SECOND-ORDER OPTIMIZATION METHODS FOR
MACHINE LEARNING

NAMAN AGARWAL

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE
ADVISER: PROFESSOR ELAD HAZAN

SEPTEMBER 2018
Abstract

In recent years first-order stochastic methods have emerged as the state-of-the-art in large-scale machine learning optimization. This is primarily due to their efficient per-iteration complexity. Second-order methods, while able to provide faster convergence, are less popular due to the high cost of computing the second-order information. The main problem considered in this thesis is can efficient second-order methods for optimization problems arising in machine learning be developed that improve upon the best known first-order methods.

We consider the ERM model of learning and propose linear time second-order algorithms for both convex as well as non-convex settings which improve upon the state-of-the-art first-order algorithms. In the non-convex setting second-order methods are also shown to converge to better quality solutions efficiently. For the convex case the proposed algorithms make use of a novel estimator for the inverse of a matrix and better sampling techniques for stochastic methods derived out of the notion of leverage scores. For the non-convex setting we propose an efficient implementation of the cubic regularization scheme proposed by Nesterov and Polyak.

Furthermore we develop second-order methods for achieving approximate local minima on Riemannian manifolds which match the convergence rate of their Euclidean counterparts. Finally we show the limitations of second/higher-order methods by deriving oracle complexity lower bounds for such methods on sufficiently smooth convex functions.
Contents

Abstract ........................................................................................................ iii
List of Tables ................................................................................................. viii
List of Figures ............................................................................................... ix

1 Introduction ................................................................................................. 1
  1.1 Background ........................................................................................... 2
  1.2 Our contributions ................................................................................ 5
  1.3 Published work .................................................................................. 9
  1.4 Notation and Definitions ..................................................................... 10

2 Second-Order Stochastic Optimization for Machine Learning in Linear Time 12
  2.1 Introduction ........................................................................................ 12
     2.1.1 Overview of Techniques ............................................................ 15
     2.1.2 Comparison with Related Work .............................................. 15
     2.1.3 Discussion and Subsequent Work .......................................... 18
  2.2 Preliminaries ....................................................................................... 20
  2.3 LiSSA: Linear (time) Stochastic Second-Order Algorithm ................. 23
     2.3.1 Estimators for the Hessian Inverse ......................................... 23
     2.3.2 Algorithm ............................................................................... 24
     2.3.3 Main Theorem ....................................................................... 25
     2.3.4 Proof of Main Theorem[2.3.3] ................................................. 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.5 Leveraging Sparsity</td>
<td>30</td>
</tr>
<tr>
<td>2.4 Experiments</td>
<td>32</td>
</tr>
<tr>
<td>2.5 LiSSA: Extensions</td>
<td>37</td>
</tr>
<tr>
<td>2.6 Conclusion</td>
<td>41</td>
</tr>
<tr>
<td>3 Leverage Score Sampling for Faster Accelerated Regression and GLM</td>
<td>43</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>43</td>
</tr>
<tr>
<td>3.1.1 Previous Results</td>
<td>45</td>
</tr>
<tr>
<td>3.1.2 Main Results</td>
<td>46</td>
</tr>
<tr>
<td>3.1.3 Our Approach</td>
<td>50</td>
</tr>
<tr>
<td>3.1.4 Chapter Organization</td>
<td>52</td>
</tr>
<tr>
<td>3.2 Regression Algorithm Given Leverage Score Estimates</td>
<td>52</td>
</tr>
<tr>
<td>3.3 Regression Algorithm Without Leverage Score Estimates</td>
<td>56</td>
</tr>
<tr>
<td>3.4 Extension for GLMs (under Assumption A3.1.7)</td>
<td>60</td>
</tr>
<tr>
<td>3.4.1 Proof of Theorem 3.4.2</td>
<td>63</td>
</tr>
<tr>
<td>3.5 Extensions to GLM (under Assumption 3.1.9)</td>
<td>72</td>
</tr>
<tr>
<td>3.6 Supplementary Results</td>
<td>73</td>
</tr>
<tr>
<td>3.6.1 Previous Best Regression Algorithm</td>
<td>73</td>
</tr>
<tr>
<td>3.6.2 Accelerated Coordinate Descent for GLM</td>
<td>77</td>
</tr>
<tr>
<td>3.6.3 Reduction from High Probability Solvers to Expected Running Times</td>
<td>79</td>
</tr>
<tr>
<td>3.6.4 Matrix Concentration for Sampling with Replacement</td>
<td>81</td>
</tr>
<tr>
<td>3.7 Conclusion</td>
<td>81</td>
</tr>
<tr>
<td>4 Finding Approximate Local Minima Faster Than Gradient Descent</td>
<td>82</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>82</td>
</tr>
<tr>
<td>4.1.1 Related work</td>
<td>85</td>
</tr>
<tr>
<td>4.1.2 Our Techniques</td>
<td>88</td>
</tr>
<tr>
<td>4.2 Preliminaries</td>
<td>89</td>
</tr>
</tbody>
</table>

v
List of Tables

2.1 Run time comparisons. Refer to Section 2.2 for definitions of the various notions of condition number. 17

2.2 A description of data sets used in the experiments. 34

3.1 History of improvements to the running time for regression in terms of the problem parameters \( \text{nnz}(A) \), \( n \), \( d \), \( \kappa \equiv \kappa(A^\top A) \), \( \kappa_{\text{sum}} \equiv \kappa_{\text{sum}}(A^\top A) \), and \( s \equiv s(A) \). 45

4.1 Comparison of known methods in existence before the release of the first manuscript with this work. 85

6.1 Table summarizing known results for convex optimization. 169

6.2 Lower bounds for higher order oracles. 172
# List of Figures

2.1 Performance of LiSSA as compared to a variety of related optimization methods for different data sets and choices of regularization parameter $\lambda$.

$S_1 = 1, S_2 \sim \kappa \ln(\kappa)$.

2.2 Convergence of LiSSA over time/iterations for logistic regression with MNIST, as compared to NewSamp and Newton’s method.

2.3 Performance of LiSSA as compared to a variety of related optimization methods for different data sets and choices of regularization parameter $\lambda$.

$S_1 = 1, S_2 \sim \kappa \ln(\kappa)$.

2.4 Differing convergence rates for LiSSA based on different choices of the $S_2$ parameter.

2.5 Comparison with accelerated first-order methods in ill-conditioned settings.

5.1 Performance profiles of ARC for $\theta = 1, 0.1, 0.01$, as compared to the Riemannian trust-region method (RTR) and the conjugate gradients method (CG) in terms of (a) time and (b) iterations.
Chapter 1

Introduction

In recent literature, stochastic first-order optimization has taken the stage as the primary workhorse for training machine learning (ML) models, due in large part to its affordable computational costs which are linear (in the underlying data representation) per iteration. Research efforts devoted to improving the convergence rates of first-order methods have introduced elegant ideas and algorithms over the years, including classical techniques such as momentum/acceleration [111] to adaptive regularization [57], variance reduction [84, 54], dual coordinate ascent [129], and many more. These ideas have been effective both in convex as well as non-convex settings.

In contrast, second-order methods have typically been much less explored in large scale machine learning (ML) applications due to their prohibitive computational cost per iteration which on the surface requires computation of the Hessian in addition to a matrix inversion. These operations are infeasible for large scale problems in high dimensions.

In this thesis, we study second-order methods in the context of machine learning applications. Our primary motive is to propose algorithms which are efficient with respect to the actual run-times of their implementation. Throughout the thesis we focus on developing algorithms that run in linear time with respect to data representation, i.e. algorithms with runtime guarantees that depend linearly on the number of training examples as well as the
underlying dimension. Further when applicable we will also leverage sparsity in the data by proposing algorithms that can be implemented in linear time with respect to the sparsity in the underlying data.

In particular, we propose efficient second-order algorithms both in the convex as well as non-convex settings under appropriate and practical smoothness assumptions which match and improve upon the best-known guarantees for first order methods. While second-order algorithms provide faster algorithms across the board for applications, they also, (as we demonstrate) are guaranteed to converge to better quality solutions in the case of non-convex landscapes. We extend the presented analysis for second-order methods to Riemannian manifolds which is a general class of non-Euclidean domains. Finally we study the limits of second-order optimization in the convex case in terms of its oracle complexity.

The rest of the this chapter provides an overview of the main results presented in this thesis starting with a quick background on the relevant concepts, followed by a brief exposition of the main contributions of the thesis. The chapters that follow, each provide an in-depth exposition of the main results including a description of the motivation and related existing work and the statement and proof of the main results. If applicable they also contain numerical experiments demonstrating the efficacy of the proposed algorithm.

1.1 Background

The central problem considered in this thesis is that of empirical risk minimization. Given a set of labelled examples \( \{(a_i, b_i): (a_i, b_i) \in A \times B\} \), a class of predictors \( \mathcal{X} = \{x: A \rightarrow B\} \) and a loss function \( l: B \times B \rightarrow \mathbb{R} \), the regularized empirical risk loss is defined as follows

\[
F(x) = \sum_{i=1}^{n} l(x(a_i), b_i) + R(x),
\]

where \( R(x) \) is a regularization function that is added to the loss to ensure generalization to unseen examples. The regularized empirical risk minimization (ERM) problem requires
minimizing the above loss, i.e. finding \( \argmin_{x \in \mathcal{X}} F(x) \). Standard PAC theory \([140, 87]\) tells us that solving ERM for a large enough sample size \((n)\), is sufficient for generalization over the unseen distribution from which the samples are generated. Therefore the ERM problem forms the basis for all the known supervised learning methods including Linear regression, SVMs, Logistic Regression, Deep Neural Networks etc.

