DUKE UNIVERSITY

SVRG Escapes Saddle Points

by

Weiyao Wang

A thesis submitted to in partial fulfillment of the requirements for graduating with distinction in the Department of Computer Science degree of Duke University

Department of Computer Science, Duke University, Durham, North Carolina

2018

DUKE UNIVERSITY

Abstract

In this paper, we consider a fundamental problem of non-convex optimization that has wide applications and implications in machine learning. Previous works have shown that stochastic gradient descent with the variance reduction technique (`SVRG`) converges to a first-order stationary point in $O(\frac{n^{\frac{2}{3}}}{\epsilon^2})$ iterations in non-convex settings. However, many problems in non-convex optimization requires an efficient way to find second-order stationary points that are a subset of first-order stationary points, and therefore algorithms that converges to a first-order stationary point are not satisfying.

This paper shows how `SVRG` converges to a second-order stationary point in non-convex setting with a uniformly random perturbation. To find an ϵ-second-order stationary point, it takes $O(\frac{n^{\frac{3}{4}} polylog(\frac{d}{n\epsilon})}{\epsilon^2})$ iterations. In comparison with the convergence rate of `SVRG` to a first-order stationary point, the loss in convergence rate only depends poly-logarithmically on the dimension d and involves a small polynomial of n, $O(n^{\frac{1}{12}})$. This is the best known result in finding second-order stationary point.

We also give some intuitions and proof sketches to a new framework of analysis using the stable manifold theorem. The analysis from the new framework may help to eliminate the $n^{\frac{1}{12}}$ loss from our current analysis.

Our results can be directly applied to problems such as the training of neural networks and tensor decomposition. The intuition of stable manifold may provide independent interest to the non-convex optimization community.

Acknowledgements

This paper is based on my research independent studies mentored by Dr. Rong Ge with Ge's PhD students Xiang Wang from Duke University and Zhize Li from Tsinghua University. The ideas and proofs from this paper are results of the collective work of our group. We acknowledge that the work is still **in progress**, and there are still proofs that need to be completed and polished.

I thank my research advisor Dr. Ge for his genuine mentorship and continuous insights to the project. He has always been willing to spend lots of time with us discussing problems and offering advice. His passion about researches and teaching have inspired me during my time at Duke.

I also give special thanks Xiang and Zhize, with whom we tackled many difficulties throughout this project and without whom this paper would not be possible.

I thank Dr. Sayan Mukherjee, whose mentorship during the 2015 Data+ program at Duke motivated my interest to computer science and machine learning.

I thank Dr. Debmalya Panigrahi for organizing the Convex Optimization Seminar where I built intuitions for the field in general.

I thank Dr. Vincent Conitzer for being my computer science major advisor, who has always been willing to offer me help in my path to the degree.

Thanks to Dr. Ge, Dr. Muherjee and Dr. Panigrahi for being on my senior thesis committee!

Finally, I thank the Department of Computer Science at Duke for providing a supportive environment to gain knowledge and do research.

Contents

Abstract i

Acknowledgements ii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Our Contributions . . . 3
- 1.3 Related Works . . . 3

2 Preliminaries 6
- 2.1 Notations . . . 6
- 2.2 Definitions . . . 7
- 2.3 Algorithm: `SVRG` . . . 8

3 Main Result 11
- 3.1 Our Algorithm . . . 11
- 3.2 Convergence Rate of `PSVRG` . . . 13

4 Proof Sketch for Main Theorem (Theorem 3.2) 15
- 4.1 Proof Intuition . . . 15
  - 4.1.1 Perturbation and Intuitions for Two-Phases Analysis . . . 16
- 4.2 Some Key Lemmas . . . 17
  - 4.2.1 Upper Bounding Variance by Distance . . . 17
  - 4.2.2 Upper Bounding Distance by Function Value Decrease . . . 17
  - 4.2.3 Upper Bounding Gradient by Function Value Decrease . . . 18
- 4.3 Exploiting Large Gradient . . . 19
- 4.4 Exploiting Negative Curvature . . . 19
  - 4.4.1 Proof Sketch to Bound Width of Stuck Region . . . 20
- 4.5 Bounding Increase Induced by Perturbations . . . 21
  - 4.5.1 Proof Intuitions . . . 22
  - 4.5.2 Proof Sketch . . . 22
- 4.6 Proof of Main Theorem (Theorem 3.2) . . . 23

5 Preliminary Works Using the Stable Manifold 25
- 5.1 Stable Manifold of Gradient Descent Map . . . 26
  - 5.1.1 Definitions . . . 26
  - 5.1.2 The Stable Manifold Theorem . . . 27

5.2 Proof Sketch with $\frac{1}{\sqrt{\epsilon}}$-Smooth Assumption 27

Bibliography **31**

Chapter 1

Introduction

1.1 Background

In machine learning, many problems require optimization in non-convex settings (eg. [GHJY15], [GLM16] and [Kaw16]). In contrast to convex settings where a first-order stationary point is equivalent to an approximate global minimum, an ϵ-first-order stationary point (i.e. a point x such that $\| \nabla f(x) \| \leq \epsilon$, see Definition 2.3) can be a global minimum, a local minimum or a saddle point in a non-convex setting. This implies that optimization methods that find only first-order stationary points are not satisfactory for these problems in machine learning.

On one hand, although finding global minimum is NP-hard, many problems that require optimizing non-convex functions only require a local minimum instead of a global minimum. For example, for problems like tensor decomposition [GHJY15], matrix completion [GLM16] and some classes of deep neural networks [Kaw16], all local minima are global minimum. Such nice property relaxes the global minimum requirement to local minimum. On the other hand, saddle points fail to give the same optimality as global minimum in many problems (eg. matrix squareroot [JJKN15]). In addition, saddle points can be ubiquitous in high-dimensional and non-convex optimization problems, which has become a major problem in training neural networks (eg. [DPG+14]).

Therefore, we are driven to find algorithms that efficiently converge to a local minimum and escape saddle points. To formalize the problem setting, this paper focuses on finding algorithms that converge to an ϵ-second-order stationary point x (see Definition 2.7) of a ρ-Hessian Lipschitz function f (see Definition 2.6). In addition, given that all saddle points are strict (i.e. for any saddle point x_s, $\lambda_{min}(\nabla^2 f(x)) < 0$), all second-order stationary points are local minima ([JGN+17]).

In previous works, [JGN+17] has shown that with phasic perturbations, gradient descent (`GD`) will converge to an ϵ-second-order stationary point for l-smooth (see Definition 2.2) and ρ-Hessian Lipschitz functions. `GD` applies an update step as

$$x_{t+1} \leftarrow x_t - \frac{1}{l}\nabla f(x)$$

and it converges to an ϵ-second-order stationary point within $\tilde{O}(\frac{l(f(x_0)-f(x^*))}{\epsilon^2})$ iterations, where $\tilde{O}$ hides the polylog factors of dimension d. The algorithm is efficient in the sense that the convergence rate has only poly-logarithmic dimension loss compared to the well-known convergence rate of gradient descent to an ϵ-first-order stationary point in [Nes04].

However, many machine learning problems (eg. training neural networks) consider the unconstrained minimization problem in the following form:

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

where each sub-function f_i is l-smooth and ρ-Hessian Lipschitz.

In such category, the gradient descent method requires computing the full gradient and is n times more costly than computing the stochatic gradient $\nabla f_i(x)$. Therefore, it is natural to ask if we can generalize the analysis in [JGN+17] to some stochastic optimization method that converges to a first-order stationary point and does not require computing the full gradient.

A natural generalization would be stochastic gradient descent (`SGD`), which applies an update step as

$$x_{t+1} \leftarrow x_t - \eta\nabla f_i(x)$$

where f_i is uniformly at random sampled from $\{f_1, ..., f_n\}$. Analysis in [JG17] shows that `SGD` with phasic perturbation converges to an ϵ-second-order stationary point within $\tilde{O}(\frac{l(f(x_0)-f(x^*))\sigma^2}{\epsilon^4} + \frac{l^2(f(x_0)-f(x^*))}{\rho^{0.5}\epsilon^{2.5}})$ iterations, where σ is the sub-Gaussian tail of the variance of the stochastic gradient. `SGD` computes a single stochastic gradient ∇f_i in each update step and thus is n times faster than `GD` in each iteration. But the total number of iterations required for convergence is much more than `GD` ([GL16]).

Therefore, we would like to consider a generalization of the analysis in [JGN+17] to algorithms faster than both `GD` and `SGD` in non-convex settings. We consider the variance reduction techniques, based on the `SVRG` method proposed in [JZ13a]. [AZH16] has shown that `SVRG` converges an ϵ-first-order stationary point within $O(\frac{n^{\frac{2}{3}}l(f(x_0)-f(x^*))}{\epsilon^2})$ iterations in non-convex setting. Since each iteration of `SVRG` is n times faster than `GD`,

SVRG is $\Omega(n^{\frac{1}{3}})$ times faster than GD. Since each iteration of SVRG is as fast as SGD, it is faster than SGD as well. Thus, we ask if we are able to extend the analysis in [JGN+17] to SVRG by adding phasic perturbations to show that SVRG converges to an ϵ-second order stationary points efficiently.

1.2 Our Contributions

In this paper, we modify the algorithm presented in [AZH16] and present the perturbed version of SVRG (PSVRG, algorithm 3).

We show that the modified SVRG (algorithm 2) with phasic perturbations (PSVRG, algorithm 3) escapes saddle points and converges to an ϵ-second-order stationary point x^* in $\tilde{O}(\frac{(f(x_0)-f(x^*))}{\epsilon^2})$ iterations. The result is summarized in Theorem 1.1:

Theorem 1.1 (Simplified Version of Theorem 3.2)**.** *Assume f is l-smooth and ρ-Hessian Lipschitz. Algorithm 3 with initialization x_0 and a given set of parameters converges to an ϵ-second-order stationary point x^* in*

$$T = \tilde{O}(\frac{(f(x_0) - f(x^*))}{\epsilon^2})$$

iterations, where $\tilde{O}$ hides the polylog dependence on d, ϵ and n.

