Stochastic Trust Region Algorithms

by

Rui Shi

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

Lehigh University

May 2020

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dissertation Advisor

Committee Members:

_____
Frank E. Curtis, Committee Chair

_____
Daniel P. Robinson

_____
Katya Scheinberg

_____
Martin Takáč

Contents

List of Figures

Abstract

In recent years, data science has played a significant role in how people make decisions. Properly trained deep learning models enable computers to perform certain tasks better than some human beings, such as face recognition, reading comprehension, and the game Go. More efficient algorithms are needed for training large deep learning models in order to get better performance. The purpose of this dissertation is to design efficient algorithms for solving large scale machine learning problems with good convergence properties. We borrow ideas from the well-known class of trust region algorithms in deterministic nonlinear optimization and generalize them to stochastic settings. Moreover, we show that our new algorithms in stochastic settings have convergence results on par with state-of-the-art algorithms and exhibit better numerical performance.

Chapter 1

Introduction

In the past decade, data availability is booming. Indeed, in various settings, datasets accumulate much faster than computers are able to analyze them. In recent years, with the growth of computing power, data science has become more important for the development of science and technology. It is an emerging subject in the intersection of statistics, optimization, and computer science. Data-based decision-making has a significant impact on industries such as supply chain, bioinformatics, and finance. Statistics and machine learning models, especially deep learning models, empower the decision-making process in a way that predictions are more accurate than before.

Nowadays, one important use of data is to train machine learning models. Algorithms are designed to serve that purpose. As the data become larger and more complex, more advanced algorithms are needed for dealing with the breadth and depth of the data in various contexts such as natural language processing and computer vision. There are two reasons why we need better algorithms for training models: (1) the data used to train models are becoming larger in both size and dimension and (2) more complicated models are used to deal with more difficult tasks. This highlights the need to continually improve algorithms and software related to model training so as to deal with the increasing complexity of data and models themselves.

Extensive work has been done in the area of stochastic optimization to address the need of handling data of increasing size and dimension. The stochastic gradient method (SG) is a benchmark algorithm for solving stochastic and finite-sum optimization problems.

SG is popular and works well for a variety of problems, but it suffers from high variance of the stochastic gradients it employs. To make matters worse, stochastic gradients have no natural scaling, which means the algorithm is very sensitive to hyperparameters. The challenge of designing new algorithms lies in both theoretical guarantees for convergence and practical numerical performance. The purpose of this dissertation is to design new algorithms with comparable theoretical guarantees for convergence, but consistently better practical performance than SG.

Optimization algorithms have been a subject of work for decades, and much is known about the theoretical and practical behavior that they exhibit in *small-scale* settings. However, for solving the *large-scale* data-based problems that arise today, care needs to be taken to balance computational efforts in the presence of such extraordinary amounts of data. This is especially important since, while tons of data may be available, there is also a large degree of redundancy that can lead traditional algorithms to be inefficient; e.g., see [5].

In this work, we discuss new types of stochastic algorithms to solve optimization problems where the objective function is in the form of an expectation or finite sum. Deterministic trust region algorithms are well-known for tackling nonconvex optimization problems. As is known from the deterministic case, a first-order method such as gradient descent can be written as a special case of the trust region method by defining the trust region radius appropriately. Likewise, we discuss stochastic trust region algorithms that employ careful choices of the trust region radii. We provide solid convergence proofs for these new methods and discuss their convergence rate properties under various assumptions. We also provide results of the algorithm employed to solve convex classification problems and to solve nonconvex problems to train deep neural networks.

The structure of this dissertation is as follows. Chapter 2 includes the background of SG for solving stochastic optimization problems. Chapter 3 contains the convergence analysis and numerical results of our newly proposed TRish framework. In chapter 4, we show that a generalization of our TRish framework with second order information has convergence guarantees and good numerical performance. In chapter 5, we show that an extension of TRish using momentum has good convergence properties and the algorithm has good

numerical performance. Finally, in chapter 6, we present concluding remarks.