While the above formulation is very general, in the first part of the thesis we focus on an important and interesting special case of the above problem which arises when one considers the case of \textit{linear predictors}. This case, which we refer to as Generalized Linear Models (GLMs) (define formally in Definition 2.1.1) makes the assumption that the underlying class of predictors \( \mathcal{X} \) is that of linear predictors, i.e. \( x \in \mathcal{X} \) is such that \( x(a) = a^T x \) for some vector \( x \in \mathbb{R}^d \). Further in such settings the loss function \( l(a^T x, b) \) is assumed to be jointly convex in both arguments. This case is general enough to capture various popular methods in supervised learning including linear/logistic regression, SVMs(including Kernel methods).

The main focus of this thesis is on second-order optimization methods (Newton’s method), where in each iteration, the underlying principle is to move to the minimizer of (a possibly regularized) second-order Taylor approximation. The standard update for Newton’s method for convex function at a point \( x_t \) is given by

\[
x_{t+1} = x_t - \left[ \nabla^2 F(x_t) \right]^{-1} \nabla F(x_t).
\]

(1.1.1)

Certain desirable properties of Newton’s method include the fact that its updates are independent of the choice of coordinate system and that the Hessian provides the necessary regularization based on the curvature at the present point. Indeed, Newton’s method can be shown to eventually achieve quadratic convergence for convex functions \([112]\). Although Newton’s method comes with good theoretical convergence guarantees, the complexity per step for GLM functions naively grows roughly as \( \Omega(nd^2 + d^w) \) (the former term for com-
puting the Hessian and the latter for inversion, where $\omega \approx 2.37$ is the matrix multiplication constant), making it prohibitive in practice.

In the second part of the thesis we consider the more general case of potentially non-convex losses under appropriate smoothness. This generalization is motivated by the success for Deep Neural Networks for machine learning which form our central application for this part of the thesis.

For non-convex loss functions, since global optimality is well known to be NP-hard \cite{81}, the notion of accuracy we consider is that of approximate second-order local minima, viz. points where the norm of the gradient is small and the hessian is guaranteed to be approximately PSD (see Chapter 4 for a precise definition). Since $\nabla^2 F(x)$ is no longer guaranteed to be PSD, it is no longer true that the Newton step i.e. $[\nabla^2 F(x)]^{-1} \nabla F(x)$ is a descent direction (even infinitesimally) or is even well-defined. The natural extension of second-order methods in this setting, proposed by \cite{115} minimizes at every step the following cubic-regularized second-order Taylor expansion,

$$m(y) = \nabla F(x)^\top (y - x) + \frac{(y - x)^\top \nabla^2 F(x)(y - x)}{2} + \frac{L_3}{6} \|y - x\|^3,$$ (1.1.2)

where $L_3$ is the Lipschitz constant for $\nabla^2 F(x)$. As shown by \cite{115} the above iterative procedure is guaranteed to converge to approximate local minima, however the procedure outlined for minimizing (1.1.2) in \cite{115} still requires matrix inversion and hence super linear in $d$ time.

In both of the above cases our focus is to obtain algorithms that run in time proportional to the underlying dimension. Since the Hessian matrix is a $d \times d$ matrix, our desideratum renders even writing down the Hessian matrix infeasible. To tackle this issue, the only access to the Hessian we assume is via a Hessian-vector product oracle, i.e. given a vector $v$ and a point $x$, the oracle outputs $\nabla^2 F(x)v$. For the case of GLMs we show that such an oracle can be implemented in linear time by observing that the Hessian in this case is
guaranteed to be of rank 1. Further in Chapter 4 we leverage an observation by [119] for functions representable via a differentiable circuit (see Section 4.2.1) that for such functions the Hessian-vector product oracle can be implemented in linear time as well. We provide a simpler, more intuitive argument for this observation in Section 4.2.1.

We now detail the main contributions of this thesis.

1.2 Our contributions

LiSSA (Linear Time Second-Order Stochastic Algorithm) (Chapter 2) Our first contribution, presented in Chapter 2 is a linear time algorithm for implementing the Newton step (1.1.1). The algorithm is based on a novel stochastic estimator for the inverse of the Hessian matrix which has desirable properties of being unbiased and having low variance in practice. The estimator is based on the Neumann series for the matrix inverse $A$ given by

$$A^{-1} = I + (I - A) + (I - A)^2 + \ldots$$

We show that access to a stochastic unbiased estimator of the Hessian matrix leads to an unbiased estimator of the inverse of the Hessian via the above series. Previous attempts to construct such an estimator had led to a biased estimator which still required the computation of an approximate matrix inverse [60]. The second key observation we make is that for GLMs multiplying the above estimator with a vector (the primitive required to execute a second-order step) can be preformed in input sparsity time.

Combining these pieces gives us a second-order algorithm that has running time comparable and in certain parameter regimes (of condition number) faster than the best known first-order methods for minimization of GLMs. Moreover we demonstrate via an empirical study that LiSSA on real world regression tasks performs favorably as compared to the state of the art methods like SVRG [84], SAGA [54] and their accelerated variants [12, 96].
Faster GLM optimization via Leverage Score Sampling (Chapter 3)  In modern applications of gradient-based methods, the classical technique of momentum/acceleration \cite{111} has been carefully adapted \cite{12, 96, 97} to achieve significant improvements when the problem is severely ill-conditioned. In particular, the runtime for accelerated methods to achieve relative accuracy $\varepsilon$ scales as

$$O \left( (n \cdot d + \sqrt{\kappa} \cdot n \cdot d) \log \left( \frac{1}{\varepsilon} \right) \right)$$ (1.2.1)

where $n, d$ are the number of training examples and the underlying dimension respectively and $\kappa$ (assumed to be $\geq n$) is the condition number of the problem (defined formally in Chapter 3).

In Chapter 3, we first focus on the sub-case of linear regression (here in the loss function $l(b_1, b_2) = 1/2 \cdot (b_1 - b_2)^2$) and propose an algorithm which is guaranteed to converge to a solution with relative accuracy $\varepsilon$ and can be implemented in total time

$$\tilde{O} \left( (n \cdot d + \sqrt{\kappa} \cdot d \cdot d) \log \left( \frac{1}{\varepsilon} \right) \right).$$ (1.2.2)

The above run-time is a significant improvement over (1.2.1) which is the best known run-time for linear regression, when the number of data examples is much larger than the dimension. This is often the case in practical settings (LibSVM \cite{61}, UCI \cite{95}) and can also be seen to hold in the stochastic case. In Theorem 3.1.5, we show a more nuanced bound which is data-dependent and can better adapt to regularity in the data.

Our algorithm comes about by merging known accelerated first-order methods with a carefully constructed sampling distribution based on leverage scores \cite{56, 136}.

Further in the chapter we show algorithms with the runtime (1.2.2) for GLMs under two regularity assumptions (A3.1.9 and A3.1.7). The first assumption (A3.1.9) covers the case of a Lipschitz Hessian but requires a warm start, as is typically the case with second-order algorithms. This case shows that the run-time improvement (1.2.2) can be achieved for
GLMs as well (upto log log factors) in the case when $\varepsilon \to 0$, since the *warm start* can be achieved via any first-order algorithm without dependence on $\varepsilon$. We remark that our results are complementary to that of [21] which shows that such an improvement requires access to the second derivative of the function.

In the second part of the chapter, we get rid of the requirement for the warm start under a stronger assumption (A3.1.7) of multiplicative third-order regularity. We propose an algorithm that generalizes the algorithm proposed for linear regression directly showing the runtime improvement (1.2.2) upto the regularity constant.

**FastCubic: Finding Approximate Local Minima Faster Than Gradient Descent** (Chapter 4) In Chapter 4 we consider the more general case where the loss function can be potentially non-convex. Assuming access to a linear time Hessian-vector oracle we propose an algorithm FastCubic, building upon the cubic regularization scheme proposed by Nesterov and Polyak [115]. The key result shown by us is that FastCubic converges to approximate local minima in time roughly proportional to

$$\frac{T_g}{\varepsilon^{3/2}} + \frac{T_h}{\varepsilon^{7/4}},$$

where $T_g$ refers to the time required to compute the gradient and $T_h$ refers to the time required to compute a Hessian-vector product. As observed by [119] (see Section 4.2.1) for models representable as a differentiable circuit $T_g \sim T_h$(upto constant factors). Putting the above together we get that the run-time to obtain approximate local minima is proportional to $\sim \frac{T_g}{\varepsilon^{3/2}}$ which improves the gradient descent guarantee of $\frac{T_g}{\varepsilon^{2}}$ to converge to approximate critical points(points with small gradient but no Hessian guarantee). We further provide faster running-times leveraging the sum of components structure of the underlying objective.