Since our algorithm computes $\Omega(n^{\frac{1}{4}})$ less stochastic gradients in each iteration than GD and has the same number of iterations as GD, our algorithm is at least $\Omega(n^{\frac{1}{4}})$ faster than GD. Compared to SVRG in [AZH16] that converges to first-order stationary point in non-convex settings, our algorithm has two losses in convergence rate. The first loss depends poly-logarithmically on dimension d and ϵ. As noted by [JGN+17], this loss may be tight. The second loss is a small polynomial of n, $O(n^{\frac{1}{12}})$.

To reduce the $O(n^{\frac{1}{12}})$ loss in our algorithm, we present a potential direction of analysis that works with a smaller mini-batch size b in algorithm 3, which gives the same convergence rate as [AZH16], with only polylog loss in dimensions d and ϵ: $\tilde{O}(\frac{n^{\frac{2}{3}}}{\epsilon^2})$. We consider a geometric approach using the stable manifold and gives some preliminary proofs assuming certain geometric properties of the stable manifold.

1.3 Related Works

There have been many works dedicated to finding second-order stationary points in non-convex settings. One line of work focused on problem-specific algorithms. For example,

for problems like phase retrieval, some works use a smart initialization algorithm to give an estimate of the second order stationary point inside a local neighborhood and apply local search algorithms with the nice initialization ([NJS15] and [SQW16]). But we are interested in finding algorithms that generalize well to different types of problems, and these algorithms and their analyses are problem-based.

As summarized in [JGN+17], traditional methods that converge to second-order stationary points usually use second-order information (i.e. information about Hessian). For example, [NP06] presents a cubic regularization method that extends the Newton method and finds a second-order stationary point in $O(\frac{1}{\epsilon^{1.5}})$ iterations. Unfortunately, each iteration of these methods require inverting the Hessian matrix $\nabla^2 f(x)$, which is very costly to compute.

To avoid such computation and still take advantage of the information given by the Hessian matrix, algorithms based on Hessian-vector-product are proposed. These algorithms compute $\nabla^2 f(x) \cdot u$ or $\nabla^2 f_i(x) \cdot u$, where u is a direction vector. [AAZB+17] and [CDHS16] implement this idea and show that it is possible to converge to an ϵ-second-order stationary point in $O(\frac{1}{\epsilon^{1.75}})$ iterations. In many cases, the oracle that computes the Hessian-vector-product can be implemented efficiently, roughly the same complexity as computing the first-order gradient. However, some have noted that such oracle may not generalize well to different types of problems and may be hard to implement in several contexts ([CHDS17]).

Therefore, researchers are driven to find algorithms that are based on first-order information (i.e. gradient) for each iteration. These methods usually bring new analyses to the existing gradient-based methods or modify the existing gradient-based methods such as `GD` or `SGD`. [GHJY15] shows that `SGD` converges to a second-order stationary point in $poly(\frac{d}{\epsilon})$ with the degree of the polynomial at least 4, which is later improved to $O(\frac{d^3}{poly(\epsilon)})$ by [Lev16].

Base Algorithm	**Perturbation Method**	**Iterations (stochastic gradient)**
`GD`	Random Perturbation	$\tilde{O}(\frac{n}{\epsilon^2})$
`GD`	`neon2`	$\tilde{O}(\frac{n}{\epsilon^2})$
`SGD`	Random Perturbation	$\tilde{O}(\frac{\sigma^2}{\epsilon^4} + \frac{1}{\rho^{0.5}\epsilon^{2.5}})$
`SGD`	`neon2`	$\tilde{O}(\frac{1}{\epsilon^4})$
(**Our Work**) `SVRG`	Random Perturbation	$\tilde{O}(\frac{n^{\frac{3}{4}}}{\epsilon^2})$
`SVRG`	`neon2`	$\tilde{O}(\frac{n^{\frac{2}{3}}}{\epsilon^2} + \frac{n}{\epsilon^{1.5}} + \frac{n^{\frac{3}{4}}}{\epsilon^{1.75}})$

TABLE 1.1: Summary of the Convergence Rate for Gradient-Based Methods with the Two Different Perturbation Approaches

To obtain a better convergence rate for these algorithms, a natural intuition is to perturb the point so that if a point is near to the saddle point, the perturbation will help the

point to escape the saddle point. One line of research is to compute a nice perturbation through the negative-curvature search ([AZL17]). For every interval in a gradient-based algorithm, [AZL17] checks if the sequence is near a saddle point and adds a perturbation v along the direction where the Hessian matrix has large negative eigenvalue. This process can be implemented efficiently by `neon2`, a modified negative-curvature search algorithm.

Another line of research is simpler in implementation: [JGN+17] adds a perturbation uniformly sampled from a ball to `GD`, which is later extended to `SGD` in [JG17], and shows that with the random perturbation the algorithms are able to converge to ϵ-second-order stationary points. This paper applies the uniformly random perturbation method to `SVRG` and shows that with this phasic perturbation, `SVRG` converges to an ϵ-second-order stationary point.

Chapter 2

Preliminaries

In this chapter, we formally introduce some definitions and notations that are mentioned in previous chapter or will be used in following chapters. Then we present our algorithm `PSVRG` in details and some existing results relevant to our analysis.

2.1 Notations

For function-related notation: let $f = \frac{1}{n}\sum_{i=1}^{n} f_i : \mathbb{R}^d \to \mathbb{R}$, the function we try to minimize. Let ∇f be the gradient of function f and $\nabla^2 f$ be the Hessian matrix of f. We use f^* to denote the global minimum of f and x^* as the minimizer such that $f(x^*) = f^*$.

For matrix-related notation: let $\|\cdot\|$ denotes the l_2 norm of a vector and $\|\cdot\|$ denotes the spectral norm for a matrix. We use $\lambda_{max}(\cdot)$, $\lambda_{min}(\cdot)$ and $\lambda_i(\cdot)$ to denote the maximum eigenvalue, minimum eigenvalue, and the i^{th} largest eigenvalue of a matrix.

For complexity-related notation: we use $O(\cdot)$ to hide absolute constants that are independent of any problem parameter and $\tilde{O}(\cdot)$ to hide both absolute constants and poly-logarithmic dependence on dimension d, ϵ and n.

Let $\mathbb{B}_x^d(r)$ be a d-dimensional ball centered at x with radius r; we hide d when it is clear from the context.

Let $\mathcal{P}_S(\cdot)$ be the projection operator to the set or subspace S.

2.2 Definitions

We follow the definitions from [JGN+17]. In convex settings, we usually consider functions that are both smooth and convex.

Definition 2.1. A twice-differentiable function f is α**-strongly** convex if

$$\forall x, \lambda_{min}(\nabla^2 f(x)) \geq \alpha$$

Definition 2.2. A differentiable function f is l**-smooth** or l**-gradient Lipschitz** if

$$\forall x_1, x_2, \parallel \nabla f(x_1) - \nabla f(x_2) \parallel \leq l \parallel x_1 - x_2 \parallel$$

The smoothness of a function implies that the gradient can not change too rapidly with respect to the distance covered by the algorithm ($\parallel x_t - x_0 \parallel$).

Definition 2.3. Given a differentiable function f, x is a **first-order stationary point** if $\parallel \nabla f(x) \parallel = 0$; x is an ϵ**-first-order stationary point** if $\parallel \nabla f(x) \parallel \leq \epsilon$.

A first-order stationary point can be a local minimum, a saddle point, or a local maximum. Since we aim at minimizing the objective function f, a saddle point or a local maximum is undesirable. We abuse the naming and call both "saddle points". The formal definition follows:

Definition 2.4. For a differentiable function f, x is a local minimum is x is a first-order stationary point and there exists a neighborhood of x, such that for any y in the neighborhood, $f(x) \leq f(y)$. x is a saddle point if it is a first-order stationary point but no a local minimum. If f is further twice-differentiable, a saddle point x of f is strict (or non-degenerate) if $\lambda_{min}(\nabla^2 f(x)) < 0$.

Remark 2.5. If f is twice-differentiable, a saddle point x is of f satisfies $\lambda_{min}(\nabla^2 f(x)) \leq 0$. The reason for a strictness definition on x is that if $\lambda_{min}(\nabla^2 f(x)) = 0$, information of Hessian $\nabla^2 f(x)$ alone is insufficient to determine whether x is a saddle point or a local minimum. Thus, with a strictness condition on saddle point, convergence to a second-order stationary point implies convergence to a local minimum.

We define a notion on Hessian similar to the Lipschitz notion on gradient in Definition 2.2:

Definition 2.6. A twice-differentiable function f is ρ**-Hessian Lipschitz** if

$$\forall x_1, x_2, \parallel \nabla^2 f(x_1) - \nabla^2 f(x_2) \parallel \leq \rho \parallel x_1 - x_2 \parallel$$

Similar to l-smoothness, Hessian Lipsthiz implies that the Hessian does not change rapidly with respect to the spectral norm.

