\|m_{t+1}\|+q)} \\ &= (1-\beta)\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}\frac{p\|m_t\|}{p\|m_t\|+q} \\ &\quad + \frac{(1-\beta)p\|g(y_{t+1})\|}{p\|m_t\|+q}\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q} \\ &\leq (1-\beta)\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q} + \frac{(1-\beta)p\|g(y_{t+1})\|}{q}\frac{\|m_t\|^2}{p\|m_t\|+q},\end{aligned}$$

where the last step we use the fact that $\|m_{t+1}\| \leq \|m_t\|$ and the function $x \to x^2/(px+q)$ is increasing on $\mathbb{R}_+$. Now, taking expectation on both sides of (5.13) yields

$$\begin{aligned}\mathbb{E}\left[\frac{p\|m_{t+1}\|^2(\|m_t\|-\|m_{t+1}\|)}{(p\|m_t\|+q)(p\|m_{t+1}\|+q)}\right] &\leq (1-\beta)\mathbb{E}\left[\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}\right] + \frac{p(1-\beta)}{q}\mathbb{E}\left[\|g(y_{t+1})\|\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\ &= (1-\beta)\mathbb{E}\left[\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}\right] + \frac{p(1-\beta)}{q}\mathbb{E}\left[\mathbb{E}\left[\|g(y_{t+1})\|\middle|\mathcal{G}_t\right]\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\ &\leq (1-\beta)\mathbb{E}\left[\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}\right] + \frac{p(1-\beta)G}{q}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\ &\leq (1-\beta)\mathbb{E}\left[\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}\right] + \frac{\beta(1-\beta)}{2}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right],\end{aligned}$$

where the third inequality follows by the fact that $\mathbb{E}[\|g(y_{t+1}\||\mathcal{G}_t] \leq \sqrt{L^2+\sigma^2}$ and the last inequality follows from our choice of parameters ensuring $pG/q \leq \beta/2$. Now

we are ready to proceed the telescopic summing. Summing up over t and yields

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{E}\left[\eta_t(-\|m_{t+1}\|^2 + \beta^2\|m_t\|^2)\right] \\
&\leq (1-\beta)\sum_{t=1}^{T}\mathbb{E}\left[\frac{\|m_{t+1}\|^2}{p\|m_{t+1}\|+q}\right] + \frac{\beta-\beta^2}{2}\sum_{t=1}^{T}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\
&\quad + \beta^2\mathbb{E}\left[\frac{\|m_1\|^2}{p\|m_1\|+q}\right] + (\beta^2-1)\sum_{t=2}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\
&= \frac{\beta^2+\beta}{2}\mathbb{E}\left[\frac{\|m_1\|^2}{p\|m_1\|+q}\right] + \frac{\beta^2-\beta}{2}\sum_{t=2}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\
&= \beta^2\mathbb{E}\left[\frac{\|m_1\|^2}{p\|m_1\|+q}\right] + \frac{\beta^2-\beta}{2}\sum_{t=1}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \\
&\leq \frac{\beta^2 G^2}{q} + \frac{\beta^2-\beta}{2}\sum_{t=1}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right].
\end{aligned}$$

The first inequality uses (5.13). The third line and the foruth line regroup the terms. The last line follows by $p\|m_1\|+q \geq q$ and $\mathbb{E}[\|m_1\|^2] \leq G^2$.

Combine all term i, ii and iii in (5.10) yields

$$\frac{\beta-\beta^2}{2}\sum_{t=1}^{T+1}\mathbb{E}\left[\frac{\|m_t\|^2}{p\|m_t\|+q}\right] \leq 2\beta(1-\beta)\mathbb{E}[f(x_1)-f(x_{T+1})] + \frac{\beta^2 G^2}{q} + T(1-\beta)^2\frac{G^2}{q}.$$

Multiply both sides by $\frac{2q}{T(\beta-\beta^2)}$ to obtain

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{q\|m_t\|^2}{p\|m_t\|+q}\right] \leq \frac{4q\Delta}{T} + \frac{2\beta G^2}{T(1-\beta)} + \frac{2(1-\beta)G^2}{\beta}. \tag{5.14}$$

We may assume $\epsilon \leq G$, otherwise any x_t is a (δ,ϵ)-stationary point. Then by choosing $\beta = 1-\frac{\epsilon^2}{64G^2}$, $p = \frac{64G^2\ln(16G/\epsilon)}{\delta\epsilon^2}$, $q = \frac{256G^3\ln(16G/\epsilon)}{\delta\epsilon^2}$, $T = \frac{2^{16}G^3\Delta\ln(16G/\epsilon)}{\epsilon^4\delta}\max\{1, \frac{G\delta}{8\Delta}\}$, have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{4G\|m_t\|^2}{\|m_t\|+4G}\right] \leq \frac{\epsilon^2}{17}. \tag{5.15}$$

Note that the function $x \to x^2/(x+4G)$ is convex, thus by Jensen's inequality, for

any t, we have

$$\frac{4G\mathbb{E}\left[\|m_t\|\right]^2}{\mathbb{E}[\|m_t\|]+4G} \leq \mathbb{E}\left[\frac{4G\|m_t\|^2}{\|m_t\|+4G}\right]. \tag{5.16}$$

Let us denote

$$m_{avg} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|m_t\|\right],$$

then again by Jensen's inequality,

$$\frac{4Gm_{avg}^2}{m_{avg}+4G} \leq \frac{1}{T}\sum_{t=1}^{T}\frac{4G\mathbb{E}\left[\|m_t\|\right]^2}{\mathbb{E}[\|m_t\|]+4G} \leq \frac{\epsilon^2}{17}$$

Solving the quadratic inequality with respect to m_{avg} and using $\epsilon \leq G$, we have

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

In contrast to the smooth case, we cannot directly conclude from this inequality since m_t is not the gradient at x_t. Indeed, it is the convex combination of all previous stochastic gradients. Therefore, we still need to find a reference point such that m_t is approximately in the δ-subdifferential of the reference point. Note that

$$m_t = \sum_{i=t-K+1}^{t}\alpha_i g(y_i) + \beta^K m_{t-K}$$

Intuitively, when K is sufficiently large, the contribution of the last term in m_t is negligible. In which case, we could deduce m_t is approximately in $\partial f(x_{t-K}+\delta B)$. More precisely, with $\beta = 1-\frac{\epsilon^2}{64G^2}$, as long as $K \geq \frac{64G^2}{\epsilon^2}\ln(\frac{16G}{\epsilon})$, we have

$$\beta^K \leq \frac{\epsilon}{16G}.$$

This is a simple analysis result using the fact that $\ln(1-x) \leq -x$. Then by Assumption on the oracle, we know that $\mathbb{E}[g(y_i)|\mathbf{Y}_i] \in \partial f(y_i)$ and $\|y_i - x_{t-K}\| \leq \frac{K}{p} \leq \delta$ for

any $i \in [t-K+1, t]$. Thus,

$$\mathbb{E}[g(y_i)|x_{t-K}] \in \partial f(x_{t-K} + \delta B).$$

Consequently, the convex combination

$$\frac{1}{\sum \alpha_i} \sum_{i=t-K+1}^{t} \alpha_i \mathbb{E}[g(y_i)|x_{t-K}] \in \partial f(x_{t-K} + \delta B).$$

Note that $\sum \alpha_i = 1 - \beta^K$, the above inclusion could be rewritten as

$$\frac{1}{1-\beta^K}(\mathbb{E}[m_t|x_{t-K}] - \beta^K m_{t-K}) \in \partial f(x_{t-K} + \delta B).$$

This implies that conditioned on x_{t-K}

$$\begin{aligned} d(0, \partial f(x_{t-K} + \delta B)) &\leq \frac{1}{1-\beta^K} \left(\|\mathbb{E}[m_t \mid x_{t-K}]\| + \beta^K \|m_{t-K}\| \right) \\ &\leq \frac{1}{1-\beta^K} \left(\mathbb{E}[\|m_t\| | x_{t-K}] + \beta^K \|m_{t-K}\| \right). \end{aligned}$$

Therefore, by taking the expectation,

$$\mathbb{E}[d(0, \partial f(x_{t-K} + \delta B))] \leq \frac{1}{1-\beta^K} \left(\mathbb{E}[\|m_t\|] + \beta^K G \right)$$

$$\leq \frac{1}{1-\frac{1}{16}}(\mathbb{E}[\|m_t\|] + \frac{\epsilon}{16}) = \frac{16}{15}\mathbb{E}[\|m_t\|] + \frac{\epsilon}{15}.$$

Finally, averaging over $t = 1$ to T yields,

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[d(0, \partial f(x_{t-K} + \delta B))] \leq \frac{16}{15T}\sum_{t=1}^{T} \mathbb{E}[\|m_t\|] + \frac{\epsilon}{15} \leq \frac{\epsilon}{3}.$$

When $t < K$, $\partial f(x_{t-K} + \delta B)$ simply means $\partial f(x_1 + \delta B)$. As a result, if we randomly out put $x_{\max\{1,t-K\}}$ among $t \in [1, T]$, then with at least probability 2/3, the δ-subdifferential set contains an element with norm smaller than ϵ. To achieve $1-\gamma$ probability result for arbitrary γ, it suffices to repeat the algorithm $\log(1/\gamma)$ times.

□

5.7.6 Proof of Theorem 5.5.2

Proof. The proof idea is similar to Proof of Theorem 5.3.1. Since the algorithm does not have access to function value, our resisting strategy now always returns

$$\nabla f(x) = 1.$$

If we can prove that for any set of points $x_k, k \in [1, K], K \leq \frac{\Delta}{8\delta}$, there exists two one dimensional functions such that they satisfy the resisting strategy $\nabla f(x_k) = 1, k \in [1, K]$, and that the two functions do not have two stationary points that are δ close to each other, then we know no randomized/deterministic can return an $(\delta, \epsilon)-$stationary points with probability more than $1/2$ for both functions simultaneously. In other words, no algorithm that query K points can distinguish these two functions. Hence we proved the theorem following the definition of complexity in (5.5).

From now on, let $x_k, k \in [1, K]$ be the sequence of points queried after sorting in ascending order. Below, we construct two functions such that $\nabla f(x_k) = 1, k \in [1, K]$, and that the two functions do not have two stationary points that are δ close to each other. Assume WLOG that x_k are ascending. First, we define $f : \mathbb{R} \to \mathbb{R}$ as follows:

$$\begin{aligned}
&f(x_0) = 0, \\
&f'(x) = -1 \quad \text{if} \quad x \leq x_1 - 2\delta, \\
&f'(x) = 1 \quad \text{if exists } i \in [K] \text{ such that} \quad |x - x_i| \leq 2\delta, \\
&f'(x) = -1 \quad \text{if exists } i \in [K] \text{ such that} \quad x \in [x_i + 2\delta, \frac{x_i + x_{i+1}}{2}], \\
&f'(x) = 1 \quad \text{if exists } i \in [K] \text{ such that} \quad x \in [\frac{x_i + x_{i+1}}{2}, x_{i+1} - 2\delta], \\
&f'(x) = 1 \quad \text{if} \quad x \geq x_K + 2\delta
\end{aligned}$$

It is clear that this function satisfies the resisting strategy. It also has stationary points that are at least 4δ apart. Therefore, simply by shifting the function by 1.5δ,

we get the second function.