Our algorithm, to the best of our knowledge provides the first\footnote{Along with the parallel work of [41]} improvement over the classical guarantee given by gradient descent in terms of run-time. FastCubic is based on
approximate minimization of the regularized second-order Taylor expansion (1.1.2) via a binary search framework. Along the way we develop a characterization of the approximate solution for the induced sub-problem and leverage recent advances in fast computation of eigenvectors [16, 69].

Cubic Regularization over Manifolds  In Chapter 5 we extend the analysis of cubic regularization over the non-euclidean domain of Riemannian manifolds. In particular we analyze the adaptive cubic-regularization algorithm proposed by [45, 46] with the appropriate generalization to Riemannian manifolds. We establish that the same asymptotic convergence guarantee of $\epsilon^{-1.5}$ for achieving approximate first-order local minima as in the case of Euclidean domains can be obtained in the case of compact manifolds as well.

Our algorithm is based on minimizing a regularized local second-order Taylor approximation in the tangent space (which is a Euclidean space) of a point in the manifold. The non-euclidean curvature of the manifold implies that the progress made on the local second-order Taylor model need not ensure progress on the true function. Our analysis carefully controls this distortion based on the properties of the manifold as well as the retraction being used. See Chapter 5 for precise definitions of these quantities as well as the guarantee proved by us.

Lower Bounds for Higher-Order Optimization:  In the last part of the thesis we consider the oracle complexity of second/higher-order optimization on convex functions under the appropriate smoothness assumption. The key open question that motivated the work is whether higher-order optimization can be leveraged to produce a dimension free polynomial time convergence rate (algorithms that have logarithmic dependence in error) for smooth convex optimization. Such rates if possible could be of great practical significance as problems often have huge dimensions and are typically highly ill-conditioned.

In Chapter 6 we answer the question in the negative. In particular we demonstrate a lower bound for higher-order convex optimization that shows that a polynomial dependence
in the error parameter is necessary even when assuming access to higher-order derivatives and appropriate smoothness. Our lower bounds in particular show that existing algorithms proposed by [23, 104, 110] are in fact nearly tight. To the best of our knowledge these lower bounds are the first lower bounds for higher-order optimization assuming appropriate smoothness.

A striking aspect of our proof technique for the lower bound is its elegance and simplicity. The construction builds upon a well-known construction for first order optimization by carefully readjusting some of the parameters and then leverages integral convolution to ensure the existence higher-order derivatives and smoothness.

1.3 Published work

The matter presented in Chapter 2 is joint work with Brian Bullins and Elad Hazan and was published in the Journal of Machine Learning Research (JMLR) [8]. The matter presented in Chapter 3 is based on joint work with Sham Kakade, Rahul Kidambi, Praneth Netrapalli, Yin-Tat Lee and Aaron Sidford and is under submission. The manuscript appears as [10]. Chapter 4 is based on joint work with Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan and Tengyu Ma and was presented at the Symposium for Theory of Computing (STOC) 2017. The manuscript is available as [6]. Chapter 5 is based on joint work Nicolas Boumal, Brian Bullins and Coralia Cartis and is under submission. The manuscript is available as [7]. Chapter 6 is based on joint work with Elad Hazan and is to be presented at the Conference on Learning Theory (COLT), 2018. The manuscript is available as [9].

\(^2\)In independent and parallel work Shamir et al. [22] also demonstrate similar bounds
1.4 Notation and Definitions

Before proceeding on to the main portion of the thesis, we collect the notation used across the thesis. Other notation/definitions specific to chapters will be defined in the particular chapter itself.

We use \( \mathbb{R} \) to denote real numbers, \( \mathbb{N} \) to denote natural numbers and \([m]\) to denote the set \([1, 2, \ldots m]\). We adopt the convention of denoting vectors and scalars in lowercase and matrices in boldface uppercase.

We use \( O(\cdot) \) notation to hide absolute multiplicative constants. Concretely \( O(g) \) is a placeholder for some function \( f(x) \) such that there exists an absolute constant \( C \) such that \( \forall x \in \mathbb{R}, f(x) \leq Cg(x) \). \( \tilde{O}(g) \) captures the same inequality up to polylogarithmic factors in the underlying parameters. While using \( \tilde{O} \) we will be careful to state which parameters are hidden under the polylogarithmic factors. We analogously define \( \Omega(\cdot), \tilde{\Omega}(\cdot) \).

We will use \( \| \cdot \|_p \) to denote the \( \ell_p \) norm of a vector. For brevity we will reserve \( \| \cdot \| \) without a subscript for the \( \ell_2 \) norm for vectors and the spectral norm for matrices. \( \| \cdot \|_F \) denotes the Frobenius norm for matrices. Further, \( \| \cdot \|_0 \) will denote the number of non-zero entries of a vector/matrix. For a given positive definite matrix \( M \), we use \( \| \cdot \|_M \) to represent the semi-norm defined as \( \| x \|_M = \sqrt{x^\top M x} \).

For any given matrix \( A \in \mathbb{R}^{n \times m} \), let \( \sigma_{\max}(A) \), \( \sigma_{\min}(A) \) denote the largest and the smallest singular value respectively. Let \( \text{Tr}(A) \) represents the trace of the matrix \( A \). Further if \( A \) is real and symmetric, we use \( \lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A) \) to denote the eigenvalues of \( A \) and we let \( \lambda_{\max}(A) \triangleq \lambda_1(A) \) and \( \lambda_{\min}(A) \triangleq \lambda_n(A) \). For a given positive definite matrix \( A \), we denote by \( A^+ \) its Moor-Penrose pseudo-inverse.

For symmetric matrices \( A, B \in \mathbb{R}^{n \times n} \), we use \( A \preceq B \) to denote the condition that \( x^\top A x \leq x^\top B x \) for all \( x \in \mathbb{R}^n \) and we define \( <, \geq, \) and \( > \) analogously. We use \( \text{nnz}(A) \) to denote the number of non-zero entries in \( A \) and for a vector \( b \in \mathbb{R}^n \), we let \( \text{nnz}(b) \) denote the number of nonzero entries in \( b \).
For a twice differentiable real valued function $F: \mathbb{R}^d \to \mathbb{R}$, let $\nabla^2 F(x) \doteq [\nabla^2 F(x)]^{-1}$.

We call a function $F(x)$, $\beta$-smooth if for all $x$, $\nabla^2 F(x) \leq \beta \cdot I$ or equivalently for any $x, y$,

$$F(y) \leq F(x) + \nabla F(x)^\top (y - x) + \frac{\beta}{2} (y - x)^\top \nabla^2 F(x) (y - x). \quad (1.4.1)$$

Similarly we call a function $F(x)$, $\alpha$-strongly convex if for all $x$, $\nabla^2 F(x) \geq \alpha \cdot I$ or equivalently for any $x, y$,

$$F(y) \geq F(x) + \nabla F(x)^\top (y - x) + \frac{\alpha}{2} (y - x)^\top \nabla^2 F(x) (y - x). \quad (1.4.2)$$

11
Chapter 2

Second-Order Stochastic Optimization for Machine Learning in Linear Time

2.1 Introduction

In this chapter we propose a novel second-order algorithm, LiSSA (Linear time Stochastic Second-Order Algorithm) for convex optimization that attains fast convergence rates while also allowing for an implementation with linear time per-iteration cost, matching the running time of the best known gradient-based methods.

The main optimization problem we are concerned with is the unconstrained regularized empirical risk minimization problem over linear predictors. Such problems are also called Generalized Linear Models (GLM). Formally, we define GLM as

\[ F_p(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) = \frac{1}{n} \sum_{i=1}^{n} \psi_i(a_i^T x). \]

\[ \text{(GLM)} \]

Definition 2.1.1 (GLM). Given a matrix \( A \in \mathbb{R}^{n \times d} \) with rows \( a_1, \ldots, a_n \) and functions \( \{ \psi_1 \ldots \psi_n \} \in \mathbb{R} \rightarrow \mathbb{R} \) such that each \( \psi_i : \mathbb{R} \rightarrow \mathbb{R} \) is convex and twice differentiable, define \( F(x) : \mathbb{R}^d \rightarrow \mathbb{R} \) as

\[ F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) = \frac{1}{n} \sum_{i=1}^{n} \psi_i(a_i^T x). \]
Further given a convex function \( R(x) : \mathbb{R}^d \rightarrow \mathbb{R} \) (referred to as the regularizer), we wish to minimize

\[
\min_{x \in \mathbb{R}^d} F(x) + R(x)
\]

The above optimization problem is the standard objective minimized in most supervised learning settings. Examples include ridge and logistic regression, SVMs, etc. Further for the purpose of this chapter, we will make an additional assumption that the regularizer \( R(x) \) is the popular \( \ell_2 \) regularizer.

**Assumption 2.1.2.** \( R(x) = \lambda \|x\|^2 \).

While it is possible to generalize some of our algorithms in this chapter and the next chapter to other regularizers via standard techniques, we leave this extension as future work. Note that this assumption is general enough to capture many popular learning models including ridge and logistic regression.