We also generalize ϵ-first-order stationary point (definition 2.3) to second order:

Definition 2.7. For twice-differentiable function f, x is a **second-order stationary point** if

$$\| \nabla f(x) \| = 0, \text{ and } \lambda_{min}(\nabla^2 f(x)) \geq 0$$

If f is ρ-Hessian Lipschitz, x is an **ϵ-second-order stationary point** if

$$\| \nabla f(x) \| \leq \epsilon, \text{ and } \lambda_{min}(\nabla^2 f(x)) \geq -\sqrt{\rho\epsilon}$$

2.3 Algorithm: SVRG

We recall SVRG converges to a first-order stationary point in non-convex setting, and the algorithm is described as follows ([AZH16]):

Algorithm 1 SVRG

Require: initial point x_0, number of epoch $S = T/m$, number of iterations per epoch m, step size η,

$\widetilde{x}^0 = x_0$
for $s = 1, 2, \ldots, S$ **do**
  $x_0^s = \widetilde{x}^{s-1}$
  $g^s = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(x_0^{s-1}) = \nabla f(x_0^{s-1})$
  for $t = 1, 2, \ldots, m$ **do**
    $v_{t-1}^s = \left(\nabla f_i(x_{t-1}^s) - f_i(\widetilde{x}^{s-1})\right) + g^s$
    $x_t^s = x_{t-1}^s - \eta v_{t-1}^s$
  end for
  Select a random $m^s \in \{m, m-1, \ldots, m-m_0+1\}$ with probability proportional to

$$\{\beta_{m_0-1}, \frac{10}{9}\beta_{m_0-1}, \frac{10}{9}(\beta_{m_0-1} + \beta_{m_0-2}), \ldots, \frac{10}{9}(\beta_{m_0-1} + \ldots + \beta_1)\}$$

  where $\beta_t = (1 + \frac{1}{m_0})^{-t}$
  $\widetilde{x}^s = x_{m^s}$
end for

Similar to SGD, the update step for SVRG can also be seen as a stochastic update step of GD:

$$\mathbb{E}[v_{t-1}^s] = \nabla f(x_{t-1}^s) = \mathbb{E}[\nabla f_i(x_{t-1}^s)]$$

The advantage of SVRG is that the variance of the stochastic update v_{t-1}^s is smaller than the variance of $\nabla f_i(x_{t-1}^s)$ in SGD.

In convex settings, SVRG (algorithm 1) converges linearly to a global optimum ([JZ13b]):

Theorem 2.8. *Given a function f that is both l-smooth and α-strongly convex. Choose m to be sufficiently large so that*

$$\beta := \frac{1}{\alpha\eta(1-2l\eta)m} + \frac{2l\eta}{1-2l\eta} < 1$$

Then for any $\epsilon > 0$, if we run SVRG (algorithm 1) starting at x_0, iterate x_t will be ϵ-close to global optimum x^* in iterations:

$$T = \frac{1}{\beta} log(\frac{\| x_0 - x^* \|}{\epsilon})$$

In this paper, we introduce a mini-batch extension of SVRG introduced in [AZH16] and base our algorithm on the mini-batch version (2). The mini-batch version of SVRG does not require a random sampling of the points at the end of each inner loop (every m iterations) as required in algorithm 1.

Algorithm 2 mini-batch SVRG

Require: initial point x_0, number of epoch $S = T/m$, minibatch size b, step size η,

$\widetilde{x}^0 = x_0$
for $s = 1, 2, \ldots, S$ **do**
  $x_0^s = \widetilde{x}^{s-1}$
  $g^s = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(x_0^{s-1}) = \nabla f(x_0^{s-1})$
  for $t = 1, 2, \ldots, m$ **do**
    $v_{t-1}^s = \frac{1}{b}\sum_{i\in I_b} \left(\nabla f_i(x_{t-1}^s) - f_i(\widetilde{x}^{s-1})\right) + g^s$
    $x_t^s = x_{t-1}^s - \eta v_{t-1}^s$
  end for
  $\widetilde{x}^s = x_m^s$
end for

We note that we can re-write the update step as the gradient descent with some variance:

$$\begin{aligned}
x_t^s &= x_{t-1}^s - \eta v_{t-1}^s \\
&= x_{t-1}^s - \eta(\frac{1}{b}\sum_{i\in I_b}(\nabla f_i(x_{t-1}^s) - f_i(\widetilde{x}^{s-1})) + \nabla f(x_0^{s-1})) \\
&= x_{t-1}^s - \eta(\nabla f(x_{t-1}^s)) - \eta(\frac{1}{b}\sum_{i\in I_b}(\nabla f_i(x_{t-1}^s) - f_i(\widetilde{x}^{s-1})) + \nabla f(x_0^{s-1}) - \nabla f(x_{t-1}^s)) \\
&= x_{t-1}^s - \eta(\nabla f(x_{t-1}^s)) - \eta g_t^s
\end{aligned}$$

where $g_t^s = \frac{1}{b}\sum_{i\in I_b}(\nabla f_i(x_{t-1}^s) - f_i(\widetilde{x}^{s-1})) + \nabla f(x_0^{s-1}) - \nabla f(x_{t-1}^s)$. Therefore, we see how SVRG is related to GD: it gives an extra term g_t^s compared to GD in each iteration. Therefore, we call g_t^s **variance**.

An extension of the analysis in [AZH16] to mini-batch setting and a simplification of [LL18] gives the convergence rate of the mini-batch `SVRG` (algorithm 2) to ϵ-first-order stationary point in non-convex setting:

Theorem 2.9. *Assume f is l-smooth, algorithm 2 with initialization x_0 and parameters $m = n^{\frac{1}{3}}$, $b = n^{\frac{2}{3}}$ and $\eta = \frac{1}{l}$ converges to an ϵ-first-order stationary point x^* in*

$$T = O(\frac{(f(x_0) - f^*)}{\epsilon^2})$$

inner loop iterations in expectations (i.e. T updates for x). Since the mini-batch in each iteration computes $n^{\frac{2}{3}}$ stochastic gradients, the total number of iterations measured by the number of stochastic gradients is

$$O(\frac{(f(x_0) - f^*)n^{\frac{2}{3}}}{\epsilon^2})$$

Chapter 3

Main Result

In this chapter, we show that `PSVRG` (algorithm 3) converges to a second-order stationary point efficiently.

3.1 Our Algorithm

We modify the original version of `SVRG` (algorithm 1) presented in [AZH16] in the following ways:

1. We add a phasic perturbation after each super-epoch (i.e. combination of several epochs). The perturbation is sampled from a ball with given radius.
2. Similar to algorithm 2, we extend to the mini-batch setting: in each iteration, instead of sampling one f_i, we sample b f_i's uniformly at random and replaces the update step by

$$x_{t+1} \leftarrow x_t - \eta(\frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\tilde{x}) + \frac{1}{b}(\sum_{i\in I_b}\nabla f_i(x_t) - \nabla f_i(\tilde{x})))$$

   where I_b represents the b indexes sampled.
3. We run two sequences simultaneously, one from the perturbed point and one from the unperturbed point, and use the sequence with enough function value decrease.
4. We take the last point of the previous epoch as the snapshot point instead of sampling a point from the previous epoch as the snapshot point. This is one convenience in analysis given by the mini-batch extension.

Algorithm 3 PSVRG

Require: initial point x_0, number of epoch $S = T/m$, minibatch size b, step size η, time interval t_{thres}, decrease of function value $\mathscr{F}$, perturbation radius r

1: $x_{init} = \widetilde{x}^0 = x^0$
2: **for** $s = 1, 2, \ldots, S$ **do**
3: **if** $sm \ \% \ t_{thres} \neq 0$ **then**
4: $x_0^s = \widetilde{x}^{s-1}$
5: $g^s = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(x_0^{s-1}) = \nabla f(x_0^{s-1})$
6: **for** $t = 1, 2, \ldots, m$ **do**
7: $v_{t-1}^s = \frac{1}{b}\sum_{i \in I_b} \left(\nabla f_i(x_{t-1}^s) - f_i(\widetilde{x}^{s-1})\right) + g^s$
8: $x_t^s = x_{t-1}^s - \eta v_{t-1}^s$
9: **end for**
10: $\widetilde{x}^s = x_m^s$
11: Obtain $\{\widehat{x}_t^s\}_{t=1}^m$ by repeating Step 5–11 (replace $\widetilde{x}^{s-1}$ by $\widehat{x}^{s-1}$)
12: **else**
13: **if** $f(x_{init}) - f(\widetilde{x}^{s-1}) \geq \mathscr{F}$ **then**
14: $x^s = x_m^{s-1}$, $x_{init} = x_m^{s-1}$
15: **else**
16: $x^s = \widehat{x}_m^{s-1}$, $x_{init} = \widehat{x}_m^{s-1}$
17: **end if**
18: $\widetilde{x}^s = x^s$
19: $\widehat{x}^s = x^s + \zeta^s, \quad \zeta^s$ uniformly $\sim \mathbb{B}_0(r)$
20: **end if**
21: **end for**

There are two loops in algorithm 3: an outer loop (3-20) and an inner loop (7-8). For each inner loop, there are m iterations, and after each inner loop, we compute a new snapshot point $\tilde{x}_s$. We call each m steps, or the interval between two snapshot points, an **epoch**. In outer loop, each time the total iterations reach t_{thres}, we add a perturbation uniformly at random sampled from $\mathbb{B}_0(r)$. We call each t_{thres} iterations, or the interval between we add perturbations, a **super-epoch**.

In the algorithm, we obtain two sequences in each super-epoch: $\{\tilde{x}^s\}$ and $\{\hat{x}_t^s\}$, where $\{\tilde{x}^s\}$ is the sequence without perturbation at the initialization and $\{\hat{x}_t^s\}$ is the sequence with perturbation ζ^S at the initialization.

The number of stochastic gradients computed in each iteration is b, since we sampled b f_i's uniformly at random from $\{f_1, ..., f_n\}$ in each iteration.

The algorithm perturbs after a super-epoch, and we keep both a running sequence starting at the unperturbed point and a running sequence starting at the perturbed point. After each epoch, we choose the sequence that decreases enough in value of $\mathscr{F}$. If neither sequence has enough decrease in function value, our analysis shows that the sequences must be close to a second-order stationary point.