The only thing left to check is that $\sup_k f(x_k) - \inf_x f(x) \leq \Delta$. By construction, we note that the value from x_i to x_{i+1} is non decreasing and increase by at most 4δ

$$\sup_k f(x_k) - f(x_0) \leq 4\delta K \leq \Delta/2. \tag{5.17}$$

We further notice that the global minimum of the function is achieved at $x_0 - 2\delta$, and $f(x_0 - 2\delta) = -2\delta \leq 4\delta K \leq \Delta/2$. Combined with (5.17), we get,

$$\sup_k f(x_k) - \inf_x f(x) \leq \Delta. \tag{5.18}$$

□

Chapter 6

Gradient Descent May Not Converge in Large Scale Applications

It is a well-known fact that nonconvex optimization is computationally intractable in the worst case. As a result, many theoretical analyses of optimization algorithms such as gradient descent often focus on local convergence to *stationary points* where the gradient norm is zero or negligible, and hope that the result can generalize to neural network experiments.

In this chapter, we examine the whether the convergence to stationary points is related to the convergence of neural network training. Specifically, we provide numerical evidence that in large-scale neural network training, such as in ImageNet + ResNet, and WT103 + TransformerXL models, the Neural Network weight variables *do not* converge to stationary points where the gradient of the loss function vanishes. Remarkably, however, we observe that while weights do not converge to stationary points, the value of the loss function converges. Inspired by this observation, we propose a new perspective based on ergodic theory of dynamical systems. We prove convergence of the distribution of weight values to an approximate invariant measure (without smoothness and assumptions) that explains how the training loss can stabilize without weights necessarily converging to stationary points. We further discuss how this perspective can better align the theory with empirical observations.

6.1 Introduction

It would not be controversial to claim that there currently exists a wide gulf between theoretical investigations of convergence to stationary points for non-convex optimization problems of the form

$$\min_{\theta} f(\theta), \tag{6.1}$$

and the empirical performance of popular algorithms used in deep learning practice. Due to the intrinsic intractability of general nonconvex problems, theoretical analysis of nonconvex optimization problems is often focused on the rates of convergence of gradient norm $\|\nabla f(\theta)\|$ instead of the suboptimality $f(\theta) - \min_\theta f(\theta)$. The vast theoretical literature on optimization for machine learning has documented the recent progress in this area. In particular, optimal gradient-based algorithms and rates have been identified in various nonconvex settings, including deterministic, stochastic and finite-sum problems [Carmon et al., 2017, Arjevani et al., 2019, Fang et al., 2018b].

In addition to theoretical interests in such problems, a practical motivation for such convergence analyses is to improve the convergence rate of large-scale optimization methods as they are used in machine learning practice, especially in training deep neural networks. As neural network models allow for efficient gradient evaluations, gradient-based algorithms have been the dominant methods to tune network parameters. Naturally, great effort was dedicated to theoretical understandings of gradient-based optimizers.

However, despite the fast progress on the theory side for gradient-based algorithms, the convergence analysis has had a limited impact on real-world neural network training. Despite many theoretical and empirical advances, the gap between theory and practice is as wide as ever. As an example, consider the variance reduction technique. Even though variance reduction theoretically accelerates convergence, recent empirical evidence in [Defazio and Bottou, 2018] suggests that it may be ineffective in speeding up neural network training. On the other extreme, ADAM [Kingma and Ba, 2014] is among the most popular algorithms in neural network training, yet

its theoretical convergence was proven to be incorrect [Reddi et al., 2019]. Despite dubious theoretical properties, ADAM is as popular as ever.

Our goal in this chapter is to address a very specific part of this theory-practice divide by providing an explanation for the ineffectiveness of theoretical convergence rates to stationarity in neural network training. First, we provide evidence that in many real-world experiments (e.g. ImageNet, Wiki103) where the model does not overfit the data, gradient-based optimization methods do not converge to stationary points as theory predicts. This mismatch questions the applicability of one of the key pillars of optimization theory as applied to neural network training. The reason for such a surprising divide is that most optimization analyses for deep learning either assume smoothness directly which leads to convergence to stationary points using classical analysis, or prove smoothness and fast convergence by relying explicitly on overparametrization. However, our empirical investigations reveal that the key premise of the theory — pointwise convergence to a fixed point — may not happen at all in practice.

Motivated by this observation, we aim to answer the following question in the rest of this chapter: *how should one define and analyze convergence of gradient-based optimization, when the training loss seems to converge yet the gradient does not converge to* 0*?*

We propose a new lens through which one should view convergence: rather than convergence of weights, we postulate that the convergence should be viewed in terms of invariant measures as used in the ergodic theory of dynamical systems. Using classical results from this literature, we then show how this new perspective can be consistent with some recent curious findings in neural network training, such as relaxed smoothness [Zhang et al., 2019b] and edge of stability [Cohen et al., 2021, Wu et al., 2018] phenomena. More concretely, our contributions are summarized as follows,

- We empirical verify through ResNet training and transformer-XL training in a wide range of applications that the iterates do not converge to a stationary point as existing theory predicts.

- We propose the invariant measure perspective from dynamical systems theory to explain why the training loss can converge without the iterates converging to a stationary point.

- Most importantly, we show that our theorems on diminishing gain of the loss without vanishing of the gradient apply to neural network training even without standard global Lipschitzness or smoothness assumptions.

- We discuss how our observations relate to interesting phenomena such as decay of function values, edge of stability, and relative smoothness.

6.1.1 Related work

Many recent results inspired us to investigate the oscillatory behavior of neural network training. One line of works is on the empirical investigations of neural network reproducibility. In [Henderson et al., 2017], the authors analyze the stability of policy reward in reinforcement learning and found large variations. In [Madhyastha and Jain, 2019], the authors study the instability for interpretation mechanisms. In [Bhojanapalli et al., 2021], the authors found that though image classification has relatively stable classification accuracy, the actual prediction on individual images has large variations. We learned from recent studies [Cohen et al., 2021, Zhang et al., 2019b, 2020a] that previous analysis assumptions on noise and smoothness not only have large variations but also adapt to the step size choice. These observations motivate us to rethink the convergence proofs used in classical optimization analysis. In addition, a few very recent results reported similarly large oscillations in Cifar10 training [Li et al., 2020, Kunin et al., 2021, Lobacheva et al., 2021], though the authors focus on SDE approximation or batch normalization. Our work instead focuses on the connection to nonconvex optimization theorems. Specifically we show that at the end of training when training loss has converged, even the full batch gradient norm does not converge to zero.

On the theory side, two lines of work are closely related to this chapter. One line of work studies the non-convergence of dynamics of algorithms in games or multiob-

jective optimization [Hsieh et al., 2019, Cheung and Piliouras, 2019, Papadimitriou and Piliouras, 2019, Letcher, 2020, Flokas et al., 2020]. Another models the SGD dynamics via Langevin dynamics [Cheng et al., 2020, Li et al., 2020, Gurbuzbalaban et al., 2021]. Our work differs from these works in that we do not look for global mixing, and do not focus on settings where additional noise needs to be injected.

6.2 Motivating examples

In this section, we provide some initial experimental results and show that the traditional notion of convergence for nonconvex functions does not really occur in deep neural network training. Our experiments are based on one of the most popular training schemes, where we trained ResNet101 on ImageNet [1] More experiments can be found later in Section 6.6.

To explain the quantities of interest, we first define our notation. Let $S = \{(x^i, y^i)\}_{i=1}^N$ be the dataset. We use $f(x, \theta)$ to denote the neural network function with model parameter θ and data input x. We use ℓ to denote the loss function such as cross-entropy after softmax. We would like to investigate the evolution of the following quantities during training. At iteration k, we evaluate

$$\begin{aligned}
&\text{Loss: } L_S(\theta_k) := \tfrac{1}{N}\textstyle\sum_{i=1}^N \ell(f(x^i, \theta_k), y^i),\\
&\text{Grad norm: } \|\nabla L_S(\theta_k)\|_2 := \|\tfrac{1}{N}\textstyle\sum_{i=1}^N \tfrac{\partial}{\partial\theta}\ell(f(x^i, \theta_k), y^i)\|_2,\\
&\text{Noise: } \sigma(\theta_k) := \sqrt{\tfrac{1}{N}\textstyle\sum_{i=1}^N \|\nabla L_S(\theta_k) - \tfrac{\partial}{\partial\theta}\ell(f(x^i, \theta_k), y^i)\|_2^2},\\
&\text{Sharpness: } \|\nabla^2 L_S(\theta_k)\|_{\text{op}} := \|\tfrac{1}{N}\textstyle\sum_{i=1}^N \tfrac{\partial^2}{\partial\theta^2}\ell(f(x^i, \theta_k), y^i)\|_{\text{op}},
\end{aligned} \tag{6.2}$$

where $\|\cdot\|_2$ is the standard vector ℓ_2 norm and $\|\cdot\|_{\text{op}}$ is the induced matrix spectral norm.

We then show how the trajectory of weights evolves for the standard training schedule, i.e., the learning rate starts at 0.1 and is decayed by a factor of 10 every 30

[1] We used the procedure from GitHub repository `https://github.com/jiweibo/ImageNet`, and were able to reproduce the original paper's result [He et al., 2016] up to 1% validation accuracy.

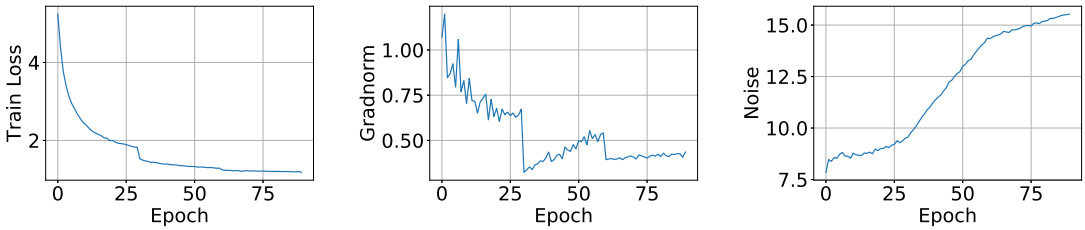


Figure 6-1: The quantities of interest (6.2) vs epoch for the default training schedule of ImageNet+ResNet101 experiment.

epochs. Evolution of the quantities is plotted in Figure 6-1. We make the following immediate observations:

- Within each period where the step size is held constant, the change in loss converges to 0.
- The gradient norm does not converge to 0 despite the fact that the loss function converges. In fact, the gradient norm is not significantly smaller than that at the start of the optimization.
- The noise level (in the stochastic gradient) increases during training.

The above observations suggest that there is *a major gap between theory and practice*. Much of the research on nonconvex optimization has focused on the convergence rate of gradient norms under a bounded-smoothness, bounded-noise setup. Faster algorithms are designed under this guidance. However, in practice, we find that the convergence of the training loss does not require the convergence of gradient norms. This may be the reason why techniques such as variance reduction or local regularization combined with Nesterov-momentum have had limited practical use, despite their massive theoretical popularity.

6.2.1 Different learning rates and training schedules

One immediate question following the observation in Figure 6-1 is whether the observed phenomenon holds solely for a particular stage-wise learning rate, which is not very common in theoretical analysis. In this subsection, we show that this cannot be

the reason and that the gradient norm does not converge to zero for *any* learning rate schedule. In particular, we run the same ResNet101 model on the ImageNet dataset just as before, except that we use a constant learning rate across all 90 epochs of training. The quantities are summarized in Figure 6-2. A quick glance at the plots verifies that gradient norm does not converge to 0 in any of the experiments. We further notice that a smaller learning rate leads to larger gradient norm, larger stochastic gradient noise intensity, and larger sharpness as observed in [Cohen et al., 2021]. We will further discuss the implications of these observations later in the chapter.