Our main contribution in this chapter is an algorithm, which performs an approximate Newton update

\[
x_{t+1} = x_t + \nabla^{-2} F(x) \nabla F(x)
\]

(2.1.1)

based on stochastic Hessian information and is implementable in linear \( O(d) \) time. The algorithm is competent with the performance of first-order methods in theory and give promising results as an optimization method on real world data sets. In the following we give a summary of the results presented in this chapter. The main algorithm we propose is called LiSSA.

**LiSSA:** Presented as Algorithm LiSSA is a practical stochastic second-order algorithm based on a novel estimator of the Hessian inverse, leading to an efficient approximate Newton step (Equation 2.1.1). The estimator is based on the well known Taylor approximation of the inverse (Fact 2.2.1) and is described formally in Section 2.3.1. We prove the following informal theorem about LiSSA.
Theorem 2.1.3 (Informal). LiSSA returns a point $x_t$ such that $F(x_t) \leq \min_x F(x) + \varepsilon$ in total time

$$\tilde{O}\left((n + S_1\kappa) d \log\left(\frac{1}{\varepsilon}\right)\right)$$

where $\kappa$ is the underlying condition number of the problem and $S_1$ is a bound on the variance of the estimator.

The precise version of the above theorem appears as Theorem 2.3.3. In theory, the best bound we can show for $S_1$ is $O(\kappa^2)$; however, in our experiments we observe that setting $S_1$ to be a small constant (often 1) is sufficient. We conjecture that $S_1$ can effectively be improved to $O(1)$ and leave this for future work. If indeed $S_1$ can be improved to $O(1)$ (as is indicated by our experiments), LiSSA enjoys a convergence rate comparable to first-order methods. We provide a detailed comparison of our results with existing first-order and second-order methods in Section 2.1.2. Moreover, in Section 2.4 we present experiments on real world data sets that demonstrate that LiSSA as an optimization method performs well as compared to popular first-order methods. We also show that LiSSA runs in time proportional to input sparsity, making it an attractive method for high-dimensional sparse data.

In all of our results stated above $\kappa$ corresponds to the condition number of the underlying problem. This is ensured to be finite due to the addition of the $\ell_2$ regularizer $R(x)$. In stating our results formally we stress on the nuances between different notions of the condition number (ref. Section 2.2), and we state our results precisely with respect to these notions. In general, all of our generalization/relaxations of the condition number are smaller than $\frac{1}{\lambda}$ where $\lambda$ is the regularization parameter, and this is usually taken to be the condition number of the problem. The condition of strong convexity has been relaxed in literature by introducing proximal methods. It is an interesting direction to adapt our results in those setting which we leave for future work.
2.1.1 Overview of Techniques

The key idea underlying LiSSA is the use of the Taylor expansion to construct a natural estimator of the Hessian inverse. Indeed, as can be seen from the description of the estimator in Section 2.3.1, the estimator we construct becomes unbiased in the limit as we include additional terms in the series. We note that this is not the case with estimators that were considered in previous works such as that of [60], and so we therefore consider our estimator to be more natural. In the implementation of the algorithm we achieve the optimal bias/variance trade-off by truncating the series appropriately.

An important observation underlying our linear time $O(d)$ step is that for GLMs (Definition 2.1.1), for any $i$, $\nabla^2 f_i(x)$ has the form

$$\nabla^2 f_i(x) = \psi_i''(a_i^T x) \cdot a_i a_i^T.$$ 

A single step of LiSSA requires us to efficiently compute $\nabla^2 f_i(x)b$ for a given vector $b$. In this case it can be seen that the matrix-vector product reduces to a vector-vector product, giving us an $O(d)$ time update. Further if $a_i$s is $s$-sparse it allows for $O(s)$ time update.

2.1.2 Comparison with Related Work

In this section we aim to provide a short summary of the key ideas and results underlying optimization methods for large scale machine learning. We divide the summary into three high level principles: first-order gradient-based methods, second-order Hessian-based methods, and quasi-Newton methods. For the sake of brevity we will restrict our summary to results in the case when the objective is strongly convex, which as justified above is usually ensured by the addition of an appropriate regularizer. In such settings the main focus is often to obtain algorithms which have provably linear convergence and fast implementations.

First-Order Methods: First-order methods have dominated the space of optimization algorithms for machine learning owing largely to the fact that they can be implemented in
time proportional to the underlying dimension (or sparsity). Gradient descent is known to converge linearly to the optimum with a rate of convergence that is dependent upon the condition number of the objective. In the large data regime, stochastic first-order methods, introduced and analyzed first by [123], have proven especially successful. Stochastic gradient descent (SGD), however, converges sub-linearly even in the strongly convex setting. A significant advancement in terms of the running time of first-order methods was achieved recently by a clever merging of stochastic gradient descent with its full version to provide variance reduction. The representative algorithms in this space are SAGA [124, 54] and SVRG [84, 150]. The key technical achievement of the above algorithms is to relax the running time dependence on \( n \) (the number of training examples) and \( \kappa \) (the condition number) from a product to a sum. Another algorithm which achieves similar running time guarantees is based on dual coordinate ascent, known as SDCA [129].

Further improvements over SAGA, SVRG and SDCA have been obtained by applying the classical idea of acceleration emerging from the seminal work of [111]. The progression of work here includes an accelerated version of SDCA [130]; APCG [97]; SPDC [152]; Catalyst [64, 96], which provides a generic framework to accelerate first order algorithms; and Katyusha [12], which introduces the concept of negative momentum to extend acceleration for variance reduced algorithms beyond the strongly convex setting. The key technical achievement of accelerated methods in general is to reduce the dependence on condition number from linear to a square root. We summarize these results in Table 2.1.

LiSSA places itself naturally into the space of fast first-order methods by having a running time dependence that is comparable to SAGA/SVRG (ref. Table 2.1). In the next chapter we leverage the quadratic structure of the sub-problem and propose an algorithm which uses efficient sampling techniques based on leverage scores (defined in Definition 3.2.2). The results improve the running time in the case when the underlying dimension is much smaller than the number of training examples. We summarize these results in Table 2.1.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runtime</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVRG, SAGA, SDCA</td>
<td>((nd + O(\hat{\kappa}d)) \log \left( \frac{1}{\varepsilon} \right))</td>
<td>[84], [150], [124], [129]</td>
</tr>
<tr>
<td>LiSSA</td>
<td>((nd + O(\hat{\kappa}t)S_1d) \log \left( \frac{1}{\varepsilon} \right))</td>
<td>Corollary 2.3.4</td>
</tr>
<tr>
<td>AccSDCA, Catalyst</td>
<td>(\tilde{O} \left( nd + d\sqrt{\hat{\kappa}n} \right) \log \left( \frac{1}{\varepsilon} \right))</td>
<td>[130], [96], [12]</td>
</tr>
<tr>
<td>Leverage Scores</td>
<td>(\tilde{O} \left( nd + d\sqrt{\hat{\kappa}d} \right) \log \left( \frac{1}{\varepsilon} \right))</td>
<td>Theorems 3.1.8 and 3.1.10</td>
</tr>
</tbody>
</table>

Table 2.1: Run time comparisons. Refer to Section 2.2 for definitions of the various notions of condition number.

**Second-Order Methods:** Second-order methods such as Newton’s method have classically been used in optimization in many different settings including development of interior point methods [108] for general convex programming. The key advantage of Newton’s method is that it achieves a quadratic convergence rate eventually. However, naive implementations of Newton’s method have two significant issues, namely that the standard analysis requires the full Hessian calculation which costs \(O(nd^2)\), an expense not suitable for machine learning applications, and the matrix inversion typically requires \(O(d^3)\) time. These issues were addressed recently by the algorithm NewSamp [60] which tackles the first issue by subsampling and the second issue by low-rank projections. We improve upon the work of [60] by defining a more natural estimator for the Hessian inverse and by demonstrating that the estimator can be computed in time proportional to \(O(d)\). We also point the reader to the works of [100, 38] which incorporate the idea of taking samples of the Hessian; however, these works do not provide precise running time guarantees on their proposed algorithm based on problem specific parameters. Second-order methods have also enjoyed success in the distributed setting [131, 151].

**Quasi-Newton Methods:** The expensive computation of the Newton step has also been tackled via estimation of the curvature from the change in gradients. These algorithms are generally known as quasi-Newton methods stemming from the seminal BFGS algorithm.
The book of [117] is an excellent reference for the algorithm and its limited memory variant (L-BFGS). The more recent work in this area has focused on stochastic quasi-Newton methods which were proposed and analyzed in various settings by [127, 103, 39]. These works typically achieve sub-linear convergence to the optimum. A significant advancement in this line of work was provided by [105] who propose an algorithm based on L-BFGS by incorporating ideas from variance reduction to achieve linear convergence to the optimum in the strongly convex setting. Although the algorithm achieves linear convergence, the running time of the algorithm depends poorly on the condition number (as acknowledged by the authors). Indeed, in applications that interest us, the condition number is not necessarily a constant as is typically assumed to be the case for the theoretical results in [105].

Our key observation of linear time Hessian-vector product computations for machine learning applications provides evidence that in such instances, obtaining true Hessian information is efficient enough to alleviate the need for quasi-Newton information via gradients.

2.1.3 Discussion and Subsequent Work

In this section we provide a brief survey of certain technical aspects of our bounds which have since been improved by subsequent work.