3.2 Convergence Rate of PSVRG

Before we present the Main Theorem that summarizes our result, we acknowledge that the proof is not yet finished and the project is still in progress. We give the following conjecture prior to the proof:

Conjecture 3.1. **We assume that the lemmas in section 4.2 not only hold in expectations as we proved, but also hold as high probability bounds: with high probability, the bounds in the lemmas hold up to polylog factors (of ϵ, d and n) without the expectation notation.**

Under Conjecture 3.1, we have the following result on the convergence rate of PSVRG:

Theorem 3.2 (Main Theorem)**.** *Assume f is l-smooth and ρ-Hessian Lipschitz. Assume Conjecture 3.1 is true. After T inner loop iterations in total, algorithm 3 with initialization x_0 and parameters $m = n^{\frac{1}{4}}$, $b = n^{\frac{3}{4}}$, $\eta = \frac{1}{l}$, $t_{thres} = O(\frac{log(\frac{d}{n\epsilon})}{\eta\epsilon^{0.5}})$, $\mathscr{F} = \epsilon^{1.5}$ and $r = \sqrt{b}\epsilon^{\frac{3}{4}}$ outputs x such that*

$$f(x_0) - \mathbb{E}[f(x)] \geq \Omega(\frac{\epsilon^2}{log(\frac{d}{n\epsilon})})T$$

if not returned an ϵ-second-order stationary point. In other words, the algorithm converges to an ϵ-second-order stationary point x^* in

$$O(\frac{(f(x_0) - f^*)log(\frac{d}{n\epsilon})}{\epsilon^2})$$

iterations in expectations. Since the mini-batch in each iteration computes $n^{\frac{3}{4}}$ stochastic gradients, the total number of iterations measured by the number of stochastic gradients is

$$\tilde{O}(\frac{(f(x_0) - f^*)n^{\frac{3}{4}}}{\epsilon^2})$$

Theorem 3.2 implies that the convergence rate of PSVRG is not very different from the convergence rate of SVRG from Theorem 2.9. There are two losses. First, there is a loss of $n^{\frac{3}{4}} - n^{\frac{2}{3}} = n^{\frac{1}{12}}$. This is induced by the change of the mini-batch size in algorithm 3 and is necessary to our analysis. Second, there is a loss poly-logarithmically depending on n, ϵ and dimension d. It is analogous to the loss of Perturbed Gradient Descent from Gradient Descent in [JGN+17]. And as argued by [JGN+17], such *polylog*(d) may be unavoidable.

An advantage of our algorithm is simplicity: we only do uniformly random perturbations instead of perturbation from negative curvature search. This makes the algorithm

simpler to implement. We also note that we are hoping to reduce the extra $n^{\frac{1}{12}}$ through new frameworks in Chapter 5.

The intuition of Theorem 3.2 is also similar to that from [JGN+17]. Consider a point x that is not yet an ϵ-second-order stationary point. x either has large gradient $\| \nabla f(x) \| > \epsilon$ or has a Hessian with large negative eigenvalue $\lambda_{min}(\nabla^2 f(x)) < -\sqrt{\rho\epsilon}$. For the first case, it is sufficient to apply the analysis similar to Theorem 2.9, which implies the algorithm converges to a point with small gradient (first-order stationary point). For the second case, we will follow the framework of [JGN+17] that after the perturbation a point with Hessian of large negative eigenvalue decreases in function value.

We also observe that the extensions of perturbed gradient descent to the context of strict-saddle in [JGN+17] works for `PSVRG`.

Chapter 4

Proof Sketch for Main Theorem (Theorem 3.2)

In this chapter, we give an outline of the proof of Theorem 3.2 that states PSVRG escapes saddle points and converges to an ϵ-second-order stationary point. We follow the framework developed by [JGN+17] that bounds the **stuck region** in which the algorithm initializes will not escape from saddle point efficiently. Then we support our proof sketch with some key lemmas we proved.

We note that the current proofs we developed are bounds in expectations, and to make the entire framework consistent, we still need to convert current proofs from bounds in expectations to bounds in high probability.

4.1 Proof Intuition

In general, the proof takes advantage of the properties of a point if it is not a second-order stationary point: either (1) it has a large gradient (section 4.3) or (2) its Hessian has a large negative eigenvalue (section 4.4). We divide the algorithm into super-epochs, each with length $\tilde{O}(\frac{1}{\eta\epsilon^{0.5}})$, and we prove that in either case the function value decrease within the super epoch will be $\Omega(\epsilon^{1.5})$ (Lemma 4.6 and Lemma 4.7). The length of super-epoch is carefully chosen, so that we can obtain the same decrease guarantee of function value for both case (1) and case (2).

4.1.1 Perturbation and Intuitions for Two-Phases Analysis

Between the super-epochs, we add a uniformly random perturbations around the snapshot point (line 19 in algorithm 3). Naively, radius r has an upper bound $r^2 \leq \mathscr{F}$, where r^2 is an approximate upper bound for function value increase induced by the perturbation, and the function value increase should be bounded by the function value decrease of a super-epoch, $\mathscr{F}$. This gives a naive upper bound for $r \leq \epsilon^{\frac{3}{4}}$. However, the perturbation should be large so that its exponential growth dominates the growth of variance when we analyze case (2) (see section 4.4). This gives a lower bound for $r \geq \sqrt{b}\epsilon^{\frac{3}{4}}$. This causes a problem of the function value increase induced by the perturbation to be as large as $b\epsilon^{1.5}$ and larger than the upper bound $\mathscr{F}$.

Specifically, we solve this problem by showing that after a small number of iterations, if the function value of the unperturbed sequence does not decrease by $\mathscr{F}$, the function value of the perturbed sequence will be at most $O(\epsilon^{1.5})$ larger than the function value of the snapshot point (section 4.5). This is the reason why we run two sequences for each super-epoch, one perturbed and one unperturbed (line 18-19 in algorithm 3). The intuition is that it is possible for a point to have a large function value increase after perturbation, but if that happens, the function value will also decrease quickly in a small number of iterations. More precisely, if the increase induced by the perturbation is large, then the perturbation has a large projection on the direction of eigenvectors of the Hessian with large eigenvalues. This implies that the point after the perturbation will decrease quickly in function value as well in a small number of iterations.

One step in proving that the function value decrease is large in the intuition above is to bound the variance. In case (2) we bound the variance either using distance or using the function value decrease depending on the relationship of n and ϵ (section 4.4.1). This causes a problem since the function value decrease we require is $b\epsilon^{1.5}$ and is larger than $\mathscr{F}$, which again gives r a lower bound even larger than $\sqrt{b}\epsilon^{\frac{3}{4}}$.

Therefore, we need to use an alternative bound at the beginning of the analysis to show that the function value decreases. We divide the analysis into **two phases**: in the first phase we use distance to bound the variance; in the second phase we use either distance or function value decrease to bound the variance depending on the relationship of n and ϵ (section 4.4.1). In this way we can show that after the perturbation, the function value decreases fast within the first phase. The **two-phases analysis** allows us to use different quantities to upper bound the variance in a consistent way.

We apply the **two-phases analysis** to the sequence of perturbed point: the first phase has a small number of iterations $2b$, whose function value increase after perturbation

and $2b$ iterations can be upper bounded by $O(\epsilon^{1.5})$; the second phase we analyze one super-epoch and shows the function value decreases by at least $\Omega(\epsilon^{1.5})$ (section 4.6).

4.2 Some Key Lemmas

In this section, we present several bounds that are frequently used in the following proof sketch. Note that there are four quantities relevant to the proof: the accumulation of variance-reduced gradient, the accumulation of variance, the accumulation of distance from one point to a previous point in the sequence, and the function value decrease. We exploit their relationships and use l-smoothness to prove several key bounds.

In particular, the following lemmas assume that in algorithm 3, f is l-smooth and $\forall i$, f_i are l-smooth. Further, the step size $\eta = \frac{1}{l}$.

4.2.1 Upper Bounding Variance by Distance

From the definition of g_t^s and l-smoothness of individual function, we obtain

Lemma 4.1. *Within an epoch s, we can upper bound the variance by the distance from a point x_t^s to the starting point of the epoch $\widetilde{x}^{s-1}$: i.e. for any epoch s and iteration t, we have*

$$\mathbb{E}[\| g_t^s \|^2] \leq \frac{l^2}{b} \| x_t^s - \widetilde{x}^{s-1} \|^2$$

Lemma 4.1 will be important when we try to find an upper bound for accumulated variance to do two-points analysis in section 4.4 and section 4.5.

4.2.2 Upper Bounding Distance by Function Value Decrease

We adapt the analysis in [LL18] to bound the accumulative distance by function value decrease and give the following lemma:

Lemma 4.2. *Within an epoch s, we can lower bound the function value decrease by the accumulation of distances from one point x_t^s to the starting point of this epoch $\widetilde{x}^{s-1}$: i.e. for any epoch s, we have*

$$\mathbb{E}[\sum_{t=1}^{m} \frac{l}{3t^2} \|x_t^s - \widetilde{x}^{s-1}\|^2] \leq \mathbb{E}[f(\widetilde{x}^{s-1}) - f(\widetilde{x}^s)]$$

which also implies

$$\mathbb{E}[\sum_{t=1}^{m} \|x_t^s - \widetilde{x}^{s-1}\|^2] \leq \frac{3m^2}{l}\mathbb{E}[f(\widetilde{x}^{s-1}) - f(\widetilde{x}^s)]$$

Lemma 4.2 is useful since we can upper bound the changing Hessian term by distance, and therefore we will need to further upper bound the distance by function value decrease in section 4.4 and section 4.5. Additionally, we can revise the proof of Lemma 4.2 to bound distance between two epochs:

Lemma 4.3. *Between two epochs $s-1$ and s, the function value decrease within the epoch s is lower bounded by the distance between the last points of the two epochs: i.e. for any epoch s, we have*

$$\mathbb{E}[\|\widetilde{x}^s - \widetilde{x}^{s-1}\|^2] \leq \mathbb{E}[\frac{4m}{5L}\big(f(\widetilde{x}^{s-1}) - f(\widetilde{x}^s)\big)]$$

Lemma 4.3 provides a convenient bound to use when we are dealing with distances of points in two adjacent epochs in section 4.4 and section 4.5.