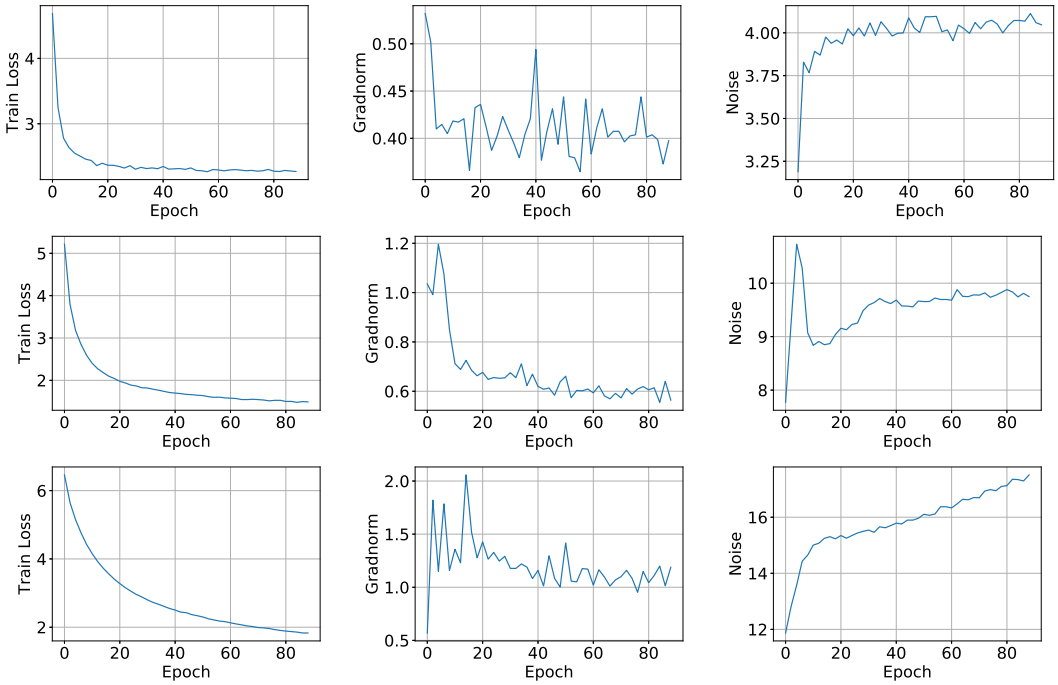


Figure 6-2: The quantities of interest (6.2) vs epoch for the constant learning rate training schedule in ImageNet experiments. The learning rate is set to be $0.1, 0.01, 0.001$ from the top row downwards. All models are trained for 90 epochs.

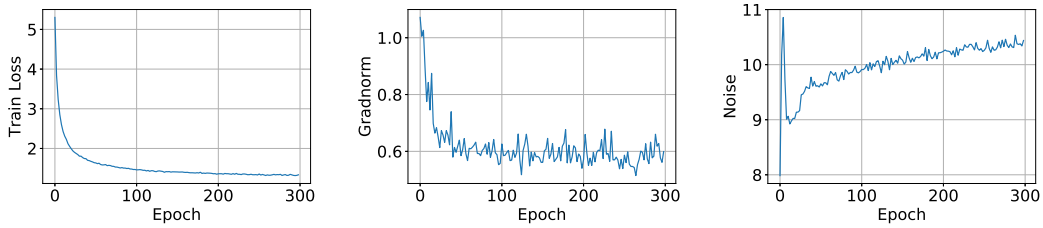


Figure 6-3: The estimated stats vs epoch for the constant learning rate $\eta = 0.01$ training schedule with total 300 epochs.

As the loss curves in the last two rows of Figure 6-2 are still decreasing, we continue the second row experiment (step size $\eta = 0.01$) for 300 epochs and present the result in Figure 6-3.

The above experiments show that in ImageNet+ResNet101 experiment, iterates of the weights do not converge to stationary points. In the next section, we test whether this phenomenon is an artifact of the data set and architecture and try to see if this also happens in other datasets and architectures.

6.2.2 Transformer XL experiments

We run Transformer-XL training on WT103 dataset for the language modeling task following the implementation of the original authors [Dai et al., 2019b]. Our training procedure is exactly the same as the official code, except that we reduce the number of attention layers for the baseline model from 6 to 4 so that the batch size fits in our GPU memory. Aside from training with a cosine learning rate schedule with initial learning rate $\eta = 0.00025$, we also experimented with different constant learning rates. The result is summarized in Figure 6-4.

We found that the observations made before also apply to transformer XL training. In particular, the gradient norm is not going to 0, and even not decreasing over the training. Furthermore, smaller step size also leads to larger gradient norm.

6.2.3 Refuted hypotheses and potential causes

With the above experiments, we can already **exclude** the following explanations on why gradient norm does not converge to zero.

1. The large gradient norm is due to the fact that the step size is not small enough or the model is not trained long enough.

2. This phenomenon is restricted to the particular ResNet + ImageNet combination.

3. The step size decreased too fast before the gradient norm could converge.

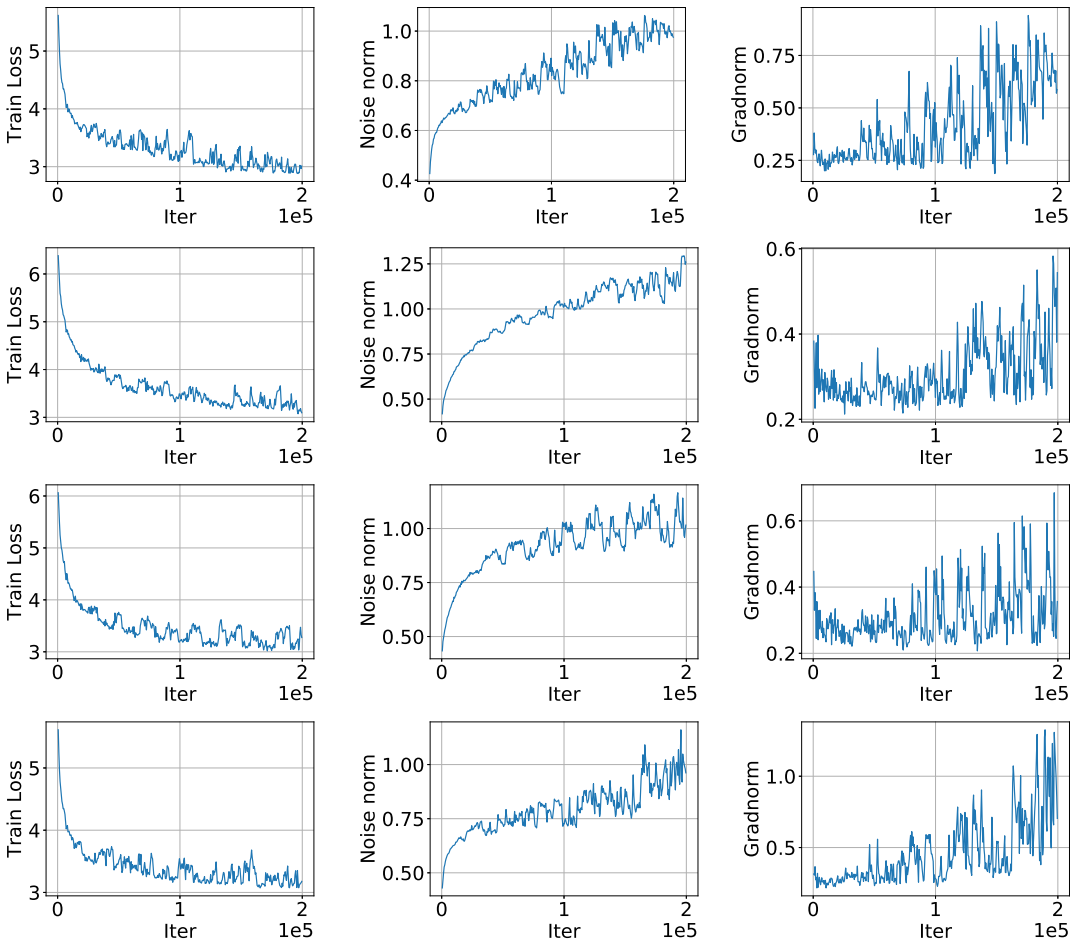


Figure 6-4: The estimated stats vs epoch for the transformer XL training. The learning rate is set to be cosine learning rate with $\eta = 0.00025$ in the first row. The learning rates are constant learning rates with $\eta = 0.00005, 0.0001, 0.00025$ from the second row downwards.

4. The large gradient norm is due to numerical or estimation error.

The first conjecture is excluded by comparing the experiment in Figure 6-3 against the experiments in the first two rows of Figure 6-2. We find that after running longer with a smaller step size, though the training loss dropped significantly, the gradient norm did not decrease at all. *This confirms that even the qualitative theoretical results (let alone the quantitative convergence rates) on when gradient norm gets smaller from canonical optimization analyses are not applicable to neural network training.*

The second conjecture is refuted by our TransformerXL [Dai et al., 2019a] experiment in Section 6.2.2. We see that this phenomenon in TransformerXL training is even more evident.

The third conjecture is refuted by comparing the experiment in Figure 6-3 against the experiments in the last two row of Figure 6-2. We see that longer training with larger constant step size also does not solve the problem.

The fourth conjecture is refuted due to our estimation precision discussed in later Section 6.6.2 with additional experimental details. We also observed that in our Cifar10 experiment in section 6.6.1, the gradient norm can indeed go to zero.

In the following sections, we attempt to provide a notion of convergence from the theory of dynamical systems. Our proposed explanation is that only the time average of the training loss converges, while the gradient norm is nonzero due to nonsmoothness, and that the actual weight iterates keep oscillating. Before diving into the theorem, we provide a conceptual explanation through a synthetic experiment in the next section.

6.3 An explanatory experiment

The curious phenomenon discussed above is not limited to neural network training. In what follows we present a simple synthetic example to illustrate the intuition behind the convergence behavior to unstable cycles rather than stationary points.

To this end, we simulate gradient descent on the objective function $f(\theta_1, \theta_2) = 100 \sin \theta_1 \sin \theta_2$ whose smoothness and Lipschitzness parameters are both $L_f = 100$.

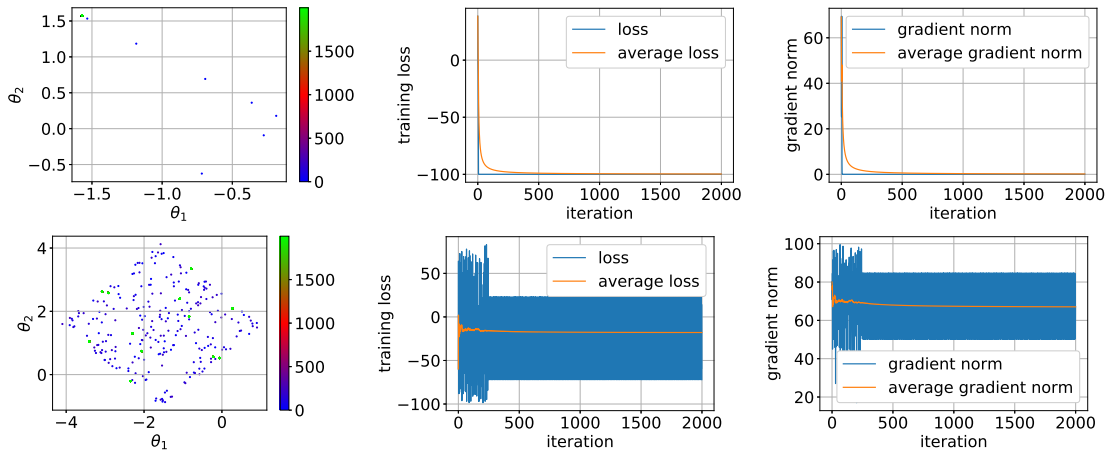


Figure 6-5: Synthetic experiment. The learning rate is set to be 0.01 and 0.04 for the first and second row respectively. Column I: the whole trajectory in 2000 iterations, where the scatter points correspond to iterates and the color of a point represents which iteration it is at; Column II: training loss and average training loss vs iteration, where the average is taken over iterations; Column III: gradient norm and average gradient norm vs iteration.