An immediate improvement in terms of $S_2 \sim \kappa$ (suggested in the original manuscript for our work) was achieved by [32] via conjugate gradient on a sub-sampled Hessian which reduces this to $\sqrt{\kappa}$. A similar improvement can also be achieved in theory through the extensions of LiSSA proposed in the paper. As we show in Section 2.4, the worse dependence on condition number has an effect on the running time when $\kappa$ is quite large\footnote{Equivalently, $\lambda$ is small.}. Accelerated first-order methods, such as APCG [97], outperform LiSSA in this regime. To the best of our knowledge second-order stochastic methods have so far not exhibited
an improvement in that regime experimentally. We believe a more practical version of LiSSA-Sample could lead to improvements in this regime, leaving this as future work.

To the best of our knowledge the factor of $S_1 = \kappa^2$ that appears to reduce the variance of our estimator has yet not been improved despite it being $O(1)$ in our experiments. This is an interesting question to which partial answers have been provided in the analysis of [147].

Significant progress has been made in the space of inexact Newton methods based on matrix sketching techniques. We refer the reader to the works of [120, 145, 51, 99, 147] and the references therein.

We would also like to comment on the presence of a warm start parameter $\frac{1}{\kappa L_3}$ in our proofs of Theorems 2.3.3 and 2.5.2. In our experiments the warm start we required would be quite small (often a few steps of gradient descent would be sufficient) to make LiSSA converge. The warm start does not affect the asymptotic results proven in Theorems 2.3.3 and 2.5.2 because getting to such a warm start is independent of $\epsilon$. However, improving this warm start, especially in the context of Theorem 2.5.2, is left as interesting future work.

On the complexity side, [21] proved lower bounds on the best running times achievable by second-order methods. In particular, they show that to get the faster rates achieved by LiSSA-Sample, it is necessary to use a non-uniform sampling based method as employed by LiSSA-Sample. We would like to remark that in theory, up to logarithmic factors, the running time of LiSSA-Sample is still the best achieved so far in the setting $n \gg d$. Some of the techniques and motivations from this work were also generalized by the authors to provide faster rates for a large family of non-convex optimization problems [6].

**Organization of the Chapter** The chapter is organized as follows: we first present some known theorems which we use in the rest of the chapter in Section 2.2. We then describe our estimator for LiSSA, as well as state and prove the convergence guarantee for LiSSA in Section 2.3. Further, we present an experimental evaluation of LiSSA in Section 2.4. Finally, we explicitly state a general reduction from convex optimization to solving intermediate
quadratics. This reduction allows us to use the results from next chapter to achieve the best known running times. This reduction was implicitly used in the analysis of LiSSA as well.

### 2.2 Preliminaries

Throughout the chapter we denote $x^* \triangleq \arg\min_{x \in \mathbb{R}^d} F(x)$. We now collect key concepts and pre-existing results that we use for our analysis in the rest of the chapter.

**Neumann Series**  The following is a well known fact about the inverse of a matrix $A$ s.t. $\|A\| \leq 1$ and $A > 0$:

$$A^{-1} = \sum_{i=0}^{\infty} (I - A)^i.$$  \hspace{1cm} (2.2.1)

**Matrix Concentration**  : The following theorem is a standard concentration of measure result for sums of independent matrices. An excellent reference for this material is by [138].

**Theorem 2.2.1** (Matrix Bernstein, [138]). Consider a finite sequence $\{X_k\}$ of independent, random, Hermitian matrices with dimension $d$. Assume that $E[X_k] = 0$ and $\|X_k\| \leq R$.

Define $Y = \sum_k X_k$. Then we have for all $t \geq 0$,

$$\Pr (\|Y\| \geq t) \leq d \exp \left( -\frac{t^2}{4R^2} \right).$$

**Sherman-Morrison Formula**  : The following is a well-known expression for writing the inverse of rank one perturbations of matrices:

---

\(^2\)The theorem in the reference states the inequality for the more nuanced bounded variance case. We only state the simpler bounded spectral norm case which suffices for our purposes.
\[(A + vu^T)^{-1} = A^{-1} - \frac{A^{-1}vu^TA^{-1}}{1 + u^TA^{-1}v}.\]

**Definition of Condition Numbers**  We now define several measures for the condition number of a function \(F\). The differences between these notions are subtle and we use them to precisely characterize the running time for our algorithms.\(^3\)

For an \(\alpha\)-strongly convex and \(\beta\)-smooth function \(F\), the condition number of the function is defined as \(\kappa(F) \equiv \frac{\beta}{\alpha}\), or \(\kappa\) when the function is clear from the context. Note that by definition this corresponds to the following notion:

\[
\kappa = \frac{\max_x \lambda_{\max}(\nabla^2 F(x))}{\min_x \lambda_{\min}(\nabla^2 F(x))}.
\]

We define a slightly relaxed notion of condition number where the \(\max\) moves out of the fraction above. We refer to this notion as a *local* condition number \(\kappa_l\) as compared to the *global* condition number \(\kappa\) defined above:

\[
\kappa_l = \max_x \frac{\lambda_{\max}(\nabla^2 F(x))}{\lambda_{\min}(\nabla^2 F(x))}.
\]

It follows that \(\kappa_l \leq \kappa\). The above notions are defined for any general function \(F\), but in the case of functions of the form \(F(x) = \frac{1}{n} \sum_{k=1}^n f_k(x)\), we make a stronger restriction with respect to the component functions. We refer to such definitions of the condition number by \(\hat{\kappa}\). We assume that each component \(\lambda_{\max}(\nabla^2 f_k(x))\) is bounded individually for all \(x\). We can now define the following notion of condition number:

\[
\hat{\kappa} = \frac{\max_x \max_k \lambda_{\max}(\nabla^2 f_k(x))}{\min_x \lambda_{\min}(\nabla^2 F(x))}.
\]

\(^3\)During initial reading we suggest the reader to skip the subtlety with these notions with the knowledge that they are all smaller than the pessimistic bound one can achieve by considering a value proportional to \(O(\lambda^{-1})\), where \(\lambda\) is the coefficient of the \(\ell_2\) regularizer.
Similarly, we define a notion of local condition number for $\hat{\kappa}$, namely

$$
\hat{\kappa}_l \triangleq \max_x \frac{\max_k \lambda_{\max}(\nabla^2 f_k(x))}{\lambda_{\min}(\nabla^2 F(x))}.
$$

(2.2.2)

It again follows that $\hat{\kappa}_l \leq \hat{\kappa}$. For the purpose of this chapter we make a stronger assumption of a component wise upper bound on the smoothness. Algorithms like SVRG [144] have been extended to the case when the smoothness bound is averaged over all the components.

We consider this nuance carefully in the next chapter where our runtime will depend on the average quantity.

For our (admittedly pessimistic) bounds on the variance of the estimator we also need a per-component strong convexity bound, i.e. we require that $\min_k \lambda_{\min}(\nabla^2 f_k(x))$ is lower bounded away from 0 for all $x$. We now define the following quantity

$$
\hat{\kappa}^{\text{max}}_l \triangleq \max_x \frac{\max_k \lambda_{\max}(\nabla^2 f_k(x))}{\min_k \lambda_{\min}(\nabla^2 f_k(x))}.
$$

(2.2.3)

**Assumptions** : In light of the previous definitions, we make the following assumptions/redefinitions regarding the given GLM function $F(x) = \frac{1}{n} \sum_{k=1}^{n} f_k(x) + \lambda \|x\|^2$ to make the analysis easier. Firstly we redefine $f_i$ to include the regularization term as

$$
f_k(x) \triangleq f_k(x) + \lambda \|x\|^2.
$$

Under the above redefinition we have that $F(x) = \frac{1}{n} \sum_{i=1}^{n} f_k(x)$.

We further assume that the each component function $f_k$ is smooth, i.e. $\|\nabla^2 f_k(x)\|$ is bounded for all $x$. To ensure brevity in our notation without loss of generality we assume that the function $F(x)$ has been scaled such that for all $k$ and all $x \in \mathbb{R}^d$,

**Assumption 2.2.2.** $\nabla^2 f_k(x) \preceq I$.

Note that the above assumption can be achieved without loss of generality via scaling. We remark that even while looking at additive notions of error such a scaling introduces at
most a logarithmic factor into the run-time since the run-time depends at most logarithmically on the error.

Further we assume a third-order regularity condition on the function $F(x)$.

**Assumption 2.2.3.** $\nabla^2 F(x)$ is $L_3$-Lipschitz with respect to the $\| \cdot \|$ norm, i.e.

$$\forall x, y \quad \| \nabla^2 F(x) - \nabla^2 F(y) \| \leq L_3 \| x - y \|$$

Finally we denote by $\hat{\kappa}_l$ and $\hat{\kappa}_l^{\text{max}}$, the associated local condition numbers for $F(x)$ defined in (2.2.2) and (2.2.3) respectively.

### 2.3 LiSSA: Linear (time) Stochastic Second-Order Algorithm

In this section, we provide an overview of LiSSA (Algorithm 1) along with its main convergence results.