Combining Lemma 4.1 and Lemma 4.2, we can bound the accumulative variance by function value decrease:

Lemma 4.4. *Within an epoch, the function value decrease of the epoch is lower bounded by the accumulation of the variance: i.e. for any epoch s, we have*

$$\mathbb{E}[\sum_{t=1}^{m} \parallel g_t^s \parallel^2] \leq \frac{3lm^2}{b}\mathbb{E}[f(\widetilde{x}^{s-1}) - f(\widetilde{x}^s)]$$

Together with Lemma 4.1, Lemma 4.4 gives a nice upper bound for variance in different circumstances when we bound the width of the stuck region (Lemma 4.7): they handle different situations depending on the relationship of n and ϵ.

4.2.3 Upper Bounding Gradient by Function Value Decrease

Using l-smoothness, we obtain the following bound:

Lemma 4.5. *Within an epoch, the function value decrease between two points x_t and x_{t+1} is lower bounded by the norm of the gradient of x_t minus the norm of the variance of x_t: i.e. for any epoch s and iteration t, we have*

$$\mathbb{E}[f(x_t^s) - f(x_{t+1}^s)] \geq \frac{1}{2l} \parallel \nabla f(x_t^s) \parallel^2 - \frac{1}{2l}\mathbb{E}[\parallel g_t^s \parallel^2]$$

Lemma 4.5 will be important when we try to lower bound the function value decrease using gradient to show that with large gradient, the function value decrease will be large (see section 4.3).

4.3 Exploiting Large Gradient

Recall that x with gradient $\| \nabla f(x) \| > \epsilon$ is not a local minimum. Therefore, we would like to use this property to show function value decreases.

Combining Lemma 4.4 and Lemma 4.5 by telescoping Lemma 4.5, we may lower bound the function value decrease by the accumulated gradient:

Lemma 4.6. *For any epoch s, we have*

$$\mathbb{E}[f(\widetilde{x}^{s-1}) - f(\widetilde{x}^{s})] \geq \frac{1}{2l} \frac{1}{1 + \frac{3m^2}{2b}} \sum_{t=1}^{m} \| \nabla f(x_t^s) \|^2$$

Since we let $3m^2 \leq 2b$, $\mathbb{E}[f(\widetilde{x}^{s-1}) - f(\widetilde{x}^{s})] \geq \frac{1}{4l} \sum_{t=1}^{m} \| \nabla f(x_t^s) \|^2$. Therefore, if the average gradient from x_0^s to x_m^s is greater than ϵ,the function value decrease is lower bounded by $m\epsilon^2$. Since the number of epochs in a super-epoch is given by $\frac{t_{thres}}{m}$, the function value decrease in each super-epoch is $t_{thres}\epsilon^2 = O(\epsilon^{1.5})$.

4.4 Exploiting Negative Curvature

If we do not have a large average gradient for points in a super-epoch, then the fact that the points are local minimums implies they have large negative curvature: their Hessians $\nabla^2 f(x)$ have large negative eigenvalues.

We follow the two-point analysis framework in [JGN+17] to prove how we can lower bound the function value decrease given a large negative curvature. For notation convenience, we consider the sequence of the whole super-epoch indexing only by t, where $t \leq t_{thres}$. Assume x_{init} is not a local minimum and does not have a large gradient, then $\lambda_{min}(x_{init}) < -\sqrt{\rho\epsilon}$.

Lemma 4.7. *Let $\lambda_{min}(x_{init}) < -\sqrt{\rho\epsilon}$ .The parameters of algorithm 3 are chosen as $r = \sqrt{b}\epsilon^{\frac{3}{4}}$, $b = n^{\frac{3}{4}}$, $m = n^{\frac{1}{4}}$, $t_{thres} = O(\frac{log(\frac{\sqrt{d}}{b^{0.5}\epsilon^{0.25}})}{\eta\epsilon^{0.5}})$ and $\mathscr{F} = \epsilon^{1.5}$.*

Consider two starting points of a super-epoch: x_0 and x_0' with distance r away from the end point of last epoch x_{init} and $w_0 = x_0 - x_0' = r_0 \boldsymbol{e}_1$, where $r_0 \leq C\frac{r}{\sqrt{d}}$ for some constant

C. Then we have

$$min\{\mathbb{E}[f(x_{t_{thres}})] - f(x_0), \mathbb{E}[f(x'_{t_{thres}})] - f(x_0)\} \leq -\mathscr{F}$$

Define $\chi_{stuck} = \{x_0 | x_0 \in \mathbb{B}^d_{x_{init}}(r) \text{and} f(x_{t_{thres}}) - f(x_0) \geq -\frac{\mathscr{F}}{2}\}$, then the probability for the perturbed point with perturbation radius r not escaping the saddle point is $\frac{\chi_{stuck}}{\mathbb{B}^d_0(r)}$.

Suppose Lemma 4.7 is true and **works as a high probability bound** as well, we can bound χ_{stuck} by its width on the $\mathbf{e}_1$ direction: r_0, which gives

$$\text{Vol}(\chi_{stuck}) \leq r_0 \text{Vol}(\mathbb{B}^{d-1}_0(r))$$

Recall the volume of ball with radius r in d dimensional Euclidean space can be given by Γ function: $\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{n}{2}+1)} r^d$. We can bound $\frac{\chi_{stuck}}{\mathbb{B}^d_0(r)}$ as follows:

$$\begin{aligned}
\frac{\chi_{stuck}}{\mathbb{B}^d_0(r)} &\leq \frac{r_0 \text{Vol}(\mathbb{B}^{d-1}_0(r))}{\text{Vol}(\mathbb{B}^d_0(r))} \\
&= \frac{r_0 \Gamma(\frac{d}{2}+1)}{r\sqrt{\pi}\Gamma(\frac{d}{2}+\frac{1}{2})} \\
&\leq \frac{r_0}{r\sqrt{\pi}} \sqrt{\frac{d}{2}+\frac{1}{2}} \leq 2C
\end{aligned}$$

Therefore, with probability $1 - 2C$, the perturbed point decrease $\mathscr{F}$ in function value decrease.

4.4.1 Proof Sketch to Bound Width of Stuck Region

Now we give the proof sketch for Lemma 4.7. We note that the current proof idea gives the bound in expectations, and **we need to change it to a high probability bound**.

Let $w_t = x_t - x'_t$ and assume contradiction: if $\min\{\mathbb{E}[f(x_{t_{thres}})] - f(x_0), \mathbb{E}[f(x'_{t_{thres}})] - f(x_0)\} > -\mathscr{F}$, using Lemma 4.3, we can bound $\max\{\mathbb{E}[\| x_{t_{thres}} - x_0 \|], \mathbb{E}[\| x'_{t_{thres}} - x'_0 \|]\}$ by $\sqrt{\frac{1}{L}}\sqrt{\epsilon}$. Since $\| x_0 - x_{init} \| = \| x'_0 - x_{init} \| = \sqrt{b}\epsilon^{\frac{3}{4}}$, we can upper bound w_t by

$$\begin{aligned}
\mathbb{E}[\| w_t \|] &\leq \mathbb{E}[\| x_t - x_{init} \| + \| x'_t - x_{init} \|] \\
&\leq \mathbb{E}[(\| x_t - x_0 \| + \| x_0 - x_{init} \|) + (\| x'_t - x'_0 \| + \| x'_0 - x_{init} \|)] \\
&\leq (2\sqrt{b}\epsilon^{0.25} + \sqrt{\frac{2}{L}})(\sqrt{\epsilon})
\end{aligned}$$

Let $\mathcal{H} = \nabla^2 f(x_{init})$, then we can inductively write w_{t+1} as

$$w_{t+1} = (I - \eta\mathcal{H})^{t+1} w_0 - \eta \sum_{\tau=0}^{t} (I - \eta\mathcal{H})^{t-\tau} (\triangle_\tau w_\tau + g_\tau)$$

where $g_\tau = g_t^{\lfloor \frac{t}{m} \rfloor}(x_t) - g_t^{\lfloor \frac{t}{m} \rfloor}(x'_t)$, the difference of the variance of x_t and x'_t, and $\triangle_\tau = \int_0^1 [\nabla^2(x'_t + \theta(x_t - x'_t)) - \mathcal{H}] d\theta$.

The intuition here is that w_0 is on $\mathbf{e}_1$ direction, which means $(I - \eta\mathcal{H})^t w_0 = (1 + \eta\rho^{0.5}\epsilon^{0.5})^t w_0$, and the the norm of w_t grows exponentially with t. We can upper bound $\| \sum_{\tau=0}^{t} (I - \eta\mathcal{H})^{t-\tau} \triangle_\tau w_\tau \|$ and $\| \sum_{\tau=0}^{t} (I - \eta\mathcal{H})^{t-\tau} g_\tau \|$ by $\| (I + \eta\epsilon^{0.5})^t w_0 \|$ respectively.

The first term can be bounded since $\triangle_\tau$ can be bounded using the ρ-Hessian Lipschitz condition that upper bounds $\triangle_\tau$ by distance. Then we can apply lemma 4.2 to upper bound distance by function value decrease. In order to separate $\| \Delta_\tau w_\tau \|$ to $\| \Delta_\tau \| \| w_\tau \|$, we need a **high probability bound**.

The analysis of the second term depends on the relationship of n and ϵ. If $n^{\frac{3}{4}} \geq \frac{1}{\sqrt{\epsilon}}$, we can bound $\| g_\tau \|$ by $\frac{l^2}{n^{\frac{3}{4}}} \| w_\tau \|$ using an alternative version of Lemma 4.1. Otherwise, we upper bound $\| g_\tau \|$ by $\mathbb{E}[\| g_t^{\lfloor \frac{t}{m} \rfloor}(x_t) \|] + \mathbb{E}[\| g_t^{\lfloor \frac{t}{m} \rfloor}(x'_t) \|]$, where each variance can be upper bounded by function value decrease using Lemma 4.4.