It is well known that gradient descent with a learning rate $\eta < 2/L_f = 0.02$ provably converges to stationary points for such a smooth function. As shown in the first row of Figure 6-5, the iterates converge to a fixed point very fast with $\eta = 0.01$, which also implies the convergence of both training loss and gradient norm. Moreover, the gradient norm converges to zero, which means a stationary point is reached at convergence.

However, when $\eta > 2/L_f$, which is often the case for neural network training, gradient descent no longer converges to stationary points as shown in the second row of Figure 6-5 with $\eta = 0.04$. During the last 500 iterations, the iterates only take values around a few points and keep oscillating among them. As a result, the training loss and gradient norm also oscillate and do not converge in the usual sense. However, the oscillation among these points follows some periodic pattern. If we collect all the iterates during a long enough training process, their empirical distribution will converge to a discrete distribution over those fixed points. As a result, the empirical distributions of the training losses and gradient norms at these iterates also converge. Then if we take an average of the training losses or gradient norms over time, it must converge to the expected value of the corresponding empirical distribution, as shown

in the last two images in Figure 6-5. However, although the average gradient norm converges, the convergence value can not be zero in presence of oscillation, as gradient descent makes no updates if the gradient is zero.

The above example shows that the key to function value convergence could be that a time average rather than a spatial average is taken in evaluating the function loss. In fact, we could verify this intuition through the example below. Recall that in ImageNet training, the plotted training loss is a moving average of previous iterations. In the left plot of Figure 6-6 we instead plot the training loss of the last 50 epochs in the experiment shown in Figure 6-3 and see that the variation across iterations is quite large considering the learning rate is 0.01 and the gradient norm is about 0.6. On the right plot, we show the last a few iterations of transformerXL training (see section 6.6 for more details) and observe even larger oscillations. Furthermore, even in Cifar10 experiments, both Li et al. [2020], Lobacheva et al. [2021] show very strong periodic divergence in training loss when the number of training epoch is huge (> 1000 epochs).

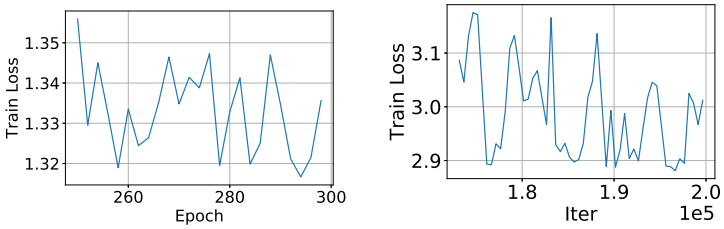


Figure 6-6: (Left) The training loss vs epochs with a constant learning rate $\eta = 0.01$ in the last 50 epochs of the 300-epoch ImageNet experiment. (Right) The training loss vs epochs with a constant learning rate $\eta = 0.01$ in the last iterations of the 2-million step transformerXL experiment.

Therefore, we see that the loss looks smooth because the oscillation is much smaller than the long-term loss decrease due to optimization. However, the variation is non-trivial considering the step size and gradient norm. Hence, the convergence only happens in the **time average** sense. With this intuition, we are now ready to present a more rigorous characterization of the phenomenon.

6.4 Convergence beyond stationary points

We have seen that even though the per-iteration loss does not converge, the time average with a long enough window size can converge. In this section, we provide a simple mathematical analysis to explain why that happens. In particular, we prove that the change in training loss evaluated as a time average converges to 0 for neural networks. Our analysis is motivated by, and closely follows the proof of the celebrated Krylov-Bogolyubov theorem on invariant measures. As a result, we refer to our interpretation as the *invariant measure perspective*.

Our key insight is that that the convergence of the training loss occurs in a time-average sense. To see this, consider the following dynamical system

$$\theta_{t+1} = F(\theta_t).$$

The time average naturally leads to the following notion of empirical measure:

$$\mu_k := \tfrac{1}{k} \textstyle\sum_{t=1}^{k} \delta_{\theta_t}, \tag{6.3}$$

where δ_θ denotes the Dirac measure supported on the value θ, i.e., $\delta_\theta(A) = 1$ if and only if $\theta \in A$, and $\{\theta_1, \theta_2, \cdots\}$ are the sequence of iterates generated by the dynamical system.

With this notation, we can conveniently write the time average of a scalar function $\phi : \mathcal{X} \to \mathbb{R}$ as

$$\mu_k(\phi) = \mathbb{E}_{\theta \sim \mu_k}[\phi(\theta)]. \tag{6.4}$$

In this chapter, we focus on the case when the dynamic system $F(\theta_t)$ denotes the SGD update, i.e.,

$$F(\theta_t) = \theta_t - \eta g(\theta_t),$$

where $g(\theta_t)$ denotes the stochastic gradient and η denotes the step size.

6.4.1 Invariant Measures

We say a measure μ is an **invariant measure** for the map $F : \mathcal{X} \to \mathcal{X}$ if for any measurable set A

$$\mu(A) = \mu(F^{-1}(A)) = \int_{\theta} \mathbb{1}_{\{F(\theta) \in A\}} d\mu(\theta),$$

where $F^{-1}(A) = \{\theta | F(\theta) \in A\}$. Notice that if F is a stochastic update, then this should be read as

$$\mu(A) = \mu(F^{-1}(A)) = \int_{\theta} \mathbb{P}\{F(\theta) \in A\} d\mu(\theta). \tag{6.5}$$

Invariance of measure is closely related to convergence of function values when $\theta \sim \mu$ is sampled from an invariant probability measure. In such a scenario, for any continuous function ϕ, the function value does not change after one update,

$$\mathbb{E}_{\theta \sim \mu}[\phi(\theta)] = \mathbb{E}_{\theta \sim \mu}[\phi(F(\theta))].$$

In other words, the function value does not change in expectation. Motivated by this observation, we state the following theorem:

Theorem 6.4.1 (convergence of function values)**.** *Consider a continuous scalar function $\phi : \mathcal{X} \to \mathbb{R}$. Assume that the update map F has the property that $\phi \circ F : \mathcal{X} \to [-M, M]$ has a bounded value for any $\theta \in \mathcal{X}$, then with probability $1 - \delta$ over the randomness of F,*

$$\left| \mathbb{E}_{\theta \sim \mu_n} \left[\phi(\theta) - \phi(F(\theta)) \right] \right| = \mathcal{O}\left(\frac{\log(1/\delta)}{\sqrt{n}} \right). \tag{6.6}$$

Proof. The proof can be found in Appendix 6.7.1. □

The above theorem shows that for a stochastic dynamical system, any bounded scalar function when evaluated on the empirical measure has vanishing updates as the number of iterations goes to infinity. We will later see how this relates to the

vanishing gain of training loss in neural network training.

The tightness of the $1/\sqrt{n}$ dependence in the above theorem easily follows by considering a stochastic map F such that $F(\theta)$ distributes uniformly over $\mathcal{X}$ for any θ. However, the tightness of the dependence on n is less clear when F is deterministic. We show below that even for maps on a compact set, the window size n in the time-average has to tend to infinity for the sequence to converge.

Lemma 6.4.2. *For any positive integer n, there exists a $1-$Lipschitz objective function f and a positive value $\mathcal{O}(1/n)$ such that the update $F(\theta) = \theta - \nabla f(\theta)$ has a compact invariant set, and the sequence of time average with a window size n starting at iteration t, $M_t = \frac{1}{n}\sum_{i=t}^{t+n-1} f(\theta_i)$ does not converge and*

$$\limsup_{t\to\infty} |M_t - M_{t-1}| \geq \mathcal{O}(1/n) > 0.$$

Proof. The proof can be found in Appendix 6.7.2. □

The above result suggests that even for gradient descent update, the per-iteration loss function value $\mathbb{E}_{\theta_t}[\phi(\theta_t)]$ may not converge unless the function value is evaluated as time average $\mu_t(\phi)$. The step size can be chosen arbitrarily but the exact dependence between step size η and window size n and the sequence difference $\limsup_{t\to\infty} |M_t - M_{t-1}|$ requires further investigation.

We also note the fact that the update vanishes to zero does not imply that the limit $\lim_{t\to\infty} \mu_t(\phi)$ exists. In fact, an explicit counterexample as stated below.

Theorem 6.4.3 (Yoccoz)**.** *There exists a dynamic system with deterministic continuous map $F : \mathcal{X} \to \mathcal{X}$ on a compact set and a scalar function $\phi \in C^\infty$, such that sequence $\frac{1}{n}\sum_{k\leq n} \phi(\theta_k)$ has no limit, where $\theta_{k+1} = F(\theta_k)$.*

Proof. See the original note in [Yoccoz]. □

Given the above negative result, we provide the following theorem to conclude this section.

Theorem 6.4.4 (convergence of distribution)**.** *Assume that F maps a compact set $\mathcal{X}$ to itself. Then the empirical distribution has a subsequence converging weakly to an ergodic distribution. In other words, there exists an invariant distribution μ, and a* ***subsequence*** *of positive integers $\{n_k\}_{k\in\mathbb{Z}}$ such that*

$$\mu_{n_k} \to_w \mu. \tag{6.7}$$

The proof of the above theorem is similar to the proof for the Krylov-Bogolyubov theorem. We include the proof in Appendix 6.7.3 for completeness. Note that since the iterates may not have a limit, only subsequence convergence is possible. We note that the above two theorems do not make use of the gradient descent structure. Whether the dynamic system resulting from gradient descent has exactly the same property is left as a challenging future problem.

With the above results, we are now ready to prove the vanishing property of the update of training loss in neural network training.

6.4.2 Vanishing change in training loss of neural networks

We are now ready to provide a theoretical analysis to prove the vanishing gain of training losses in neural network training, and thus explain how the training loss can stabilize even when the norm of the loss function gradient is non-zero . Our analysis is distinct from previous ones in the literature in that it does not assume global Lipschitzness or smoothness, does not rely on bounded noise assumptions, and it does not require perfectly fitting the data as in Neural tangent kernel (NTK) models or mean-field style arguments. The downside of this generality is that we only prove convergence of function values and do not make any comments on local or global optimality or generalization. We believe that remains to be done here and we have just scratched the surface.

In order to start the discussion, we define the following L-layer deep neural network

$f(x, \theta)$, where x is the input and $\theta = (W_0, \ldots, W_{L-1})$ is the network weights:

$$\begin{aligned} &x_0 = x, \\ &x_{l+1} = \sigma_{l+1}(W_l x_l), \quad l = 0, \ldots L-1 \\ &f(x, \theta) = x_L, \end{aligned} \tag{6.8}$$

where σ_l is a coordinate-wise activation function (e.g., ReLU or sigmoid). In practice, the last layer usually does not use any activation function so σ_L is the identity mapping. We do not consider pooling layers, convolutional layers, or skip connections for now and it should be easy to extend our analysis to these settings. Iteration (6.8) does not include batch normalization layers which we will analyze later in this section. Given a training dataset $S = \{(x^i, y^i)\}_{i=1}^N$, the empirical training loss is defined as

$$L_S(\theta) := \tfrac{1}{N} \textstyle\sum_{i=1}^N \ell(f(x^i, \theta), y^i),$$

where $\ell : \mathbb{R}^d \times [d] \to \mathbb{R}$ is a loss function and we assume $\|x^i\|_2 \leq 1$. The network is trained by SGD with weight decay, which is equivalent to running SGD on the following regularized loss

$$L_S^\gamma(\theta) := L_S(\theta) + \frac{\gamma}{2}\|\theta\|_2^2,$$

where $\|\theta\|_2$ denotes the ℓ_2 norm of vectorized θ. We will focus on the most widely used loss function for classification tasks, the cross-entropy after softmax, defined as follows.

$$\ell(x, y) = x_y - \log\left(\textstyle\sum_{j=1}^d e^{x_j}\right), \tag{6.9}$$

which has the following properties that we will use later.