#### 2.3.1 Estimators for the Hessian Inverse

Based on a recursive reformulation of the Neumann series (Equation 2.2.1), we now describe an unbiased estimator of the Hessian. For any matrix $A$, define $A_j^{-1}$ as the first $j$ terms in the Neumann series, i.e.,

$$A_j^{-1} = \sum_{i=0}^{j} (I - A)^i, \quad \text{or equivalently} \quad A_j^{-1} = I + (I - A) A_{j-1}^{-1} \quad \text{with} \quad A_0^{-1} = I.$$

Note that $\lim_{j \to \infty} A_j^{-1} \to A^{-1}$. Using the above recursive formulation, we now describe an unbiased estimator of $\nabla^{-2} F$ by first deriving an unbiased estimator $\tilde{\nabla}^{-2} F_j$ for $\nabla^{-2} F_j$ for any $j$.  

23
Definition 2.3.1 (Estimator). Given \(j\) independent random variables \(\{X_1 \ldots X_j\}\) such that for all \(t\),

\[
\mathbb{E}[X_t] = \nabla^2 F,
\]

define \(\{\nabla^{-2} F_0, \ldots, \nabla^{-2} F_j\}\) recursively as follows:

\[
\nabla^{-2} F_0 = I \quad \text{and} \quad \nabla^{-2} F_t = I + (I - X_t)\nabla^{-2} F_{t-1} \quad \text{for} \quad t = 1, \ldots, j.
\]

Using the fact that \(\{X_0 \ldots X_j\}\) are independent and have expectation \(\nabla^2 F\) we get that

\[
\mathbb{E}[\nabla^{-2} F_j] = \nabla^{-2} F_j.
\]

Therefore we also have that

\[
\mathbb{E}[\nabla^{-2} F_j] \rightarrow \nabla^{-2} F \quad \text{as} \quad j \rightarrow \infty,
\]

providing us with an unbiased estimator in the limit.

Remark 2.3.2. One can also define and analyze a simpler (non-recursive) estimator based on directly sampling terms from the series in Equation (2.2.1). Theoretically, one can get similar guarantees for the estimator; however, empirically our proposed estimator exhibited better performance.

2.3.2 Algorithm

The algorithm starts from a warm start (2.3.1) and takes approximate Newton steps by constructing the estimator from Definition 2.3.1 in place of the true Hessian inverse. We use two parameters, \(S_1\) and \(S_2\), to define the Newton step. \(S_1\) represents the number of unbiased estimators of the Hessian inverse we average to get better concentration for our estimator, while \(S_2\) represents the depth to which we capture the Neumann series (i.e. \(j\) in Definition
Algorithm 1 LiSSA: Linear (time) Stochastic Second-Order Algorithm

1: **Input:** \( T, F(x) = \frac{1}{n} \sum_{k=1}^{n} f_k(x), S_1, S_2, x_1 \)

2: **for** \( t = 1 \) to \( T \) **do**

3: **for** \( i = 1 \) to \( S_1 \) **do**

4: \( y[i,0] = \nabla F(x_t) \)

5: **for** \( j = 1 \) to \( S_2 \) **do**

6: Sample an index \( s[i, j] \) uniformly at random from \( \{1 \ldots m\} \)

7: Define \( \tilde{\nabla^2} f_{s[i,j]}(x_t) \)

8: Set \( y[i,j] = \nabla F(x_t) + (I - \tilde{\nabla^2} f_{s[i,j]})y[i,j-1] \)

9: **end for**

10: **end for**

11: \( y_t = \frac{1}{S_1} \cdot \left( \sum_{i=1}^{S_1} y[i,s_2] \right) \)

12: \( x_{t+1} = x_t - y_t \)

13: **end for**

14: **return** \( x_{T+1} \)

2.3.1. In the algorithm, instead of constructing an estimate of the Hessian inverse, we describe the computation of the approximate Newton step directly. As observed earlier this computation can be performed in \( O(d) \) time owing to the GLM structure of \( F(x) \).

2.3.3 Main Theorem

In this section we present our main theorem which analyzes the convergence properties of LiSSA.

**Theorem 2.3.3.** Suppose we are given a GLM function \( F(x) \) satisfying assumptions **A2.1.2**

**A2.2.2 A2.2.3** and with local condition numbers \( \hat{\kappa}_l \) and \( \hat{\kappa}_l^{\max} \) respectively. Let \( x^* = \arg\min_{x \in \mathbb{R}^d} F(x) \). Further suppose we are given \( x_1 \) such that

\[
\|x_1 - x^*\| \leq \frac{1}{4\hat{\kappa}_l L_3}.
\] (2.3.1)

Then for Algorithm 7 with parameters set as follows:

\[
S_1 = O((\hat{\kappa}_l^{\max})^2 \ln(d\delta^{-1})) \quad \text{and} \quad S_2 \geq 2\hat{\kappa}_l \ln(4\hat{\kappa}_l),
\]

25
the following guarantee holds for every \( t \geq 1 \) with probability \( 1 - \delta \),

\[
\|x_{t+1} - x^*\| \leq \frac{\|x_t - x^*\|}{2}.
\]

Moreover, we have that steps 3-12 of the algorithm can be implemented in time at most

\[
O(nd + S_1 \cdot S_2 \cdot d).
\]

As an immediate corollary of the above theorem, we obtain the following:

**Corollary 2.3.4.** For a GLM function \( F(x) \), there exists an algorithm which returns a point \( x \) such that with probability at least \( 1 - \delta \),

\[
F(x) \leq \min_x F(x) + \varepsilon
\]

in total time \( \tilde{O}(n + (\tilde{\kappa}^\text{max})^2 \tilde{\kappa}^1) d \ln \left( \frac{n}{\varepsilon} \right) \) for \( \varepsilon \to 0 \).

In the above theorems \( \tilde{O} \) hides log factors of \( \kappa, d, \frac{1}{\delta} \). Corollary 2.3.4 follows from Theorem 2.3.3 by noting that the point \( x_1 \) can be obtained via running any first order method (e.g. SVRG) for sufficient accuracy and then switching over to LiSSA. Note that the accuracy required for the first order method is not dependent on \( \varepsilon \) and hence the running time follows.

We note that our bound \( (\tilde{\kappa}_i^\text{max})^2 \) on the variance is possibly pessimistic and can likely be improved to a quantity that depends on an average of the underlying parameters. However, the experiments conducted by us (Section 2.4) reveal that in practice setting the parameter \( S_1 \sim O(1) \) suffices and we have not tried to optimize this dependence further.

### 2.3.4 Proof of Main Theorem 2.3.3

We now prove our main theorem about the convergence of LiSSA (Theorem 2.3.3).
Proof of Theorem 2.3.3. As can be seen from Definition 2.3.1, a single step of our algorithm is equivalent to \( x_{t+1} = x_t - \nabla^{-2}F(x_t)\nabla F(x_t) \), where \( \nabla^{-2}F(x_t) \) is the average of \( S_1 \) independent estimators \( \tilde{\nabla}^{-2}F(x_t)_{S_2} \). We now make use of the following lemma.

Lemma 2.3.5. Let \( x_{t+1} = x_t - \nabla^{-2}F(x_t)\nabla F(x_t) \), as per a single iteration of Algorithm 1, and suppose \( S_1, S_2 \) are as defined in Algorithm 1. Then if we choose \( S_2 \geq 2\tilde{\kappa}_l \ln(2\tilde{\kappa}_l) \) we have the following guarantee on the convergence rate for every step with probability \( 1 - \delta \):

\[
\|x_{t+1} - x^*\| \leq \gamma \|x_t - x^*\| + L_3 \|\nabla^{-2}F(x_t)\| \|x_t - x^*\|^2
\]

where \( \gamma = 16\tilde{\kappa}_l \max \sqrt{\frac{\ln(d^{d-1})}{S_1}} + \frac{1}{16} \).

Substituting the values of \( S_1 \) and \( S_2 \) in Lemma 2.3.5 noting that \( \|\nabla^{-2}F(x_t)\| \leq \tilde{\kappa}_l \) which follows from the definition in (2.2.2) and A2.2.2, we get that for all \( t \), the following is true

\[
\|x_{t+1} - x^*\| \leq \frac{\|x_t - x^*\|}{4} + L_3 \tilde{\kappa}_l \|x_t - x^*\|^2 \quad (2.3.2)
\]

We now argue by induction. We will show that for all \( t \)

\[
\|x_{t+1} - x^*\| \leq \frac{\|x_t - x^*\|}{2} \quad \text{as well as} \quad \|x_t - x^*\| \leq \frac{1}{4L_3\tilde{\kappa}_l} \cdot \quad (2.3.3)
\]

Lets take the case of \( t = 1 \). (2.3.1) ensures the second part of (2.3.3). Plugging the second part of (2.3.3) into (2.3.2) we get the first part of (2.3.3). For the inductive case assume (2.3.3) holds for some \( t \). Therefore we have that

\[
\|x_{t+1} - x^*\| \leq \frac{1}{4L_3\tilde{\kappa}_l}.
\]

Substituting the above in (2.3.2), we get that

\[
\|x_{t+2} - x^*\| \leq \frac{\|x_{t+1} - x^*\|}{2}.
\]

27
This finishes the proof of the first part. To observe the run-time guarantee note that by definition of GLM functions  

\[ \tilde{\nabla}^2 f_{i,j} = \psi_{s[i,j]}(a_{s[i,j]} \cdot x) a_{s[i,j]} a_{s[i,j]}^T \]

and therefore Step 8 can be implemented in time \( O(d) \). Further since \( \nabla F(x) \) can be computed in time \( O(nd) \), the run-time guarantee follows. \( \square \)

We now provide a proof of Lemma \[2.3.5\].