Chapter 2

Background and Literature Review

2.1 Literature Review

In this chapter, we provide theoretical analyses for the stochastic gradient method (SG) for solving stochastic and finite-sum optimization problems. SG was first proposed in [43]; see also [44]. The algorithm works in a very intuitive way: Unlike deterministic gradient descent where the true gradient is used over all iterations, SG only uses an approximation of the true gradient in each iteration. By using approximate gradients, SG has low computing cost per iteration. This is why SG is popular for training machine learning models over large datasets.

SG has been popular for training deep neural networks for its clean mathematical behavior and satisfactory numerical performance. However, the main disadvantage of SG is that the algorithm is very sensitive to its hyperparameters, e.g, the learning rate. Significant amounts of computing power needs to be spent on tuning the learning rate before SG can be applied effectively to train deep neural networks.

Extentive research has been done to improve both theoretical analysis and numerical performance of SG, such as, [1], [6], [11], [16], [21], [23], [24], [29], [38], [36], and [56]. For most of this work, the authors provide extensions of SG that work better in practice. Moreover, they also show better theoretical results with algorithms with higher costs per iteration. Further literature reviews will be provided in chapter 3 regarding first order extensions of SG and in chapter 4 about second order extensions of SG.

2.2 Problem Description

Our problem of interest is a stochastic optimization problem in which the goal is to minimize over a vector of decision variables, indicated by $x \in \mathbb{R}^n$, a function $f : \mathbb{R}^n \to \mathbb{R}$ defined by the expectation of another function $F : \mathbb{R}^n \times \Xi \to \mathbb{R}$ that depends on a random variable ξ, i.e.,

$$\min_{x \in \mathbb{R}^n} \; f(x), \quad \text{where} \quad f(x) = \mathbb{E}_\xi[F(x, \xi)], \tag{2.1}$$

where $\mathbb{E}_\xi[\cdot]$ denotes expectation with respect to the distribution of ξ. Our algorithm is also applicable for finite-sum minimization where the objective takes the form

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

Such an objective often arises in sample average approximations of (2.1); e.g., see [48].

2.3 Stochastic Gradient Method

For reference in this dissertation, let us present known theoretical properties of the stochastic gradient method (SG) for minimizing convex and nonconvex objectives. Formally, let us consider the algorithm stated as Algorithm 1. Each iteration involves taking a step along the negative of a stochastic gradient direction. In the context of minimizing $f(x) = \mathbb{E}_\xi[F(x, \xi)]$, this stochastic gradient can be viewed as $g_k = \nabla_x F(x_k, \xi_k)$, where x_k is the current iterate and ξ_k is a realization of the random variable ξ. In the context of minimizing a finite sum, it can be viewed as $g_k = \nabla_x f_{i_k}(x_k)$ where i_k has been chosen randomly as an index in $\{1, \ldots, N\}$. In addition, in either case, g_k could represent an average of such quantities, i.e., over a set of independently generated realizations $\{\xi_{k,j}\}_j$ or over independently generated indices $\{i_{k,j}\}_j$ (leading to a so-called *mini-batch* approach). In the algorithm, we simply write $g_k \approx \nabla f(x_k)$ to cover all of these situations, since in any case g_k represents a stochastic gradient estimate for f at x_k.

Our analysis of Algorithm 1 follows that presented in [4]. To start, let us make the following common assumption.

Algorithm 1 Stochastic Gradient (SG)

1: choose an initial iterate $x_1 \in \mathbb{R}^n$ and positive stepsizes $\{\alpha_k\}$
2: **for all** $k \in \mathbb{N} := \{1, 2, \dots\}$ **do**
3: generate a stochastic gradient $g_k \approx \nabla f(x_k)$
4: set

$$x_{k+1} \leftarrow x_k - \alpha_k g_k$$

5: **end for**

Assumption 2.3.1. *The objective $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below by $f_* = \inf_{x \in \mathbb{R}^n} f(x) \in \mathbb{R}$. In addition, at any $x \in \mathbb{R}^n$, the objective is bounded above by a first-order Taylor series approximation of f at x plus a quadratic term with constant $L \in (0, \infty)$, i.e.,*

$$f(x) \leq f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) + \tfrac{1}{2} L \|x - \bar{x}\|_2^2 \quad \text{for all} \quad (x, \bar{x}) \in \mathbb{R}^n \times \mathbb{R}^n.$$

We also make the following assumption about the computed stochastic gradients.