Therefore, if we let t to be large so that $(1 + \eta\epsilon^{0.5})^t > 2\sqrt{d} + \frac{\sqrt{2d}}{\sqrt{Lb}\epsilon^{0.5}}$, we have a contradiction since $\mathbb{E}[\| w_t \|] \geq (1 + \eta\epsilon^{0.5})^t r_0 > (2\sqrt{b}\epsilon^{0.25} + \sqrt{\frac{2}{L}})(\sqrt{\epsilon}) \geq \mathbb{E}[\| w_t \|]$. Here, $t < O(\frac{log(\frac{\sqrt{d}}{b\epsilon})}{\epsilon})$

4.5 Bounding Increase Induced by Perturbations

As noted previously, the perturbation might induce an increase in function value. Since we hope to reach a function value decrease by $\epsilon^{1.5}$ compared to the unperturbed point, we need to bound the possible increase induced by the perturbation: $\mathbb{E}[f(\hat{x}^s)] - f(\tilde{x}^s)$. For notation convenience, we consider the sequence of the whole super-epoch indexing only by t, where $t \leq t_{thres}$. The sequence of unperturbed point is $\{\tilde{x}_0, ..., \tilde{x}_{t_{thres}}\}$ and the sequence of perturbed point is $\{\hat{x}_0, ..., \hat{x}_{t_{thres}}\}$. The initial point $x_{init} = \tilde{x}_0$.

4.5.1 Proof Intuitions

Instead of directly bounding $\mathbb{E}[f(\hat{x}_0)] - f(\tilde{x}_0)$, we bound $\mathbb{E}[f(\hat{x}_{2b})] - f(\tilde{x}_0)$. The intuition originates from considering in what circumstances will the function value increase induced by perturbation is large. Using Taylor expansion, we get

$$f(x_{init} + \zeta^s) - f(x) \approx \nabla f(x)^T \zeta^s + \zeta^{sT} \nabla^2 f(x) \zeta^s$$

If the function value increase induced by ζ^s is large (left-hand-side is large) and the gradient at the snapshot point before perturbation (i.e. x_{init}) is small ($\nabla f(x)^T \zeta^s$ is small), term $\zeta^{sT} \nabla^2 f(x) \zeta^s$ has to be large. This implies that the perturbation ζ^s need to have a large projection on the direction of the eigenvectors of the Hessian $\nabla^2 f(x_{init})$ with large positive eigenvalues. Then if we track the difference of $\hat{x}_t - \widetilde{x}_t$, which is ζ^s initially, its norm will shrink exponentially fast. Therefore, $\mathbb{E}[f(\hat{x}_t)] - \mathbb{E}[f(\widetilde{x}_t)]$ will be small as $\| \hat{x}_t - \widetilde{x}_t \|$ shrinks.

4.5.2 Proof Sketch

Based on the intuition, we construct the analysis as follows: we break $\mathbb{E}[f(\hat{x}_{2b})] - f(\tilde{x}_0)$ into two parts: $\mathbb{E}[f(\hat{x}_{2b})] - \mathbb{E}[f(\tilde{x}_{2b})]$ and $\mathbb{E}[f(\tilde{x}_{2b})] - f(\tilde{x}_0)$. By Lemma 4.6, we can upper bound $\mathbb{E}[f(\tilde{x}_{2b})] - f(\tilde{x}_0)$ by zero: $\mathbb{E}[f(\tilde{x}_{2b})] - f(\tilde{x}_0) \leq 0$.

We give the following lemma to bound $\mathbb{E}[f(\hat{x}_{2b})] - \mathbb{E}[f(\tilde{x}_{2b})]$:

Lemma 4.8. *Suppose* $\mathbb{E}[f(\tilde{x}_{2b})] - f(\tilde{x}_0) \geq -\epsilon^{1.5}$*, then for some constant* C*, we have*

$$\mathbb{E}[f(\hat{x}_{2b})] - \mathbb{E}[f(\tilde{x}_{2b})] \leq C\epsilon^{1.5}$$

If Lemma 4.8 is true, we can bound the function value increase by $O(\epsilon^{1.5})$ when the sequence $\{\tilde{x}_t\}$ does not decrease the function value to the target $\Omega(\epsilon^{1.5})$. This implies that the sequence $\{\hat{x}_t\}$ will decrease the function value by $\Omega(\epsilon^{1.5})$ from the discussions in section 4.4 and 4.5 when the sequence $\{\tilde{x}_t\}$ does not decrease the function value to the target $\Omega(\epsilon^{1.5})$, which suffices to prove that in a super-epoch, algorithm 3 decrease the function value by $\Omega(\epsilon^{1.5})$.

We give the proof idea for Lemma 4.8. Let $\mathcal{H} = \nabla^2 f(\tilde{x}_0)$. We use Taylor Expansion of $\mathbb{E}[f(\hat{x}_{2b})] - \mathbb{E}[f(\tilde{x}_{2b})]$,

$$\begin{aligned}\mathbb{E}[f(\hat{x}_{2b})] - \mathbb{E}[f(\tilde{x}_{2b})] &\leq \mathbb{E}[\nabla f(\tilde{x}_{2b})^T w_{2b}] + \frac{1}{2}\mathbb{E}[w_{2b}^T \mathcal{H} w_{2b}] + \frac{1}{2}\mathbb{E}[w_{2b}^T(\nabla^2 f(\tilde{x}_{2b}) - \mathcal{H})w_{2b}] \\ &\quad + \frac{\rho}{6}\mathbb{E}[\| w_t \|^3] \\ &\leq \mathbb{E}[\| \nabla f(\tilde{x}_{2b}) \| \| w_t \|] + \frac{1}{2}\mathbb{E}[w_{2b}^T H w_{2b}] + \frac{\rho}{2}\mathbb{E}[\| w_{2b} \| \| \tilde{x}_{2b} - \tilde{x}_0 \| \| w_{2b} \|] \\ &\quad + \frac{\rho}{6}\mathbb{E}[\| w_{2b} \|^3]\end{aligned}$$

We can bound each term separately. The linear term $\mathbb{E}[\| \nabla f(\tilde{x}_{2b}) \| \| w_t \|]$ can be bounded since $\mathbb{E}[f(\tilde{x}_{2b})] - f(\tilde{x}_0) \geq -\epsilon^{1.5}$ implies that $\| \tilde{x}_{2b} \| \leq \epsilon$ on average.

$\| w_{2b} \|$ can be bounded by a similar induction as the proof idea for Lemma 4.7: $\| w_{2b} \| \leq (1+\eta\epsilon^{0.5})^{2b} \| w_0 \| \leq O(\epsilon^{0.5})$.

The distance term $\| \tilde{x}_{2b} - \tilde{x}_0 \|$ in the changing Hessian term can be bounded by $O(\epsilon^{1.5})$ using Lemma 4.3 because $\mathbb{E}[f(\tilde{x}_{2b})] - f(\tilde{x}_0) \geq -\epsilon^{1.5}$.

$\frac{1}{2}\mathbb{E}[w_{2b}^T H w_{2b}]$ can be re-written using the inductively form of w_{2b} as

$$w_{2b} = (I - \eta\mathcal{H})^{2b} w_0 - \eta \sum_{\tau=0}^{2b-1} (I - \eta\mathcal{H})^{t-\tau}(\triangle_\tau w_\tau + g_\tau)$$

The part with variance term g_τ and changing Hessian term $\triangle_\tau w_\tau$ can be bounded similarly as the proof idea of Lemma 4.7. The remaining term is

$$\begin{aligned} w_0^T (I - \eta H)^{4b} H w_0 &\leq \|w_0\| \|(I - \eta H)^{4b} H\| \|w_0\| \\ &\leq \lambda_{\max}((I - \eta H)^{4b} H) \|w_0\|^2 \\ &\leq \frac{1}{4b\eta} b\epsilon^{1.5} \\ &= \frac{\epsilon^{1.5}}{4\eta} \end{aligned}$$

which implies $w_0^T (I - \eta H)^{4b} H w_0 \leq O(\epsilon^{1.5})$.

Therefore, we can bound each term by $O(\epsilon^{1.5})$, concluding the proof.

4.6 Proof of Main Theorem (Theorem 3.2)

We combine (i.e. telescope) the super-epochs so that the function value decrease in each super-epoch accumulates. Suppose there are total M super-epochs in the T iterations,

we have

$$\Delta f = f(x_0) - f(x^S) = \sum_{i=1, M_i = i \cdot t_{thres}}^{M} f(x^{M_{i-1}}) - f(x^{M_i})$$

For each super-epoch,

$$f(x^{M_{i-1}}) - f(x^{M_i}) = \begin{cases} f(\widetilde{x}^{M_{i-1}}) - f(\widetilde{x}^{M_i}), \text{if } f(\widetilde{x}^{M_{i-1}}) - f(\widetilde{x}^{M_i}) > \epsilon^{1.5} \\ (f(\hat{x}^{M_{i-1}+\frac{2b}{m}}) - f(\hat{x}^{M_i})) + (f(x^{M_{i-1}}) - f(\hat{x}^{M_{i-1}+\frac{2b}{m}})), \text{otherwise} \end{cases}$$

In each super-epoch, the function value will decrease by at least $\Omega(\epsilon^{1.5})$ either due to a large gradient (section 4.3) or a large negative curvature (section 4.4). This implies that

$$\max(f(\widetilde{x}^{M_{i-1}}) - f(\widetilde{x}^{M_i}), f(\hat{x}^{M_{i-1}+\frac{2b}{m}}) - f(\hat{x}^{M_i})) \geq \epsilon^{1.5}$$

The potential function value increase induced by perturbation shrinks to $O(\epsilon^{1.5})$ in $\frac{2b}{m}$ epochs: $f(\hat{x}^{M_{i-1}+\frac{2b}{m}}) - f(\hat{x}^{M_i}) = O(\epsilon^{1.5})$ (section 4.5). Consequently, the function value decrease $f(x^{M_{i-1}}) - f(x^{M_i})$ is at least $\Omega(\epsilon^{1.5})$. After running $O(\frac{\Delta f}{\epsilon^{1.5}})$ super-epochs, the function value will decrease at least Δf if the algorithm has not converged to an ϵ-second-order stationary point. Since there are $\tilde{O}(\frac{1}{\eta\epsilon^{0.5}})$ iterations in each super-epoch, after running $\tilde{O}(\frac{\Delta f}{\epsilon^2})$ iterations (i.e. $O(\frac{\Delta f}{\epsilon^{1.5}})$ super-epochs), the function value decreases by at least Δf.