**Proof of Lemma \[2.3.5\]** Define \( \chi(x_t) = \int_0^1 \nabla^2 F(x^* + \tau(x_t - x^*)) d\tau \). Note that \( \nabla F(x_t) = \chi(x_t)(x_t - x^*) \). Following an analysis similar to that in [112], we have that

\[
\|x_{t+1} - x^*\| = \|x_t - x^* - \tilde{\nabla}^{-2} F(x_t) \nabla F(x_t)\| \\
= \|x_t - x^* - \tilde{\nabla}^{-2} F(x_t) \chi(x_t)(x_t - x^*)\| \\
\leq \|I - \tilde{\nabla}^{-2} F(x_t) \chi(x_t)\| \|x_t - x^*\|.
\]

Following from the previous equations, we have that

\[
\frac{\|x_{t+1} - x^*\|}{\|x_t - x^*\|} \leq \|I - \tilde{\nabla}^{-2} F(x_t) \chi(x_t)\| = \|I - \nabla^{-2} F(x_t) \chi(x_t)\| - \left( \tilde{\nabla}^{-2} F(x_t) - \nabla^{-2} F(x_t) \right) \chi(x_t). \tag{2.3.4}
\]

We now analyze the above two terms \( a \) and \( b \) separately:

\[
\|a\| = \|I - \nabla^{-2} F(x_t) \chi(x_t)\| \\
\leq \|\nabla^{-2} F(x_t) \int_0^1 \left( \nabla^2 F(x_t) - \nabla^2 F(x^* + \tau(x_t - x^*)) \right) d\tau \| \\
\leq L_3 \|\nabla^{-2} F(x_t)\| \|x_t - x^*\|.
\]

The second inequality follows from \[A2.2.3\]. To bound the second term we make use of the following lemma.
Lemma 2.3.6. Let \( \tilde{\nabla}^{-2} F(x_t) \) be the average of \( S_1 \) independent samples of \( \tilde{\nabla}^{-2} F(x_t)_{S_2} \), as defined in Definition 2.3.1 and as used in Step \( 11 \) of Algorithm 7. If we set \( S_2 \geq 2\tilde{\kappa}_l \ln(\tilde{\kappa}_l S_1) \), then we have that

\[
\Pr \left( \| \tilde{\nabla}^{-2} F(x_t) - \nabla^{-2} F(x_t) \| > 16\tilde{\kappa}_l \ln \left( \frac{\delta}{\tilde{\kappa}_l S_1} \right) + 1/16 \right) \leq \delta.
\]

Further note that using A2.2.2 we have that for all \( x \), \( \| \nabla^2 F(x) \| \leq 1 \), and hence \( \| \chi(x) \| \leq 1 \). Therefore second term can be bounded as follows:

\[
\| b \| \leq \left( \| \left( \tilde{\nabla}^{-2} F(x_t) - \nabla^{-2} F(x_t) \right) \| \| \chi(x_t) \| \right) \leq \gamma.
\]

Putting the above bounds on \( \| a \| \) and \( \| b \| \) together and using the triangle inequality in (2.3.4), we have that

\[
\frac{\| x_{t+1} - x^* \|}{\| x_t - x^* \|} \leq L_3 \| \nabla^{-2} F(x_t) \| \| x_t - x^* \| + \gamma
\]

which concludes the proof.

Proof of Lemma 2.3.6 First note the following statement which is a straightforward implication of our construction of the estimator:

\[
\mathbb{E} [ \tilde{\nabla}^{-2} F(x_t) ] = \mathbb{E} [ \tilde{\nabla}^{-2} F(x_t)_{S_2} ] = \sum_{i=0}^{S_2} (I - \nabla^2 F(x_t))^i.
\]

We also know from Equation (2.2.1) that for matrices \( X \) such that \( \| X \| \leq 1 \) and \( X > 0 \),

\[
X^{-1} = \sum_{i=0}^{\infty} (I - X)^i.
\]

Using A2.2.2 it follows that

\[
\nabla^{-2} F(x_t) = \mathbb{E} [ \tilde{\nabla}^{-2} F(x_t)_{S_2} ] + \sum_{i=S_2+1}^{\infty} (I - \nabla^2 F(x_t))^i.
\]
Also it follows from the definition in (2.2.2) and A2.2.2 that $\nabla^2 F(x_t) \geq \frac{1}{\hat{\kappa}_l}$. Therefore we have that $\|I - \nabla^2 F(x_t)\| \leq 1 - \frac{1}{\hat{\kappa}_l}$. Observing the second term in the above equation,

$$\left\| \sum_{i=S_2}^{\infty} (I - \nabla^2 F(x_t))^i \right\| \leq \left(1 - \frac{1}{\hat{\kappa}_l}\right)^{S_2} \left(\sum_{i=0}^{\infty} (1 - \frac{1}{\hat{\kappa}_l})^i\right) \leq (1 - \frac{1}{\hat{\kappa}_l})^{S_2} \hat{\kappa}_l \leq \exp\left(-\frac{S_2}{\hat{\kappa}_l}\right) \hat{\kappa}_l.$$

Since we have chosen $S_2 \geq 2\hat{\kappa}_l \ln(4\hat{\kappa}_l)$, we get that the above term is bounded by $\frac{1}{16}$. We will now show, using the matrix Bernstein inequality (Theorem 2.2.1), that the estimate $\nabla^{-2} F$ is concentrated around its expectation. To apply the inequality we first need to bound the spectral norm of each random variable. To that end we note that $\nabla^{-2} F_{S_2}$ has maximum spectral norm bounded by

$$\|\nabla^{-2} F_{S_2}\| \leq \sum_{i=0}^{S_2} (1 - 1/\hat{\kappa}_l^{\text{max}})^i \leq \hat{\kappa}_l^{\text{max}}.$$

We can now apply Theorem 2.2.1, which gives the following:

$$\Pr\left(\|\nabla^{-2} F - \mathbb{E}[\nabla^{-2} F]\| > \varepsilon\right) \leq 2d \exp\left(\frac{-\varepsilon^2 S_1}{64(\hat{\kappa}_l^{\text{max}})^2}\right).$$

Setting $\varepsilon = 16\hat{\kappa}_l^{\text{max}} \sqrt{\frac{\ln(\frac{d}{\delta})}{S_1}}$ gives us that the probability above is bounded by $\delta$. Now putting together the above bounds and Equation (2.3.5) we get the required result. \hfill \square

### 2.3.5 Leveraging Sparsity

A key property of real-world data sets is that although the input is a high dimensional vector, the number of non-zero entries is typically very low. The following theorem shows that
LiSSA can be implemented in a way to leverage the underlying sparsity of the data. Our key observation is that for GLM functions, the rank one Hessian-vector product can be performed in $O(s)$ time where $s$ is the sparsity of the input $x_k$.

**Theorem 2.3.7.** Given a GLM function $F(x)$, let $s(A)$ be a bound on the number of non-zeros of any $a_i$, i.e.

$$s(A) = \max_k \|a_k\|_0$$

Then Algorithm 3 returns a point $x_T$ such that with probability at least $1 - \delta$

$$F(x_T) \leq \min_x F(x) + \varepsilon$$

in total time $O((m + (\hat{\kappa}_l^{\max})^2 \cdot s(A)) \ln \left(\frac{1}{\varepsilon}\right) \text{ for } \varepsilon \to 0.$

We will prove the following lemma, from which Theorem 2.3.7 immediately follows using Theorem 2.3.3.

**Lemma 2.3.8.** Given a GLM function $F(x)$, let $s(A)$ be a bound on the number of non-zeros of any $a_i$, i.e.

$$s(A) = \max_k \|a_k\|_0.$$

Then steps 3-12 of Algorithm 3 can be implemented in time $O((m + S_1 \cdot S_2) \cdot s)$.

**Proof of Lemma 2.3.8.** Consider Algorithm 3 and steps 3-12. We will provide an implementation of the steps which leverages sparsity. To this end note that by definition

$$\nabla^2 f_{i,j}(x) = \psi''_{s(i,j)} (a_{s(i,j)}^T x) a_{s(i,j)} a_{s(i,j)}^T + \lambda I$$

for some index $s(i,j)$. Using this we now define the following quantities.

Let $c_0 = 1, d_0 = 1, v_0 = \vec{0}$. Further define the following series recursively,

$$c_{j+1} = 1 + (1 - \lambda)c_j \quad d_{j+1} = (1 - \lambda)d_j$$

$$v_{j+1} = v_j - \frac{\psi''_{s(i,j)} (a_{s(i,j)}^T x)}{(1 - \lambda)d_j} a_{s(i,j)} a_{s(i,j)}^T (c_j \nabla F(x) + d_j v_j),$$

$$31$$
where \( \lambda \) is the regularization parameter.