Assumption 2.3.2. *For all $k \in \mathbb{N}$, the stochastic gradient g_k is an unbiased estimator of $\nabla f(x_k)$ in the sense that $\mathbb{E}_k[g_k] = \nabla f(x_k)$, where $\mathbb{E}_k$ denotes expectation conditioned on quantities being computed with respect to x_k. In addition, there exists a pair $(M_1, M_2) \in (0, \infty) \times (0, \infty)$ (independent of k) such that, for all $k \in \mathbb{N}$, the squared norm of g_k satisfies*

$$\mathbb{E}_k[\|g_k\|_2^2] \leq M_1 + M_2 \|\nabla f(x_k)\|_2^2.$$

Based on the assumptions above, we are able to prove the following lemma.

Lemma 2.3.3. *Under Assumptions 2.3.1 and 2.3.2, the iterates of Algorithm 1 satisfy, for all $k \in \mathbb{N}$,*

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\alpha_k \left(1 - \frac{LM_2}{2} \alpha_k\right) \|\nabla f(x_k)\|_2^2 + \frac{LM_1}{2} \alpha_k^2.$$

Proof. By Assumption 2.3.1, we have

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \nabla f(x_k)^T (x_{k+1} - x_k) + \frac{L}{2} \|x_{k+1} - x_k\|_2^2 \\ &= f(x_k) - \alpha_k \nabla f(x_k)^T g_k + \frac{L}{2} \alpha_k^2 \|g_k\|^2. \end{aligned}$$

Taking expectation on both sides, we have

$$\begin{aligned}\mathbb{E}_k[f(x_{k+1})] &\leq f(x_k) - \alpha_k \nabla f(x_k)^T \mathbb{E}_k[g_k] + \frac{L}{2}\alpha_k^2 \mathbb{E}_k[\|g_k\|^2] \\ &= f(x_k) - \alpha_k \|\nabla f(x_k)\|_2^2 + \frac{L}{2}\alpha_k^2 \mathbb{E}_k[\|g_k\|^2].\end{aligned}$$

By Assumption 2.3.2, we have

$$\begin{aligned}\mathbb{E}_k[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|_2^2 + \frac{L}{2}\alpha_k^2 (M_1 + M_2 \|\nabla f(x_k)\|_2^2) \\ &= f(x_k) - \alpha_k \left(1 - \frac{LM_2}{2}\alpha_k\right) \|\nabla f(x_k)\|_2^2 + \frac{LM_1}{2}\alpha_k^2.\end{aligned}$$

Therefore,

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\alpha_k \left(1 - \frac{LM_2}{2}\alpha_k\right) \|\nabla f(x_k)\|_2^2 + \frac{LM_1}{2}\alpha_k^2,$$

as desired. □

2.3.1 P-L condition, Fixed Step Sizes

In this section, we assume that the objective function satisfies the "P-L"condition stated as follows:

Assumption 2.3.4. *At any $x \in \mathbb{R}^n$, the Polyak-Łojasiewicz condition holds with $c \in (0, \infty)$, i.e.,*

$$2c(f(x) - f_*) \leq \|\nabla f(x)\|_2^2 \quad \textit{for all} \quad x \in \mathbb{R}^n.$$

Assumptions 2.3.1 and 2.3.4 do not ensure that a stationary point for f exists, though, when combined, they do guarantee that any stationary point for f is a global minimizer of f. Assumption 2.3.4 holds when f is c-strongly convex, but it is also satisfied for other functions that are not convex. We direct the interested reader to [30] for a discussion on the relationship between the Polyak-Łojasiewicz condition and the related *error bounds*, *essential strong convexity*, *weak strong convexity*, *restricted secant inequality*, and *quadratic growth* conditions. In short, when f has a Lipschitz continuous gradient, the Polyak-Łojasiewicz is the weakest of these except for the quadratic growth condition, though these two are

equivalent when f is convex.