Lemma 6.4.5. *The cross-entropy after softmax loss $\ell : \mathbb{R}^d \times [d] \to \mathbb{R}$ defined in* (6.9) *satisfies*

1. *If $\max_i x_i - \min_i x_i \leq c$, we have $\ell(x, y) \leq c + \log d$ for any $y \in [d]$.*

2. $\ell(x, y)$ is c_ℓ Lipschitz w.r.t. x for some numerical constant c_ℓ.

Next, we make the following assumption for the activation function. It holds for most activation functions including ReLU and tanh.

Assumption 6.4.1. Each activation function σ_l is (sub)-differentiable and c_σ coordinate-wise Lipschitz for some numerical constant $c_\sigma > 0$. Also assume $\sigma_l(0) = 0$.

Now we can prove the vanishing gain of the function values.

Theorem 6.4.6. *Suppose θ is initialized within the compact set $C_w := \{(W_0, \ldots, W_{L-1}) : \|W_l\|_{op} \leq w\}$ for some $w \leq (\gamma/c_\ell c_\sigma^L)^{1/(L-2)}$. Then the iterate θ_k for every k lies in C_w and the empirical measure generated by SGD with a stepsize $\eta \leq 1/\gamma$ satisfies*

$$\mathbb{E}_{\theta\sim\mu_n}[L_S(\theta) - L_S(F(\theta))] = \mathcal{O}(\tfrac{1}{\sqrt{n}}).$$

We notice that the initialization choice may not always hold in practice, especially when there is batch normalization design. We further note that similar to the above theorem, all (piece-wise) continuous scalar functions including the noise norm are bounded by compactness, and hence should stabilize after long enough training. However, in the third column of Figure 6-2, the noise norm does not really converge. To explain this observation, we propose the following theorem that studies neural networks with batch normalization.

For simplicity of analysis, we assume the last layer is one of the layers with batch normalization. For a vector x, we use x^2, $|x|$ and $\sqrt{x}$ to denote its coordinate-wise square, absolute value, and square root respectively. In the l-th layer, if it uses batch normalization, given a batch $\mathcal{B} = \{(x^i, y^i)\}_{i=1}^m$ sampled from some distribution $\mathcal{P}_\mathcal{B}$, batch normalization makes the following transformation from $\{x_{l-1}^i\}_{i\in\mathcal{B}}$ to $\{x_l^i\}_{i\in\mathcal{B}}$:

$$\begin{aligned}
\mu_{\mathcal{B},l-1} =& \tfrac{1}{m}\textstyle\sum_{i\in\mathcal{B}} x_{l-1}^i, \quad \sigma_{\mathcal{B},l-1}^2 = \tfrac{1}{m}\textstyle\sum_{i\in\mathcal{B}} \left(x_{l-1}^i - \mu_{\mathcal{B},l-1}\right)^2, \\
\hat{x}_l^i =& \tfrac{x_{l-1}^i - \mu_{\mathcal{B},l-1}}{\sqrt{\sigma_{\mathcal{B},l-1}^2 + \epsilon}}, \qquad x_l^i = a_l \cdot \hat{x}_l^i + b_l,
\end{aligned}$$

where a_l and b_l are the scale and shift parameters to be trained. We also use SGD with weight decay to train the network.

Theorem 6.4.7 (With batch normalization)**.** *Suppose the parameter of batch normalization layer a_L is initialized within the compact set $|a_L| \leq 2\sqrt{m}/\gamma$. Then the empirical measure generated by SGD with $\eta \leq 1/\gamma$ satisfies*

$$\mathbb{E}_{\theta\sim\mu_n,\mathcal{B}\sim\mathcal{P}_\mathcal{B}}[L_\mathcal{B}(\theta) - L_\mathcal{B}(F(\theta))] = \mathcal{O}(\tfrac{1}{\sqrt{n}}).$$

We have shown in this section how the expected change of the training loss in per iterate update converges to zero for neural network training without any smoothness or Lipschitzness assumptions. One weakness of our analysis is that the limit of the training loss may not exist. However, we are not sure whether or when in real-world experiments this happens, due to some recent research on how the loss could further improve or even diverge in extra long training epochs (> 500 epochs) [Liu et al., 2019, Li et al., 2020, Wightman et al., 2021].

Another caveat is that our gain is measured in terms of empirical measure instead of the last iterate distribution. On the theory side, it is very easy to construct nonconverging last iterate distribution by inducing some periodic loops. On the practice side, we believe that further efforts in understanding last iteration stability are required.

In the next section, by utilizing the structures of stochastic gradient descent, we see some interesting implications if we assume that the iterates are from a distribution when the change in training loss is zero.

6.5 Implications of the invariant measure theorem

While we have taken the first step, simply showing that the change in training loss converges to zero may not be interesting enough. In particular, up to now, the analysis in the previous section has not explained when and why the function value

could decrease. In this section, we look at implications of the invariant measure perspective from a different angle. We assume that the iterates already come from a distribution where the per step change is zero

$$\mathbb{E}_{\theta\sim\mu_k}[L_S(\theta) - L_S(F(\theta))] = 0,$$

and see what it implies in real world experiments.

6.5.1 Decreasing stepsize leads to smaller objective values

One well-known observation in neural network training is that when the training loss plateaus, reducing the learning rate can further reduce the objective. This phenomenon can be proved in theory if the function has globally bounded noise and smoothness constant. However, as we showed that the smoothness and noise level changes adversarially to the step size. In this section, we provide a partial explanation on when a smaller step size can decrease the function value. In particular, we consider the neural network setup introduced in Section 6.4.2. We make the following assumption:

Assumption 6.5.1. The neural network is second order differentiable though may not necessarily have bounded smoothness.

Then we could prove that reducing the step size by enough ratio would result in a decrease in function value.

Theorem 6.5.1. *Consider the stochastic gradient update $F : \mathcal{X} \to \mathcal{X}$ on a compact set defined as $F(\theta) = \theta - \eta g(\theta)$ for a fixed step size $\eta > 0$. Let μ be the invariant distribution such that $\mathbb{E}_{F,\theta\sim\mu}[L_S(\theta)] = \mathbb{E}_{F,\theta\sim\mu}[L_S(F(\theta))]$. If μ is not supported on stationary points (i.e. $\mathbb{E}_{\theta\sim\mu}[\|\nabla f(\theta)\|_2^2] > 0$), then there exists a small enough $c \in (0, 1)$ such that for any positive step size $\eta' < c\eta$, the update $F'(\theta) = \theta - \eta' g(\theta)$ will lead to a smaller function value, i.e.*

$$\mathbb{E}_{F',\theta\sim\mu}[L_S(F'(\theta))] < \mathbb{E}_{\theta\sim\mu}[L_S(\theta)].$$

Proof. See Appendix 6.7.7. □

The above theorem states that once the change in loss vanishes, by selecting a smaller step size, one could further reduce the loss. This reflects the observation in Figure 6-1. The challenge in the proof is that reducing the step size might lead to worse smoothness that is too large for the step size, and hence may increase the training objective. To prove the theorem, we draw intuition from the proofs of [Zhang et al., 2020b] and consider the line integral as an expectation. We then apply Markov inequality to control the level of variation caused by nonsmoothness.

6.5.2 Connections to edge-of-stability and relaxed smoothness

We now provide an informal discussion on how the invariant measure prospective can give insight to the edge-of-stability observation and relaxed smoothness phenomenon. Our argument will be informal and heuristic, and somewhat speculative. We believe a rigorous analysis is both interesting and challenging and leave them as future directions.

We start from the equation in the proof of the previous theorem in Appendix 6.7.7:

$$\mathbb{E}_{\theta,g}[\|\nabla L_S(\theta)\|_2^2] = \mathbb{E}_{\theta,g}\left[\eta \int_0^1 \int_0^1 \langle \nabla L_S(\theta) - g(\theta), \nabla^2 L_S(\gamma_{\theta,g}(t\tau\eta)) g(\theta)\rangle dt d\tau\right],$$

where $g(\theta)$ is the stochastic gradient and $\gamma_{\theta,g(\theta)}(r) = \theta - rg(\theta)$ denotes the line segment. Clearly, both sides are positive. We boldly extract an equation

$$\text{Grad}^2 = \eta \Sigma \mathcal{L} G, \tag{6.10}$$

where Grad denotes the gradient norm, Σ denotes the noise norm, $\mathcal{L}$ denotes the sharpness and G denotes the square root of the second moment of stochastic gradients. This is of course very rough, and is only true when we replace the inner product on the right hand side with its Cauchy-Schwartz upper bound, yet it has some interesting connection to the following two observations.

First, we recall the edge-of-stability framework [Cohen et al., 2021], which observes

that the actual smoothness constant during training neural network has an inverse relation to step size. This is true from the above equation if we hold Grad, η, Σ, G constant.

Second, in another work [Zhang et al., 2019b], the authors identified a positive correlation between the gradient norm and the smoothness constant. This relation can also be extracted from the equation if all other quantities are held constant.

In fact, as we observe that in practice, the relation between the sharpness and step size is not a direct inverse but indeed has some negative correlation. Therefore, we believe by studying the property of the equilibrium, one could understand why many counter-intuitive behaviors could happen.

6.6 Additional experiments details

In this section, we add some additional experiments and experimental details that supplement the results in Section 6.2. We showed that the observed phenomenon happens in large scale tasks. To supplement the result, we briefly comment on how smaller dataset presents different behavior by taking Cifar experiment as an example. In the end, we will discuss some experimental details on how the quantities in (6.2) are estimated.

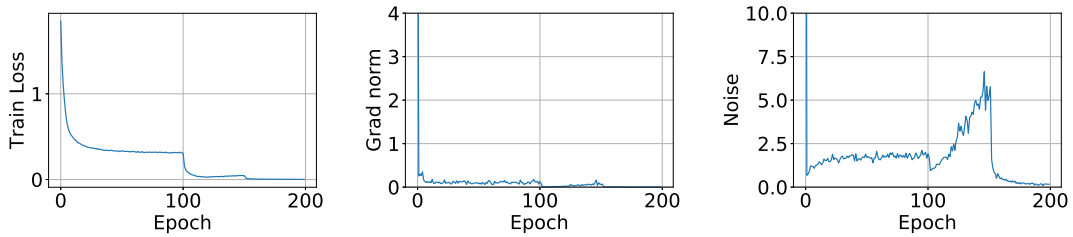


Figure 6-7: The estimated stats vs epoch for Cifar10 training. The learning rate starts at 0.1 and decay by a factor of 10 at epoch 100 and epoch 150.

6.6.1 Cifar10 Experiment

In this section, we show how noise, gradient norm and training loss evolve in Cifar10 with ResNet training. Our training procedure is based on the implementation[2]. The key result is demonstrated in Figure 6-7. We observe that in this case, the gradient norm indeed converges to 0. In fact, this is expected, as for cross entropy loss, the train loss could bound the gradient norm when weights are bounded.