Chapter 5

Preliminary Works Using the Stable Manifold

Theorem 3.2 implies that the convergence rate is $\tilde{O}(\frac{1}{\epsilon^2})$. Each mini-batch calculates $b = n^{\frac{3}{4}}$ stochastic gradients, and therefore the total number of stochastic gradients calculated is $\tilde{O}(\frac{n^{\frac{3}{4}}}{\epsilon^2})$.

In this chapter, we ask the question if we can have a better bound: if we can reduce the number of stochastic gradients in each mini-batch step to $b = n^{\frac{2}{3}}$. The loss of escaping saddle point will be polylog of n, dimension d and ϵ, as suggested by Theorem 2.9's convergence rate $O(\frac{n^{\frac{2}{3}}}{\epsilon^2})$.

To start with, if we change $b = n^{\frac{3}{4}}$ to $b = n^{\frac{2}{3}}$ and $m = n^{\frac{1}{4}}$ to $m = n^{\frac{1}{3}}$, we observe that the only place where the previous proof does not hold is the proof of Lemma 4.7. Specifically, the part where we try to bound $\| \sum_{\tau=0}^{t}(I-\eta\mathcal{H})^{t-\tau}g_\tau \|$ by $\| (I+\eta\epsilon^{0.5})^t w_0 \|$ does not hold if $d \geq n^{\frac{2}{3}}$ and $n^{\frac{2}{3}} \leq \frac{1}{\sqrt{\epsilon}}$.

The intuition for why this happens is that the projection of w_t on the subspace orthogonal to $\mathbf{e}_1$ can grow as fast as its projection on $\mathbf{e}_1$. This happens as $\sum_{\tau=0}^{t}(I-\eta\mathcal{H})^{t-\tau}g_\tau$ accumulates with t, and we have no control over the direction of $\sum_{\tau=0}^{t}(I-\eta\mathcal{H})^{t-\tau}g_\tau$. As t grows, the norm $\| \sum_{\tau=0}^{t}(I-\eta\mathcal{H})^{t-\tau}g_\tau \|$ catches up with $\| (I+\eta\epsilon^{0.5})^t w_0 \|$, obtaining the same or even larger magnitude. Unfortunately, we have no bound on its projection on a given direction, so we can only assume the worst case scenario to bound the magnitude by $(\sum_{\tau=0}^{t}(I+\eta\epsilon^{0.5})^{t-\tau}) \| g_t \|$, and it may as well be the magnitude of the projection on the subspace orthogonal to $\mathbf{e}_1$ in some worst case scenarios. This implies that the projection of w_t on the subspace orthogonal to $\mathbf{e}_1$ may have greater magnitude than the projection of w_t on $\mathbf{e}_1$ as t grows, which breaks the induction hypothesis that $(I+\eta\epsilon^{0.5})^t w_0$ is the leading term in the previous proof.

Therefore, we are seeking a way that change the two-points analysis framework so that we can track some quantity whose projection on the subspace with large negative eigenvalue grows exponentially. We also need to be able to control the growth of the the quantity's projection on the subspace orthogonal to the subspace with large negative eigenvalue. To avoid the accumulation of variance, we hope to "re-align" the quantity we are tracking so that it will always be parallel to $\mathbf{e}_1$. In this new framework, we also propose a different way to bound the accumulation of variance $\| g_\tau \|$ when $n^{\frac{2}{3}} \leq \frac{1}{\sqrt{\epsilon}}$.

5.1 Stable Manifold of Gradient Descent Map

To implement the intuition, we consider a stable manifold.

5.1.1 Definitions

Let map $F : \mathbb{R}^d \to \mathbb{R}^d$ be the gradient map of f with step size η: $F(x) = x - \eta\nabla f(x)$. We further define T_{x_0} as the local approximation of F: $T_{x_0}(x) = \nabla F(x_0)(x - x_0)^T + F(x_0) = (I - \eta\nabla^2 f(x_0))(x - x_0)^T + F(x_0)$. Let $\mathcal{H} = \nabla^2 f(x_0)$, $T_{x_0}(x) = (I - \eta\mathcal{H})(x - x_0)^T + F(x_0)$.

Suppose there is a saddle point x^* of f, we consider $T = T_{x^*}$. Without loss of generality, assume $x^* = \mathbf{0}$ and $f(x^*) = 0$, then $T(x) = (I - \eta\mathcal{H})x^T$. Suppose we can separate $\mathbb{R}^d$ as the direct sum of two subspaces E_s and E_u, $\mathbb{R}^d = E_s \oplus E_u$, and T is **hyperbolic** with respect to this separation. In other words, T contracts E_s and expands E_u: there exists θ, $0 < \theta < 1$, such that

$$\| T|_{E_s} \| < \theta \text{ and } \| (T|_{E_u})^{-1} \| < \theta$$

where $T|_{E_s}$ is the projection of T on domain E_s and $T|_{E_u}$ is the projection of T on domain E_u. We call E_s the **stable subspace** of F at $\mathbf{0}$ and E_u the **unstable subspace** of F at $\mathbf{0}$.

We note that in our case, the requirement indicates that there is no 0 eigenvalue for $\mathcal{H}$ and there is a separation of the eigenvalue of $\mathcal{H}$: the positive eigenvalues $\lambda^+ > \frac{1-\theta}{\eta}$ and negative eigenvalues $\lambda^- < \frac{1-\frac{1}{\theta}}{\eta}$.

Let $E_s(r)$ and $E_u(r)$ be the ball around $\mathbf{0}$ with radius r in E_s and E_u respectively.

Definition 5.1. A stable set of F is the set $W^s(F) = \{x \in E_s(r) \times E_u(r) | F^n(x) \to \mathbf{0} \text{ as } n \to \infty\}$

In other words, the stable set is the set of points in $\mathbb{B}_0^d(r)$ that converge to the saddle point running gradient descent.

Definition 5.2. Given function f that is l-smooth, we let $Lip(f) = l$, the Lipschitz constant.

5.1.2 The Stable Manifold Theorem

Now we are ready to introduce the stable manifold from a modified version of the Stable Manifold Theorem in [SFL87]:

Theorem 5.3. *If there exists ϵ such that $Lip(F - T) < \epsilon < 1 - \theta$ and a $\delta > 0$ such that $\| F(0) \| < \delta$, then $W^s(F)$ is the graph of a function $\phi : E_s(r) \to E_u(r)$. We call $W^s(F)$ the stable manifold of F. In addition, F is C^k implies ϕ is C^k.*

The proof of Theorem 5.3 follows from [SFL87].

Intuitively, Theorem 5.3 tells us that if T is **a good approximation** of F in the local ball $\mathbb{B}_0^d(r)$ and there is **a nice separation of the eigenvalues** of $\mathcal{H}$, we have a smooth manifold and it is as smooth as F (i.e. $\nabla f(\cdot)$) with respect to differentiation.

5.2 Proof Sketch with $\frac{1}{\sqrt{\epsilon}}$-Smooth Assumption

This section shows how we can reduce the extra $O(n^{\frac{1}{12}})$ losses in the original analysis. We need to assume the following:

Conjecture 5.4. Assume the snapshot x_{init} point be a saddle point. Let $W^s(F)$ be the stable manifold of x_{init}. Assume the graph ϕ of $W^s(F)$ is $\frac{1}{\sqrt{\epsilon}}$-smooth.

Under conjecture 5.4, we may give a new proof for lemma 4.7 without the $O(n^{\frac{1}{12}})$ loss.

For point $x \in R^d$, let x^M denote the projection of x on the stable manifold $M = W^s(F)$ along the E_u subspace. Instead of tracking two sequences and track their difference w_t, now we consider one sequence $\{x_t\}$, and track the quantity $x_t - x_t^M$. In this way, $x_t - x_t^M$ is always in the union of eigenspaces of $\mathcal{H}$ with large negative eigenvalues. This indicates that $\| (I - \eta\mathcal{H})^t (x_t - x_t^M) \| \geq (1 + \eta\theta)^t \| x_t - x_t^M \|$.