Theorem 2.3.5. *Under Assumptions 2.3.1, 2.3.2, 2.3.4, suppose that Algorithm 1 is run with $\alpha_k = \alpha$ for all $k \in \mathbb{N}$ such that*

$$0 < \alpha \leq \min\left\{\frac{1}{c}, \frac{1}{LM_2}\right\}.$$

Then, for all $k \in \mathbb{N}$, the expected optimality gap satisfies

$$\mathbb{E}_k[f(x_{k+1})] - f_* \leq \frac{LM_1\alpha}{2c} + (1-c\alpha)^{k-1}\left(f(x_1) - f_* - \frac{LM_1\alpha}{2c}\right) \xrightarrow{k\to\infty} \frac{LM_1\alpha}{2c}.$$

Proof. By Lemma 2.3.3,

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\alpha\left(1 - \frac{LM_2}{2}\alpha\right)\|\nabla f(x_k)\|_2^2 + \frac{LM_1}{2}\alpha^2.$$

By strong convexity, we have

$$2c(f(x_k) - f_*) \leq \|\nabla f(x_k)\|_2^2 \quad \text{for all} \quad x \in \mathbb{R}^n.$$

Combining the above two equations, we have

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -2c\alpha\left(1 - \frac{LM_2}{2}\alpha\right)(f(x_k) - f_*) + \frac{LM_1}{2}\alpha^2.$$

Therefore,

$$\mathbb{E}_k[f(x_{k+1})] - f_* \leq \left[1 - 2c\alpha\left(1 - \frac{LM_2}{2}\alpha\right)\right](f(x_k) - f_*) + \frac{LM_1}{2}\alpha^2.$$

Since we require $\alpha \leq \frac{1}{LM_2}$, we have $1 - \frac{LM_2}{2}\alpha \geq \frac{1}{2}$. Thus,

$$\mathbb{E}_k[f(x_{k+1})] - f_* \leq (1-c\alpha)(f(x_k) - f_*) + \frac{LM_1}{2}\alpha^2.$$

Consequently, we have

$$\mathbb{E}_k[f(x_{k+1})] - f_* - \frac{LM_1\alpha}{2c} \leq (1 - c\alpha)\left(f(x_k) - f_* - \frac{LM_1\alpha}{2c}\right).$$

Since we require $0 \leq 1 - c\alpha < 1$, we have

$$\mathbb{E}_k[f(x_{k+1})] - f_* \leq \frac{LM_1\alpha}{2c} + (1 - c\alpha)^{k-1}\left(f(x_1) - f_* - \frac{LM_1\alpha}{2c}\right) \xrightarrow{k\to\infty} \frac{LM_1\alpha}{2c},$$

as desired. □

2.3.2 P-L Condition, Diminishing Step Sizes

Theorem 2.3.6. *Under Assumptions 2.3.1, 2.3.2, 2.3.4, suppose that Algorithm 1 is run with a sequence of diminishing steps sizes, such that, for all $k \in \mathbb{N}$,*

$$\alpha_k = \frac{a}{b+k} \quad \textit{for some} \quad a > \frac{1}{c} \quad \textit{and} \quad b > 0 \quad \textit{such that} \quad \alpha_1 < \frac{1}{LM_2}.$$

Then, for all $k \in \mathbb{N}$, the expected optimality gap satisfies

$$\mathbb{E}_k[f(x_k)] - f_* \leq \frac{\nu}{b+k},$$

where

$$\nu = \max\left\{\frac{a^2LM_1}{2(ac-1)}, (b+1)(f(x_1) - f_*)\right\}.$$

Proof. Similar to the proof of Theorem 2.3.5, we have

$$\mathbb{E}_k[f(x_{k+1})] - f_* \leq (1 - c\alpha_k)(f(x_k) - f_*) + \frac{LM_1}{2}\alpha_k^2.$$

We prove the result by mathematical induction.