The implications of the above observations are many. First, this separation behavior between small overfitting model on Cifar10 and larger model on ImageNet shows that the study of overparametrization and convergenece to stationary point may still be true in many cases. However, we should be careful that these analysis does not apply to larger models that do not overfit the data. Second, this shows that the SDE modeling in [Li et al., 2020, Lobacheva et al., 2021] can also be valid. It also shows that our work studies a problem of a different nature (non-zero grad norm).

6.6.2 Estimating the statistics

Here we provide additional details on how the values in (6.2) are estimated. Notice that these quantities are defined using all N data points in the entire dataset, which is too large in practice. Therefore, we use a random batch $m < N$ to estimate the quantities. For training loss, gradient norm, and noise norm, the estimation is straight-forward.

By Jensen's inequality, the estimated norms would be larger than the true value. However, the value should converge as the sampled batch size m converges to the total data number N. We show in Figure 6-8 and Figure 6-9 how these estimator values converge in practice. Based on these plots, we select the batch size to be 1.6×10^5 for ImageNet training and the token size to be 9×10^5 for the WT103 training. These sample sizes give a high enough precision level for making the observations in previous sections. Note that the estimated smoothness for the ImageNet experiment has very large variations, and hence we didn't make many comments on that plot throughout

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

this chapter.

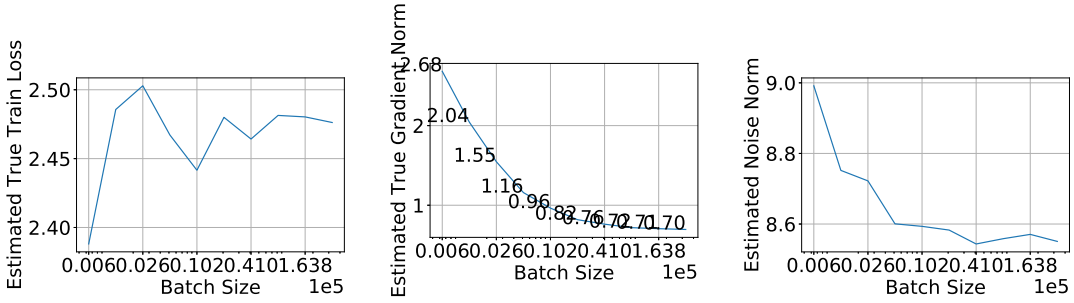


Figure 6-8: The estimated stats vs batch size for ImageNet training.

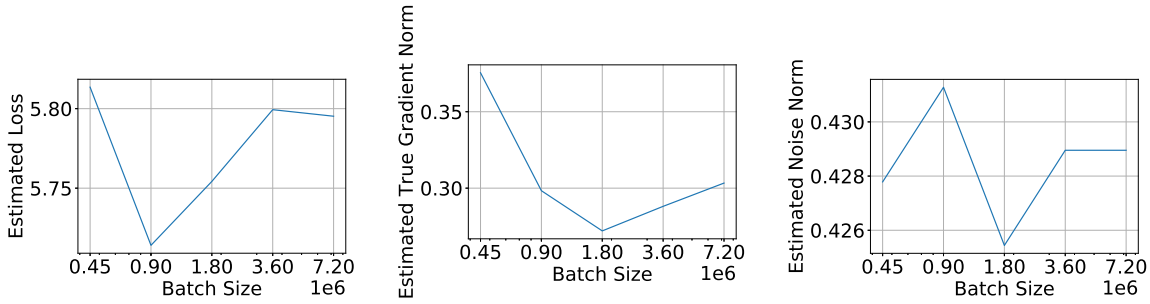


Figure 6-9: The estimated stats vs batch size for WT103 training.

6.7 Proofs

6.7.1 Proof of Theorem 6.4.1

Proof. By the fact that $\phi \circ F : \mathcal{X} \to \mathbb{R}$ has bounded value $[-M, M]$, we can denote the subgaussian norm at θ as

$$\sigma(\theta) = \inf\{\sigma > 0 | \mathbb{P}(\|\phi(F(\theta)) - \mathbb{E}[\phi(F(\theta))]\| \geq t) \leq 2e^{-t^2/2\sigma^2}\}.$$

In fact, $\forall \theta, \sigma(\theta) \leq M < \infty$. Hence, we can further denote the upperbound on the sub-Gaussian norm as

$$\sigma = \sup_{\theta} \sigma(\theta).$$

Then we consider two distributions. One is the empirical distribution of a sampled trajectory,

$$\mu_n = \frac{1}{n}\sum_{t=1}^{n} \delta_{\theta_t}.$$

The other one is the pushforward distribution $\mu_k(F^{-1})$ as defined in (6.5). Then,

$$\begin{aligned}
\mathbb{E}_{\theta\sim\mu_n}\left[\phi(\theta) - \phi(F(\theta))\right] &= \frac{1}{n}\sum_{t=1}^{n}\phi(\theta_t) - \frac{1}{n}\sum_{t=1}^{n}\phi(F(\theta_t)) \\
&= \frac{1}{n}(\phi(\theta_0) - \phi(F(\theta^n))) + \frac{1}{n}\sum_{t=1}^{n}\phi(\theta_t) - \phi(F(\theta_{-1})) \\
&= \mathcal{O}\left(\frac{1}{n}\right) + \frac{1}{n}\sum_{t=1}^{n}\phi(\theta_t) - \phi(F(\theta_{t-1})).
\end{aligned}$$

Then the claim follows by applying Hoeffding's inequality on the second term. □

6.7.2 Proof of Lemma 6.4.2

Proof. First, we construct a function

$$f(\theta) = \begin{cases} -\frac{1}{n+1}\theta & \theta \le 0, \\ \frac{n}{n+1}\theta & \theta > 0. \end{cases} \tag{6.11}$$

We notice that if $\theta_0 = \frac{0.2}{n+1}$, then $\theta_k = -1 + \frac{k-0.8}{n+1}, k \le n$ and $\theta_0, \theta_1, \theta_2 ... \theta_n$ is periodic with period $n+1$. We further notice that within a period $\theta_1, ..., \theta_{n+1}$, the function values are distinct, and $|f(\theta_0) - f(\theta_1)| \ge 0.5$. Therefore, for any $n > 0$, there exists a $t > n$ such that

$$|M_t - M_{t-1}| = \frac{1}{n}|f(\theta_{t+n-1}) - f(\theta_{t-1})| \ge \frac{1}{2n}.$$

□

6.7.3 Proof of Theorem 6.4.4

Proof. Since $\mathcal{X}$ is a compact metric space, we can find a dense countable set of the family of continuous functions $C(\mathcal{X})$, denoted as $\{\phi_1, \phi_2...\}$. Since $\mathcal{X}$ is compact, we have that $\mu_k(\phi_j)$ exists for any k, j. Therefore, by the diagonal argument, there exists a subsequence $\{n_k\}_k$ such that for all $j = 1, 2, ...,$

$$\lim_{k\to\infty} \frac{1}{n_k} \sum_{l\le n_k} \phi_j(\theta_l) = J(\phi_j).$$

Then by denseness of the set $\{\phi_1, \phi_2...\}$, we know that the above limit also exists for any $\phi \in C(\mathcal{X})$. Denote the functional as

$$J(\phi) = \lim_{k\to\infty} \frac{1}{n_k} \sum_{l\le n_k} \phi(\theta_l). \tag{6.12}$$

Since J is obviously linear and bounded, there exist a unique probability measure ϕ such that $J(\phi) = \mu(\phi)$.

The invariance of μ follows by the fact that for any continuous ϕ,

$$\begin{aligned}
\lim_{k\to\infty} |\mathbb{E}_{\theta\sim\mu_k} [\phi(\theta) - \phi(F(\theta))] | &= \lim_{k\to\infty} \frac{1}{n_k} \sum_{l\le n_k} \phi(\theta_l) - \frac{1}{n_k} \sum_{l\le n_k} \phi(F(\theta_l)) \\
&= \lim_{k\to\infty} \frac{1}{n_k} \sum_{l\le n_k} \phi(\theta_l) - \frac{1}{n_k} \sum_{l\le n_k} \phi(\theta_{l+1}) \\
&\quad + \lim_{k\to\infty} \frac{1}{n_k} \sum_{l\le n_k} \phi(\theta_{l+1}) - \frac{1}{n_k} \sum_{l\le n_k} \phi(F(\theta_l)) \\
&= \lim_{k\to\infty} \frac{1}{n_k} (\phi(\theta_1) - \phi(\theta_{n_k+1})) \\
&\quad + \lim_{k\to\infty} \frac{1}{n_k} \sum_{l\le n_k} (\phi(\theta_{l+1}) - \phi(F(\theta_l))) \to 0.
\end{aligned} \tag{6.13}$$

In the last line, the first term goes to zero by boundedness of function value on the compact set. The second term goes to zero by noticing that the sequence

$$M_n = \frac{1}{n} \sum_{l\le n} (\phi(\theta_{l+1}) - \phi(F(x_l))$$

is a martingale sequence. By the fact that each the induced martingale difference sequence has uniformly bounded sub-Gaussian norm, we can apply Hoeffding's inequality and know that M_n converge in probability to 0, which implies convergence in distribution.

□

6.7.4 Proof of Lemma 6.4.5

Proof.

1. Let $x_m = (\max_i x_i + \min_i x_i)/2$ and define $z_i = x_i - x_m$. We know $|z_i| \leq c/2$. Then we have

$$\begin{aligned}
|\ell(x,y)| =& \left| z_y + x_m - \log\left(\sum_{j=1}^d e^{z_j + x_m}\right)\right| \\
=& \left| z_y - \log\left(\sum_{j=1}^d e^{z_j}\right)\right| \\
\leq& c/2 + \log\left(de^{c/2}\right) \\
=& c + \log d.
\end{aligned}$$

2. As ℓ is differentiable, it suffices to bound its gradient norm. For any fixed $1 \leq k \leq d$, we have

$$\frac{\partial \ell(x,y)}{\partial x_k} = \delta_{y,k} - \frac{e^{x_k}}{\sum_{j=1}^d e^{x_j}}.$$

   Then we can bound

$$\begin{aligned}
\left\|\frac{\partial \ell(x,y)}{\partial x}\right\|_2 =& \sqrt{\sum_{k=1}^d \left(\frac{\partial \ell(x,y)}{\partial x_k}\right)^2} \\
\leq& \sqrt{1 + \frac{\sum_{k=1}^d e^{2x_k}}{\left(\sum_{j=1}^d e^{x_j}\right)^2}} \\
\leq& \sqrt{2}.
\end{aligned}$$

□

6.7.5 Proof of Theorem 6.4.6

Proof. Denote $\rho = wc_\sigma$. Then it is easy to show that within C_w, we have $\|x_l\|_2 \le \rho^l$ for every l. We define $z_{l+1} = W_l\theta_l$ and thus $x_l = \sigma_l(z_l)$. For any mini-batch $\mathcal{B} = \{(x^i, y^i)\}_{i=1}^m$, we can bound the gradient norm of the loss.

$$\left|\frac{\partial L_{\mathcal{B}}}{\partial W_l}\right| = \left|\frac{1}{m}\sum_{i\in\mathcal{B}} x_l^i(\nabla_x \ell(x_L^i, y^i))^\top D_L^{(i)}\left(\prod_{s=l+1}^{L-1} W_s D_s^{(i)}\right)\right| \le c_\ell c_\sigma \rho^{L-1} \le \gamma w,$$

where we define $D_l^{(i)} = \mathrm{Diag}(\sigma_l'(z_l^i))$. By the SGD rule, we have

$$W_l^{k+1} = (1-\eta\gamma)W_l^k - \eta\frac{\partial L_{\mathcal{B}}}{\partial W_l}.$$