Let $u_t = x_t - x_t^M$. We hope to get a lower bound on $\mathbb{E}[\| u_{t+1} \|]$. We may re-write u_{t+1} as follows:

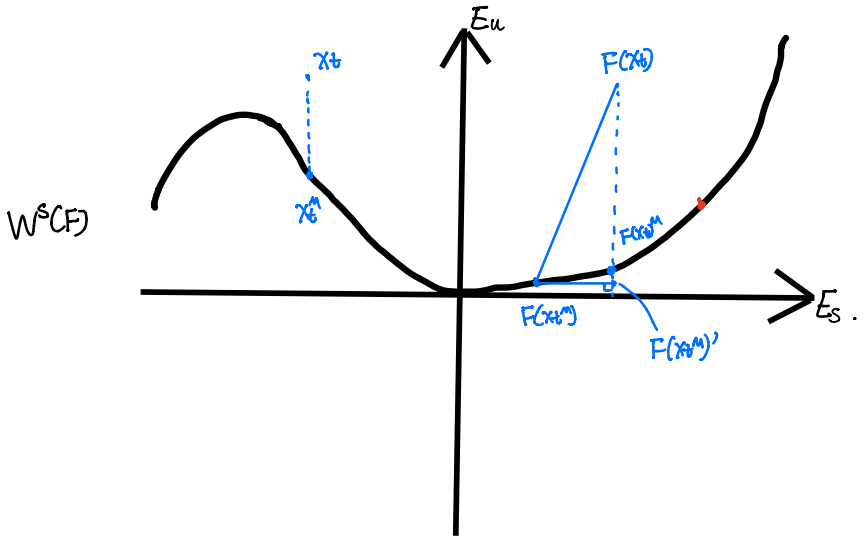

FIGURE 5.1: From $F(x_t) - F(x_t^M)$ to $F(x_t) - F(x_t)^M$

$$\begin{aligned} u_{t+1} &= x_{t+1} - x_{t+1}^M \\ &= (x_{t+1} - F(x_t)) + (F(x_t) - F(x_t)^M) + (F(x_t)^M - x_{t+1}^M) \\ &= (x_{t+1} - F(x_t)) + (F(x_t) - F(x_t^M)) + (F(x_t^M) - F(x_t)^M) + (F(x_t)^M - x_{t+1}^M) \end{aligned}$$

Let $\mathcal{P}_u$ be the projection operator of subspace E_u. By definition $\mathcal{P}_u u_{t+1} = u_{t+1}$. Since $\mathcal{P}_u$ is a linear operator, we have

$$\begin{aligned} u_{t+1} &= \mathcal{P}_u u_{t+1} \\ &= \mathcal{P}_u((x_{t+1} - F(x_t)) + (F(x_t) - F(x_t^M)) + (F(x_t^M) - F(x_t)^M) + (F(x_t)^M - x_{t+1}^M)) \\ &= \mathcal{P}_u(x_{t+1} - F(x_t)) + \mathcal{P}_u(F(x_t) - F(x_t^M)) + \mathcal{P}_u(F(x_t^M) - F(x_t)^M) + \mathcal{P}_u(F(x_t)^M - x_{t+1}^M) \\ &= \mathcal{P}_u(\eta g_t) + (I - \eta\mathcal{H})u_t + \mathcal{P}_u(F(x_t^M) - F(x_t)^M) + \mathcal{P}_u(F(x_t)^M - x_{t+1}^M) \\ &= (I - \eta\mathcal{H})u_t + \mathcal{P}_u(F(x_t^M) - F(x_t)^M) + (\mathcal{P}_u(\eta g_t) + \mathcal{P}_u(F(x_t)^M - x_{t+1}^M)) \end{aligned}$$

We observe that $\mathcal{P}_u(F(x_t^M) - F(x_t)^M)$ is induced by the change of Hessian and is similar to the term $\Delta_t w_t$ in the previous two-points analysis. In addition, $\mathcal{P}_u(\eta g_t) + P_u(F(x_t)^M - x_{t+1}^M)$ is induced by the variance g_t and is similar to the variance term in the previous two-points analysis. Intuitively, we may break the analysis into two parts: we first align $F(x_t) - F(x_t^M)$ to $F(x_t) - F(x_t)^M$ (FIGURE 5.1), which induces a changing Hessian

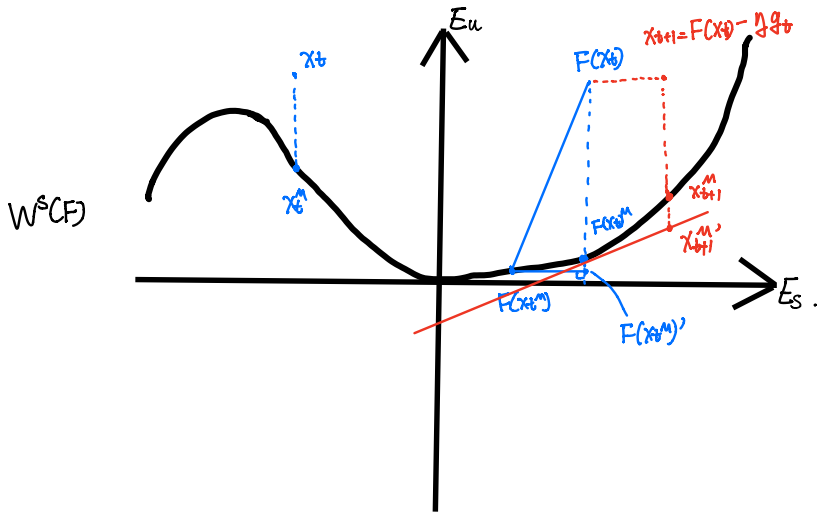

FIGURE 5.2: From $F(x_t) - F(x_t^M)$ to $x_{t+1} - x_{t+1}^M$

term; then we align $F(x_t) - F(x_t^M)$ to $x_{t+1} - x_{t+1}^M$ (FIGURE 5.2), which induces a variance term.

We hope to show that $(I - \eta\mathcal{H})u_t$ is the dominating term of u_{t+1} in the alignment process by showing that the norm of the other terms is $O(\sqrt{\epsilon} \parallel u_t \parallel)$. This requires two properties of the graph ϕ of the stable manifold $W^s(F)$: 1-Lipschitz and $\frac{1}{\sqrt{\epsilon}}$-smooth.

First, we align $F(x_t) - F(x_t^M)$ to $F(x_t) - F(x_t)^M$ (FIGURE 5.1). The loss is given by

$$\parallel \mathcal{P}_u(F(x_t^M) - F(x_t)^M) \parallel = \parallel F(x_t)^M - F(x_t^M)' \parallel$$

where $F(x_t^M)'$ is the projection of $F(x_t^M)$ along the direction of E_u on the line orthogonal to E_u passing $F(x_t)$ and $F(x_t)^M$. Using the 1-Lipschitz property of ϕ, we can upper bound $\parallel F(x_t)^M - F(x_t^M)' \parallel$ by $\parallel F(x_t)^M - F(x_t^M) \parallel$. Since $\parallel F(x_t)^M - F(x_t^M) \parallel$ is upper bounded by the norm of the changing Hessian term, we can bound it as the proof of Lemma 4.7.

Second, we align $F(x_t) - F(x_t^M)$ to $x_{t+1} - x_{t+1}^M$ (FIGURE 5.2). Let Q be the tangent plane of ϕ passing $F(x_t)^M$, and let ${x_{t+1}^M}'$ be the intersection of Q and the line orthogonal to E_u passing x_{t+1} and x_{t+1}^M. In a 2-D case as FIGURE 5.2, Q is the red line tangent to $W^s(F)$ at $F(x_t)^M$. Since $\mathbb{E}[g_t] = 0$, $\mathbb{E}[x_{t+1}] = F(x_{t+1})$, and therefore the alignment of $F(x_t)$ to x_{t+1} (i.e. $P_u(\eta g_t)$) has zero norm in expectations. The alignment from $F(x_t)^M$ to x_{t+1}^M gives loss

$$\parallel P_u(F(x_t)^M - x_{t+1}^M) \parallel = \parallel x_{t+1}^M - {x_{t+1}^M}' \parallel$$

which can be bounded by the $\frac{1}{\sqrt{\epsilon}}$-smoothness of ϕ: it is upper bounded by $\frac{1}{\sqrt{\epsilon}} \parallel g_t \parallel^2$. This solves the original problem because we are now dealing the square of $\parallel g_t \parallel$, which is much smaller than $\parallel g_t \parallel$.

Bibliography

[AAZB+17] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 1195–1199, New York, NY, USA, 2017. ACM.

[AZH16] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 699–707. JMLR.org, 2016.

[AZL17] Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles, 2017.

[CDHS16] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for non-convex optimization, 2016.

[CHDS17] Yair Carmon, Oliver Hinder, John C. Duchi, and Aaron Sidford. "convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions, 2017.

[DPG+14] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, 2014.

[GHJY15] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition, 2015.

[GL16] Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for non-convex nonlinear and stochastic programming. *Mathematical Programming*, 156(1):59–99, Mar 2016.

[GLM16] Rong Ge, Jason D. Lee, and Tengyu Ma. Matrix completion has no spurious local minimum, 2016.

[JG17] Chi Jin and Rong Ge. `SGD` finds second-order stationary point in polylog(d) iterations. Unpublished Script, 2017.

[JGN+17] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently, 2017.

[JJKN15] Prateek Jain, Chi Jin, Sham M. Kakade, and Praneeth Netrapalli. Global convergence of non-convex gradient descent for computing matrix square-root, 2015.

[JZ13a] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc., 2013.

[JZ13b] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 315–323, USA, 2013. Curran Associates Inc.

[Kaw16] Kenji Kawaguchi. Deep learning without poor local minima, 2016.

[Lev16] Kfir Y. Levy. The power of normalization: Faster evasion of saddle points, 2016.

[LL18] Zhize Li and Jian Li. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization, 2018.

[Nes04] Yurii Nesterov. *Introductory lectures on convex optimization : a basic course*. Boston Kluwer Academic Publishers, 2004.

[NJS15] P. Netrapalli, P. Jain, and S. Sanghavi. Phase retrieval using alternating minimization. *IEEE Transactions on Signal Processing*, 63(18):4814–4826, Sept 2015.

[NP06] Yurii Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, Aug 2006.

[SFL87] M. Shub, A. Fathi, and R. Langevin. *Global stability of dynamical systems*. Springer-Verlag, 1987.

[SQW16] J. Sun, Q. Qu, and J. Wright. A geometric analysis of phase retrieval. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2379–2383, July 2016.