(1) When $k = 1$, by the definition of ν, we have

$$\nu \geq (b+1)(f(x_1) - f_*).$$

Therefore, we have

$$\mathbb{E}_k[f(x_1)] - f_* \leq \frac{\nu}{b+1}.$$

(2) Assume that when $k = m$, the conclusion is true, i.e.,

$$\mathbb{E}_k[f(x_m)] - f_* \leq \frac{\nu}{b+m}.$$

Then for $k = m + 1$, we have

$$\begin{aligned} E[f(x_k)] - f_* = E[f(x_{m+1})] - f_* &\leq (1 - c\alpha_m)(f(x_m) - f_*) + \frac{LM_1}{2}\alpha_m^2 \\ &\leq (1 - c\alpha_m)\frac{\nu}{b+m} + \frac{LM_1}{2}\alpha_m^2 \\ &= (1 - \frac{ac}{b+m})\frac{\nu}{b+m} + \frac{LM_1}{2}\frac{a^2}{(b+m)^2} \\ &= \frac{(b+m-ac)\nu + a^2 LM_1/2}{(b+m)^2}. \end{aligned}$$

Since $\nu \geq a^2 LM_1/2(ac-1)$, we have

$$(b+m-ac)\nu + a^2 LM_1/2 \leq (b+m-1)\nu.$$

Therefore, we have

$$\begin{aligned} E[f(x_{m+1})] - f_* &\leq \frac{(b+m-1)\nu}{(b+m)^2} \\ &\leq \frac{(b+m-1)\nu}{(b+m)^2 - 1} \\ &= \frac{1}{b+m+1}. \end{aligned}$$

By Combining (1) and (2), the conclusion follows for all $k \in \mathbb{N}$. □

2.3.3 No P-L Condition, Fixed Step Size

Theorem 2.3.7. *Under Assumptions 2.3.1, 2.3.2, suppose that Algorithm 1 is run with $\alpha_k = \alpha$ for all $k \in \mathbb{N}$ such that*

$$0 < \alpha \leq \frac{1}{LM_2}.$$

Then the expected sum-of-squares and average-squared gradients of f satisfy the following inequalities:

$$\mathbb{E}_k\left[\sum_{k=1}^{K}\|\nabla f(x_k)\|_2^2\right] \leq K\alpha LM_1 + \frac{2(f(x_1)-f_*)}{\alpha} \tag{2.2a}$$

$$\textit{and}\ \ \mathbb{E}_k\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\|_2^2\right] \leq \alpha LM_1 + \frac{2(f(x_1)-f_*)}{\alpha K} \xrightarrow{K\to\infty} \alpha LM_1. \tag{2.2b}$$

Proof. By Lemma 2.3.3, we have

$$\mathbb{E}_k[f(x_{k+1})] - f(x_k) \leq -\alpha_k\left(1-\frac{LM_2}{2}\alpha_k\right)\|\nabla f(x_k)\|_2^2 + \frac{LM_1}{2}\alpha_k^2.$$

Taking total expectation on both sides, we have

$$\mathbb{E}_k[f(x_{k+1})] - \mathbb{E}_k[f(x_k)] \leq -\alpha_k\left(1-\frac{LM_2}{2}\alpha_k\right)\mathbb{E}_k[\|\nabla f(x_k)\|_2^2] + \frac{LM_1}{2}\alpha_k^2.$$

Since $\alpha_k = \alpha$, we have

$$\mathbb{E}_k[f(x_{k+1})] - \mathbb{E}_k[f(x_k)] \leq -\alpha\left(1-\frac{LM_2}{2}\alpha\right)\mathbb{E}_k[\|\nabla f(x_k)\|_2^2] + \frac{LM_1}{2}\alpha^2.$$

Summing the above inequalities up from $k = 1, 2, \ldots, K$, we have

$$\begin{aligned}\mathbb{E}_k[f(x_{k+1})] - f(x_1) &\leq -\alpha\left(1-\frac{LM_2}{2}\alpha\right)\mathbb{E}_k\left[\sum_{k=1}^{K}\|\nabla f(x_k)\|_2^2\right] + \frac{LM_1K}{2}\alpha^2 \\ &\leq -\frac{\alpha}{2}\mathbb{E}_k\left[\sum_{k=1}^{K}\|\nabla f(x_k)\|_2^2\right] + \frac{LM_1K}{2}\alpha^2.\end{aligned}$$