Choosing $\eta \le 1/\gamma$, if $\|W_l^k\|_{\mathrm{op}} \le w$, we also have

$$\|W_l^{k+1}\|_{\mathrm{op}} \le (1-\eta\gamma)w + \eta\gamma w \le w.$$

By induction on k, the iterates of SGD optimizing the above objective always lie in C_w if the stepsize satisfies $\eta \le 1/\gamma$. Then we have $\|x_L^i\|_2 \le \rho^L$ and can bound that for any k

$$|L_S(\theta_k)| \le \log d + c_\ell\|x_L^i\|_2 \le \log d + \left(\frac{\gamma}{c_\ell c_\sigma^2}\right)^{L/(L-2)}.$$

The claim then follows by applying Theorem 6.4.1. □

6.7.6 Proof of Theorem 6.4.7

Proof. We first show that the coordinates of $\hat{x}_L^i$ are bounded.

$$\left|\hat{x}_L^i\right| = \frac{\left|x_{L-1}^i - \mu_{\mathcal{B},L-1}\right|}{\sqrt{\frac{1}{m}\sum_{i\in\mathcal{B}}\left(x_{L-1}^i - \mu_{\mathcal{B},L-1}\right)^2 + \epsilon}} \le \sqrt{m}.$$

As in Theorem 6.4.6, we show that during the training process, a_L always satisfies $|a_L| \le 2\sqrt{m}/\gamma$. Note that

$$\begin{aligned}\left|\frac{\partial L_{\mathcal{B}}}{\partial a_L}\right| &= \left|\frac{1}{m}\sum_{i\in\mathcal{B}} \hat{x}_L^i \cdot \nabla_x \ell(x_L^i, y^i)\right| \\ &= \left|\frac{1}{m}\sum_{i\in\mathcal{B}}\sum_{k=1}^{d} (\hat{x}_L^i)_k \left(\delta_{y,k} - \frac{e^{(x_L^i)_k}}{\sum_{j=1}^d e^{(x_L^i)_j}}\right)\right| \\ &\le \max_{i\in\mathcal{B}} \left|(\hat{x}_L^i)_y - \frac{\sum_{k=1}^d (\hat{x}_L^i)_k e^{(x_L^i)_k}}{\sum_{j=1}^d e^{(x_L^i)_j}}\right| \\ &\le 2\sqrt{m}.\end{aligned}$$

Therefore if $\left|a_L^k\right| \le 2\sqrt{m}/\gamma$, we have

$$\left|a_L^{k+1}\right| = \left|(1-\eta\gamma)a_L^k - \eta\frac{\partial L_{\mathcal{B}}}{\partial a_l}\right| \le (1-\eta\gamma)\cdot 2\sqrt{m}/\gamma + 2\eta\sqrt{m} \le 2\sqrt{m}/\gamma.$$

Then by induction, the above is true for every k. By Lemma 6.4.5, we have for every k

$$L_{\mathcal{B}}(\theta_k) \le 4m/\gamma + \log d.$$

Then the training loss $|L_{\mathcal{B}}(\theta_k)| \le 4m/\gamma + \log d$ is bounded during the training process if the stepsize satisfies $\eta \le 1/\gamma$. The theorem follows by applying Theorem 6.4.1. □

6.7.7 Proof of Theorem 6.5.1

Proof. For simplicity, we denote

$$\begin{aligned} f(\theta) &:= L_S(\theta), \\ \delta &:= \mathbb{E}_{\theta\sim\mu}[\|\nabla f(\theta)\|_2^2] > 0. \end{aligned}$$

By compactness of $\mathcal{X}$, we could denote the following quantities:

$$\begin{aligned}
G &= \sup_{\theta\in\mathcal{X}} \|g(\theta)\|_2 < \infty, \\
M^2 &= \sup_{\theta,\zeta\in\mathcal{X}} \mathbb{E}_{z\sim\text{unif}[\theta,\zeta]}[\|\nabla^2 f(z) - \mathbb{E}_{z'\sim\text{unif}[\theta,\zeta]}[\nabla^2 f(z')]\|_{\text{op}}^2] < \infty, \\
\Sigma &= \sup_{\theta\in\mathcal{X}} \|g(\theta) - \nabla f(\theta)\|_2 < \infty.
\end{aligned}$$

For clarity, note that for any function $f : \mathcal{X} \to \mathbb{R}^d$,

$$\mathbb{E}_{z\sim\text{unif}[\theta,\zeta]}[f(z)] = \int_0^1 f(t\theta + (1-t)\zeta)dt.$$

Therefore, we have that for any $c \in (0, 1)$

$$\begin{aligned}
&\int_0^1 \|\nabla^2 f(ct\theta + (1-ct)\zeta) - \mathbb{E}_{z\sim\text{unif}[\theta,\zeta]}[\nabla^2 f(z)]\|_{\text{op}}^2 dt \\
=&\frac{1}{c}\int_0^c \|\nabla^2 f(t\theta + (1-t)\zeta) - \mathbb{E}_{z\sim\text{unif}[\theta,\zeta]}[\nabla^2 f(z)]\|_{\text{op}}^2 dt \\
\leq&\frac{1}{c}\mathbb{E}_{z\sim\text{unif}[\theta,\zeta]}[\|\nabla^2 f(z) - \mathbb{E}_{z'}[\nabla^2 f(z')]\|_{\text{op}}^2].
\end{aligned}$$

Therefore, by Jensen's inequality, we have

$$\int_0^1 \|\nabla^2 f(ct\theta + (1-ct)\zeta) - \mathbb{E}_{z\sim\text{unif}[\theta,\zeta]}[\nabla^2 f(z)]\|_{\text{op}} dt \leq \sqrt{\frac{M^2}{c}}. \tag{6.14}$$

By applying Taylor expansion twice we get the following equations,

$$\begin{aligned}
\mathbb{E}_{\theta,F}[f(\theta) - f(F(\theta))] &= \mathbb{E}_{\theta,g}[f(\theta) - f(\theta - \eta g(\theta))] \\
&= \mathbb{E}_{\theta,g}[-\eta\int_0^1 \langle g(\theta), \nabla f(\gamma_{\theta,g(\theta)}(\eta t))\rangle dt] \\
&= \mathbb{E}_{\theta,g}[-\eta\|\nabla f(\theta)\|_2^2 + \eta\int_0^1 \langle g(\theta) - \nabla f(\theta), \nabla f(\gamma_{\theta,g(\theta)}(\eta t))\rangle dt] \\
&= \mathbb{E}_{\theta,g}[-\eta\|\nabla f(\theta)\|_2^2 + \eta\int_0^1 \langle g(\theta) - \nabla f(\theta), \nabla f(\gamma_{\theta,g(\theta)}(\eta t)) - \nabla f(\theta)\rangle dt] \\
&= \mathbb{E}_{\theta,g}[-\eta\|\nabla f(\theta)\|_2^2 - \eta^2\int_0^1\int_0^1 \langle g(\theta) - \nabla f(\theta), \nabla^2 f(\gamma_{\theta,g}(t\tau\eta))g(\theta)\rangle dtd\tau].
\end{aligned}$$

where $\gamma_{\theta,g(\theta)}(r) = \theta - rg(\theta)$ denotes the line segment.

By invariance of the function value, we get that

$$\begin{aligned}
&\mathbb{E}_{\theta,g}[-\eta\|\nabla f(\theta)\|_2^2 - \eta^2 \int_0^1 \int_0^1 \langle g(\theta) - \nabla f(\theta), \nabla^2 f(\gamma_{\theta,g}(t\tau\eta))g(\theta)\rangle dt d\tau] = 0 \\
\implies &\mathbb{E}_{\theta,g}[\|\nabla f(\theta)\|_2^2] = \mathbb{E}_{\theta,g}[\eta \int_0^1 \int_0^1 \langle \nabla f(\theta) - g(\theta), \nabla^2 f(\gamma_{\theta,g}(t\tau\eta))g(\theta)\rangle dt d\tau].
\end{aligned}$$

Therefore we have that

$$\begin{aligned}
&\mathbb{E}_{\theta,F'}[f(\theta) - f(F'(\theta))] \\
=&\mathbb{E}_{\theta,g}[-c\eta\|\nabla f(\theta)\|_2^2 - c^2\eta^2 \int_0^1 \int_0^1 \langle g(\theta) - \nabla f(\theta), \nabla^2 f(\gamma_{\theta,g}(t\tau\eta))g(\theta)\rangle dt d\tau] \\
\leq& c\eta \left(-\delta + c\Sigma G\sqrt{\tfrac{M}{c}}\right),
\end{aligned}$$

where in the last line we used (6.14). The claim follows by setting c small enough.

□

Chapter 7

Conclusions and Future Work

7.1 Summary

We will conclude this thesis by first reviewing what we have discussed. We start in Chapter 2 by introducing the oracle complexity framework of optimization analysis and by doing this, we can have an accurate interpretation of theoretical convergence rates. With the knowledge of theoretical results for optimization algorithms, we then move on to identify the mismatch between neural network experiments and optimization theory.

Once we described the different behaviors between theory predictions and neural network experiments, we aim to address why the gap exists and how we could improve the analysis in the next chapters. We do so by examining a few key assumptions in the theory analysis, including those on function class, oracle, and optimality measure.

The first approach we take is to study the smoothness assumption. We show in Chapter 7 that instead of bounding the smoothness with a global constant, the empirical observation suggests that maybe the smoothness can be better modeled by a gradient norm dependent function. Interestingly, we found that this model can justify the faster convergence of gradient clipping, which is a very popular empirical technique.

The second perspective we provide is to analyze the property of stochastic gradient oracles. In particular, we found that the better performance of Adam against SGD is

correlated with the existence of heavy-tailed stochastic noises. We show that in the more noisy settings, SGD suffers from the problem more than ADAM and may not converge. We further developed a new optimizer based on this intuition and improve over the ADAM on Bert related training tasks.

In the next project, we become more pedantic and try to address the question: "what theory can we develop if we do not ignore the fact neural networks can be non-differentiable?" We provide the first nonasymptotic analysis for the set of Lipschitz, potentially nondifferentiable, nonconvex functions. We achieve this by defining a new optimality measure and utilize ideas from adaptive step size and momentum.

In the last project, we identified a key gap between optimization theory and deep learning practice,namely, that during training with (stochastic) gradient descent, weights do not converge to stationary points of the loss function. Our numerical experiments reveal that this phenomenon cannot be eliminated by using a smaller learning rate or more training epochs. We further show that this is prevalent in applications from image classification to language modeling tasks. Using dynamical systems theory, we show that the convergence of training loss is only in the time average sense and that per iteration oscillation, though small compared to the long term trend, is significant compared to the step size, confirming the oscillation of iterates.

We hope that the perspectives and experiments in this thesis could provide researchers with additional insights and inspire efforts down the path of practice-driven theory research. However, we am well aware that this theis up till to is limited and far from the ultimate goal—to construct theorems that can guide practice. We will discuss some possible ideas and directions in the next chapter.

7.2 Discussions and future directions

The fact that theory and practice has a gap is well acknowledged in the machine learning community. However, compared to the vastly expanding conference publications(Neurips accepted list increased by 3 times over five years), there has been

limited research effort in trying to describe the mismatch and pinpoint the potential causes. I believe that I have just scratched the surface and much work remains to be done. We list some potential future directions below.