Since $\mathbb{E}_k[f(x_{k+1})] - f(x_1) \geq f_* - f(x_1)$, we have

$$\mathbb{E}_k\left[\sum_{k=1}^{K}\|\nabla f(x_k)\|_2^2\right] \leq K\alpha LM_1 + \frac{2(f(x_1)-f_*)}{\alpha}.$$

Dividing K on both sides, we have

$$\mathbb{E}_k\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla f(x_k)\|_2^2\right] \leq \alpha LM_1 + \frac{2(f(x_1)-f_*)}{\alpha K} \xrightarrow{K\to\infty} \alpha LM_1,$$

as desired. □

Chapter 3

A Stochastic Trust Region Algorithm

3.1 Introduction

One disadvantage of the SG method is that stochastic gradients, like the gradients that they approximate, possess *no natural scaling*. By this, we mean that in order to guarantee convergence, the algorithm needs to choose stepsizes in a problem-dependent manner; e.g., common theoretical guarantees require that the stepsize is proportional to $1/L$, where L is a Lipschitz constant for the gradient of the objective function. This is in contrast to Newton's method for minimization, for which one can obtain (local) convergence guarantees with a stepsize of 1. Admittedly, Newton's method is not generally guaranteed to converge from remote starting points with unit stepsizes, but these observations do highlight a shortcoming of first-order methods, namely, that for convergence guarantees the stepsizes need always be chosen in a problem-dependent manner.

The purpose of this chapter is to propose a new algorithm for stochastic and finite-sum minimization. Our proposed approach can be viewed as a modification of the SG method. The approach does not completely overcome the issue of requiring problem-dependent stepsizes, but we contend that our approach does, for practical purposes, reduce somewhat this dependence. This is achieved by employing, under certain conditions, *normalized* steps. We

motivate our proposed approach by illustrating how it can be derived from a trust region methodology. This work can be viewed as a first step toward designing new classes of first- and second-order trust region methods for solving stochastic and finite-sum minimization problems.

The use of normalized steps has previously been proposed in the context of (stochastic) gradient methods for solving minimization problems. For example, in a method that is similar to ours, [26] propose an approach that employs normalized steps in every iteration. They show that, if the objective function is *M-bounded* and *strictly-locally-quasi-convex*, the stochastic gradients are sufficiently accurate with respect to the true gradients (specifically, when mini-batch sizes are $\Omega(\epsilon^{-2})$), and a sufficiently large number of iterations are run (specifically, $\Omega(\epsilon^{-2})$), then their method will, with high probability, yield a solution estimate that is ϵ-optimal. By contrast, our approach, by employing a modified update that does not always involve the use of a normalized step, enjoys convergence guarantees under different assumptions. We argue in this chapter that employing normalized steps in all iterations cannot lead to general convergence guarantees, which perhaps explains the additional assumptions required for convergence by [26].

It is also worthwhile to mention the broader literature. For important work on SG-type methods and their corresponding theoretical analyses, see, e.g., [1], [6], [11], [16], [21], [23], [24], [29], [38], [36], and [56]. There are also numerous variants of SG methods based on gradient aggregation, iterative averaging, second-order techniques, momentum, acceleration, and beyond; for work on these, see [4] and the references therein. More related to our work are techniques that normalize steplengths based on *accumulated* gradient information; see, e.g., [18] and [45]. In a different direction, one should also contrast our work with stochastic trust region approaches, such as those in [34] and [10]. The approaches proposed in these papers, which are based on the use of randomized models of the objective function constructed during each iteration, are quite distinct from our proposed method. For example, these approaches follow a traditional trust region strategy of accepting or rejecting each step based on the magnitude of an (approximate) *actual-to-predicted reduction ratio*. Our method, on the other hand, is closer to the SG method in that it accepts the computed step in every iteration. Another distinction is that these other approaches rely on the