7.2.1 Adaptive algorithms and nonstationary system

The role of estimating moment with an moving average could need better understanding. In particular, one idea is to view RMSProp as a clipping algorithm and prove its convergence under shifting noise. The update for RMSProp can be written with effective step-sizes h_{rms} and h_{clip} respectively as below:

$$\begin{aligned} x_{k+1} &= x_k - \tfrac{\alpha}{\epsilon+\sqrt{\beta_2 v_k+(1-\beta_2)|g_k|^2}} g_k =: x_k - h_{\text{Adam}} g_k \,, \text{ and} \\ x_{k+1} &= \quad x_k - \eta_k \min\{\tfrac{\tau_k}{|g_k|}, 1\} g_k \quad =: x_k - h_{\text{clip}} g_k. \end{aligned}$$

Given any set of parameters for RMSProp, if we set the parameters for ACClip as

$$\eta_k = \tfrac{2\alpha}{\epsilon+\sqrt{\beta_2 v_k}} \quad \text{and} \quad \tau_k = \tfrac{\epsilon+\sqrt{\beta_2 v_k}}{\sqrt{1-\beta_2}},$$

then $\frac{1}{2} h_{\text{clip}} \leq h_{\text{Adam}} \leq 2 h_{\text{clip}}$. Thus, RMSProp can be seen as clipping where τ_k is set using $\sqrt{v_k}$, which estimates $\mathbb{E}[|g_k|^2]^{1/2}$, and a correspondingly decreasing step-size. An analysis of RMSprop (and Adam) by viewing them as adaptive clipping methods is a great direction for future work.

7.2.2 Different smoothness conditions

In chapter 3, we showed that different smoothness condition would motivate the design of different optimal algorithms. Though our proposed smoothness condition relaxes the usual Lipschitz assumption, it is unclear if there is an even better condition that also matches the experimental observations while also enabling a clean theoretical analysis. Second, we only study convergence of clipped gradient descent. Studying the convergence properties of other techniques such as momentum, coordinate-wise learning rates (more generally, preconditioning), and variance reduction is also inter-

esting. Finally, the most important question is: *"can we design fast algorithms based on relaxed conditions that achieve faster convergence in neural network training?"*

Our experiments regarding the smoothness esitmation also have noteworthy implications. First, though advocating clipped gradient descent in ResNet training is not a main point of this work, it is interesting to note that gradient descent and clipped gradient descent with large step sizes can achieve a similar *test performance* as momentum-SGD. Second, we learned that the performance of the baseline algorithm can actually beat some recently proposed algorithms. Therefore, when we design or learn about new algorithms, we need to pay extra attention to check whether the baseline algorithms are properly tuned.

7.2.3 Nonconvex nondifferentiable problems

Our results in Chapter 5 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.

7.2.4 Neural network training as a dynamic system

We took the first step in providing an invariant measure perspective according to which instead of analyzing a single trajectory of weights during training we study whether distribution of values that the weights take under the action of the gradient-based algorithms converge to an invariant measure. More specifically, we showed that the empirical measure has a subsequence that converges to an invariant measure and provide a negative example from [Yoccoz] that shows the sequence of distributions itself may not converge. An open question we left unanswered is whether the sequence itself can converge when the dynamical system follows a fixed step size update. A more interesting question for this research is to characterize the invariant measure and the conditions under which it may be unique, as well as to study convergence rate by exploiting more of the structure from neural network and gradient updates.

Our result only proves that that for multilayer perceptron, the weights under the action of the gradient update stay within a compact invariant set. Yet, we did not analyze the dependence of function properties (such as smoothness) on the initialization and optimizer hyperparameters. Understanding the dependence may naturally lead to better understanding and better algorithms. Another interesting question for future research would be to investigate the properties of the invariant measure and training loss as a function of the stepsize.

7.3 Concluding words

I feel fortunate to be able to run real-world experiments as a theory student, and to be able to read and examine theory as an engineer. I hope to close this thesis by citing the starting words from the average case analysis of simplex method from Spielman and Teng [2009], which the authors attribute to Professor Donald E. Knuth: "My experiences also strongly confirmed my previous opinion that the best theory is inspired by practice and the best practice is inspired by theory."

Bibliography

A. Agarwal and L. Bottou. A lower bound for the optimization of finite sums. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 78–86, 2015.

A. Agarwal, M. J. Wainwright, P. L. Bartlett, and P. K. Ravikumar. Information-theoretic lower bounds on the oracle complexity of convex optimization. 2009.

N. Agarwal, B. Bullins, X. Chen, E. Hazan, K. Singh, C. Zhang, and Y. Zhang. The case for full-matrix adaptive regularization. *arXiv preprint arXiv:1806.02958*, 2018.

Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

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

L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $o(1/n)$. In *Advances in Neural Information Processing Systems*, pages 773–781, 2013.

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

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

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

S. Bhojanapalli, K. Wilber, A. Veit, A. S. Rawat, S. Kim, A. Menon, and S. Kumar. On the reproducibility of neural network predictions. Feb. 2021.

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

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

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

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

Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points i. *arXiv preprint arXiv:1710.11606*, 2017.

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

X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

X. Cheng, D. Yin, P. Bartlett, and M. Jordan. Stochastic gradient and langevin processes. In *International Conference on Machine Learning*, pages 1810–1819. PMLR, 2020.

Y. K. Cheung and G. Piliouras. Vortices instead of equilibria in minmax optimization: Chaos and butterfly effects of online learning in zero-sum games. In *Conference on Learning Theory*, pages 807–834. PMLR, 2019.

D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl. On empirical comparisons of optimizers for deep learning. Oct. 2019.

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

J. M. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. Feb. 2021.

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

A. Cutkosky and H. Mehta. Momentum improves normalized sgd. *arXiv preprint arXiv:2002.03305*, 2020.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019a.

Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019b. URL `http://arxiv.org/abs/1901.02860`.

G. B. Dantzig. Origins of the simplex method. In *A history of scientific computing*, pages 141–151. 1990.

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

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

A. Defazio and L. Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv preprint arXiv:1812.04529*, 2018.

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, pages 1646–1654, 2014.

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

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

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

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

C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator. *arXiv preprint arXiv:1807.01695*, 2018a.

C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator, 2018b.

L. Flokas, E.-V. Vlatakis-Gkaragkounis, T. Lianeas, P. Mertikopoulos, and G. Piliouras. No-regret learning and mixed nash equilibria: They do not mix. *arXiv preprint arXiv:2010.09514*, 2020.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *ArXiv e-prints*, May 2017.

E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the heavy-ball method for convex optimization. Dec. 2014.

S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

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

P. Gong and J. Ye. Linear convergence of variance-reduced stochastic gradient without strong convexity. *arXiv preprint arXiv:1406.1102*, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

E. Gorbunov, M. Danilova, and A. Gasnikov. Stochastic optimization with heavy-tailed noise via accelerated gradient clipping. *arXiv preprint arXiv:2005.10785*, 2020.

M. Gurbuzbalaban, U. Simsekli, and L. Zhu. The heavy-tail phenomenon in sgd. In *International Conference on Machine Learning*, pages 3964–3975. PMLR, 2021.

E. Hazan, K. Levy, and S. Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2015.

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

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. Sept. 2017.

Y.-G. Hsieh, F. Iutzeler, J. Malick, and P. Mertikopoulos. On the convergence of single-call stochastic extra-gradient methods. *arXiv preprint arXiv:1908.08465*, 2019.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Feb. 2015.

C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

K. Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

V. Klee and G. J. Minty. How good is the simplex algorithm. *Inequalities*, 3(3): 159–175, 1972.

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

D. Kunin, J. Sagastuy-Brena, L. Gillespie, E. Margalit, H. Tanaka, S. Ganguli, and D. L. Yamins. Rethinking the limiting dynamics of sgd: modified loss, phase space oscillations, and anomalous diffusion. *arXiv preprint arXiv:2107.09133*, 2021.

L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

A. Letcher. On the impossibility of global convergence in multi-loss optimization. *arXiv preprint arXiv:2005.12649*, 2020.

K. Y. Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. *arXiv preprint arXiv:1805.08114*, 2018.

Z. Li, K. Lyu, and S. Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. *Advances in Neural Information Processing Systems*, 33, 2020.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

E. Lobacheva, M. Kodryan, N. Chirkova, A. Malinin, and D. Vetrov. On the periodic behavior of neural network training with batch normalization and weight decay. *arXiv preprint arXiv:2106.15739*, 2021.

P. Madhyastha and R. Jain. On model stability as a function of random seed. Sept. 2019.

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

S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SyyGPP0TZ`.

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

A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

A. Nemirovskii, D. B. Yudin, and E. R. Dawson. Problem complexity and method efficiency in optimization. 1983.

Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

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

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

T. H. Nguyen, U. Şimşekli, M. Gürbüzbalaban, and G. Richard. First exit time analysis of stochastic gradient descent under heavy-tailed gradient noise. *arXiv preprint arXiv:1906.09069*, 2019.

A. Panigrahi, R. Somani, N. Goyal, and P. Netrapalli. Non-gaussianity of stochastic gradient noise. *arXiv preprint arXiv:1910.09626*, 2019.

C. Papadimitriou and G. Piliouras. Game dynamics as the meaning of a game. *ACM SIGecom Exchanges*, 16(2):53–63, 2019.

R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2, 2012.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

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

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

B. T. Polyak. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1, 1987.

A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of ADAM and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

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

S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, pages 64–72, 2014.

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

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

U. Şimşekli, M. Gürbüzbalaban, T. H. Nguyen, G. Richard, and L. Sagun. On the heavy-tailed theory of stochastic gradient descent for deep neural networks. *arXiv preprint arXiv:1912.00018*, 2019a.

U. Şimşekli, L. Sagun, and M. Gürbüzbalaba. A tail-index analysis of stochastic gradient noise in deep neural networks. *arXiv preprint arXiv:1901.06053*, 2019b.

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

D. A. Spielman and S.-H. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, 2009.

M. Staib, S. J. Reddi, S. Kale, S. Kumar, and S. Sra. Escaping saddle points with adaptive gradient methods. *arXiv preprint arXiv:1901.09149*, 2019.

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

T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *arXiv preprint arXiv:1806.01811*, 2018.

R. Wightman, H. Touvron, and H. Jégou. Resnet strikes back: An improved training procedure in timm, 2021.

A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

B. Woodworth and N. Srebro. Tight complexity bounds for optimizing composite objectives. May 2016.

L. Wu, C. Ma, et al. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31:8279–8288, 2018.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

T. Yang, Q. Lin, and Z. Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. Apr. 2016.

J.-C. Yoccoz. An example of non convergence of birkhoff sums. `https://www.college-de-france.fr/media/jean-christophe-yoccoz/UPL54030_birkhoff.pdf`.

J. Zhang, A. Mokhtari, S. Sra, and A. Jadbabaie. Direct Runge-Kutta discretization achieves acceleration. *arXiv preprint arXiv:1805.00521*, 2018.

J. Zhang, T. He, S. Sra, and A. Jadbabaie. Analysis of gradient clipping and adaptive scaling with a relaxed smoothness condition. *arXiv preprint arXiv:1905.11881*, 2019a.

J. Zhang, T. He, S. Sra, and A. Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. May 2019b.

J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. J. Reddi, S. Kumar, and S. Sra. Why are adaptive methods good for attention models? Dec. 2019c.

J. Zhang, H. Lin, S. Das, S. Sra, and A. Jadbabaie. Stochastic optimization with non-stationary noise. *arXiv preprint arXiv:2006.04429*, 2020a.

J. Zhang, H. Lin, S. Jegelka, A. Jadbabaie, and S. Sra. Complexity of finding stationary points of nonsmooth nonconvex functions. Feb. 2020b.

Z. Zhou, Q. Zhang, G. Lu, H. Wang, W. Zhang, and Y. Yu. Adashift: Decorrelation and convergence of adaptive learning rate methods. *arXiv preprint arXiv:1810.00143*, 2018.

F. Zou and L. Shen. On the convergence of weighted AdaGrad with momentum for training deep neural networks. *arXiv preprint arXiv:1808.03408*, 2018.