We will first show that \( y_{i,j} = c_j \nabla F(x) + d_j v_j \) via induction. For the base case, note that \( y_{i,0} = c_0 \nabla F(x) + d_0 v_0 = \nabla F(x) \), as is the case in Algorithm 1. Furthermore, suppose \( y_{i,j} = c_j \nabla F(x) + d_j v_j \). Then we have that

\[
y_{i,j+1} = \nabla F(x) + (I - \tilde{\nabla}^2 f_{i,j+1})y_{i,j}
= \nabla F(x) + (I - \lambda I - \psi_{s(i,j)}'(a_{s(i,j)}^T x)a_{s(i,j)}a_{s(i,j)}^T)y_{i,j}
= \nabla F(x) + ((1 - \lambda)I - \psi_{s(i,j)}'(a_{s(i,j)}^T x)a_{s(i,j)}a_{s(i,j)}^T)(c_j \nabla F(x) + d_j v_j)
= (1 + (1 - \lambda)c_j) \nabla F(x) + (1 - \lambda)(d_j v_j) - \psi_{s(i,j)}'(a_{s(i,j)}^T x)a_{s(i,j)}a_{s(i,j)}^T(c_j \nabla F(x) + d_j v_j)
= c_{j+1} \nabla F(x) + d_{j+1} v_{j+1}.
\]

Therefore to perform steps 3-12 in Algorithm 3 we only need to maintain \( c_j, d_j, v_j \).

Note that updating \( c_j \) and \( d_j \) takes constant time. Further owing to the fact \( a_{s(i,j)} \) has at most \( s(A) \) non-zeros we get that \( v_j \) can each be updated in \( O(s(A)) \) time. It can also be seen that \( v_{j+1} \) has at most \( s(A) \) more non-zeros as compared to \( v_j \). Therefore since \( v_0 \) is 0-sparse, the number of non-zero entries of \( v_j \) is at most \( j \cdot s(A) \). Finally since \( \nabla F(x) \) can be calculated in \( O(m \cdot s(A)) \) time, it follows that the total time to calculate \( y_i \) is \( O((m + S_1 \cdot S_2) \cdot s) \).

\[\text{2.4 Experiments}\]

In this section we present experimental evaluation for our theoretical results. We perform the experiments for a classification task over two labels using the logistic regression (LR) objective function with the \( \ell_2 \) regularizer. For all of the classification tasks we choose two values of \( \lambda: \frac{1}{n} \) and \( \frac{10}{m} \), where \( m \) is the number of training examples. We perform the above classification tasks over four data sets: MNIST, CoverType, Mushrooms, and RealSIM. Figure 2.1 displays the log-error achieved by LiSSA as compared to two standard first-order

\[\text{Our code for LiSSA can be found here: } \text{https://github.com/brianbullins/lissa_code}\]
Figure 2.1: Performance of LiSSA as compared to a variety of related optimization methods for different data sets and choices of regularization parameter $\lambda$. $S_1 = 1$, $S_2 \sim \kappa \ln(\kappa)$.

Figure 2.2: Convergence of LiSSA over time/iterations for logistic regression with MNIST, as compared to NewSamp and Newton’s method.

algorithms, SVRG and SAGA [54, 54], in terms of the number of passes over the data. Figure 2.2 presents the performance of LiSSA as compared to NewSamp [60] and standard Newton’s method with respect to both time and iterations.

**Experiment Details**: In this section we describe our experiments and choice of parameters in detail. Table 2.2 provides details regarding the data sets chosen for the experiments. To make sure our functions are scaled such that the norm of the Hessian is bounded, we scale the above data set points to unit norm.
Table 2.2: A description of data sets used in the experiments.

<table>
<thead>
<tr>
<th>DATA SET</th>
<th>M</th>
<th>D</th>
<th>REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST4-9</td>
<td>11791</td>
<td>784</td>
<td>[89]</td>
</tr>
<tr>
<td>MUSHROOMS</td>
<td>8124</td>
<td>112</td>
<td>[95]</td>
</tr>
<tr>
<td>COVERTYPE</td>
<td>100000</td>
<td>54</td>
<td>[31]</td>
</tr>
<tr>
<td>REALSIM</td>
<td>72309</td>
<td>20958</td>
<td>[101]</td>
</tr>
</tbody>
</table>

Comparison with Standard Algorithms: In Figures 2.1 and 2.3 we present comparisons between the efficiency of our algorithm with different standard and popular algorithms. In both cases we plot $\log(CurrentValue - OptimumValue)$. We obtained the optimum value for each case by running our algorithm for a long enough time until it converged to the point of machine precision.

Epoch Comparison: In Figure 2.1 we compare LiSSA with SVRG and SAGA in terms of the accuracy achieved versus the number of passes over the data. To compute the number of passes in SVRG and SAGA, we make sure that the inner stochastic gradient iteration in both the algorithms counts as exactly one pass. This is done because although it accesses gradients at two different points, one of them can be stored from before in both cases. The outer full gradient in SVRG counts as one complete pass over the data. We set the number of inner iterations of SVRG to $2m$ for the case when $\lambda = 1/m$, and we parameter tune the number of inner iterations when $\lambda = 10/m$. The stepsizes for all of the algorithms are parameter tuned by an exhaustive search over the parameters.

Time Comparison: For the comparison with respect to time (Figure 2.3), we consider the following algorithms: AdaGrad [57], BFGS [37, 63, 75, 132], gradient descent, SGD, SVRG [84] and SAGA [54]. The $\log(\text{Error})$ is plotted as a function of the time elapsed from the start of the run for each algorithm. We next describe our choice of parameters for the algorithms. For AdaGrad we used the faster diagonal scaling version proposed by [57]. We implemented the basic version of BFGS with backtracking line search. In each experiment for gradient descent, we find a reasonable step size using parameter tuning. For stochastic gradient descent, we use the variable step size $\eta_t = \gamma/\sqrt{t}$ which is usually the
prescribed step size, and we hand tune the parameter $\gamma$. The parameters for SVRG and SAGA were chosen in the same way as before.

*Choice of Parameters for LiSSA:* To pick the parameters for our algorithm, we observe that it exhibits smooth behavior even in the case of $S_1 = 1$, so this is used for the experiments. However, we observe that increasing $S_2$ has a positive effect on the convergence of the algorithm up to a certain point, as a higher $S_2$ leads to a larger per-iteration cost. This behavior is consistent with the theoretical analysis. We summarize the comparison between the per-iteration convergence and the value of $S_2$ in Figure 2.4. As the theory predicts $S_2$ to be of the order $\kappa \ln(\kappa)$, for our experiments we determine an estimate for $\kappa$ and set $S_2$ to around $\kappa \ln(\kappa)$. This value is typically equal to $m$ in our experiments. We observe that setting $S_2$ in this way resulted in the experimental results displayed in Figure 2.1.

*Comparison with Second-Order Methods:* Here we present details about the comparison between LiSSA, NewSamp [60], and standard Newton’s method, as displayed in Figure 2.2. We perform this experiment on the MNIST data set and show the convergence properties of all three algorithms over time as well as over iterations. We could not replicate the results of NewSamp on all of our data sets as it sometimes seems to diverge in our experiments. For logistic regression on the MNIST data set, we could get it to converge by setting the value of $S_1$ to be slightly higher. We observe as is predicted by the theory that when compared in terms of the number of iterations, NewSamp and LiSSA perform similarly, while Newton’s method performs the best as it attains a quadratic convergence rate. This can be seen in Figure 2.2. However, when we consider the performance in terms of time for these algorithms, we see that LiSSA has a significant advantage.

*Comparison with Accelerated First-Order Methods:* Here we present experimental results comparing LiSSA with a popular accelerated first-order method, APCG [97], as seen in Figure 2.5. We ran the experiment on the RealSIM data set with three settings of $\lambda = 10^{-5}$, $10^{-6}$, $10^{-7}$, to account for the high condition number setting. We observe a trend that can be expected from the runtime guarantees of the algorithms. When $\lambda$ is not too
low, LiSSA performs better than APCG, but as $\lambda$ gets very low we see that APCG performs significantly better than LiSSA. This is not surprising when considering that the running time of APCG grows proportional to $\sqrt{\kappa m}$, whereas for LiSSA the running time can at best be proportional to $\kappa$. We note that for accelerated first-order methods to be useful, one needs the condition number to be quite large which is not often the case for applications. Nevertheless, we believe that an algorithm with running time guarantees similar to LiSSA-Sample can get significant gains in these settings, and we leave this exploration as future work.

Figure 2.3: Performance of LiSSA as compared to a variety of related optimization methods for different data sets and choices of regularization parameter $\lambda$. $S_1 = 1$, $S_2 \sim \kappa \ln(\kappa)$. 

36
2.5 LiSSA: Extensions

In this section we describe a generic reduction from convex optimization to solving systems of linear equations approximately. The reduction is a direct application of Newton’s Method. In particular, we formally describe the algorithm LiSSA-Quad (Algorithm 2) and provide its runtime guarantee (Theorem 2.5.2). The key idea underlying this algorithm is that Newton’s method essentially reduces a convex optimization problem to solving intermediate quadratic sub-problems given by the second-order Taylor approximation at a point, i.e., the
sub-problem $Q_t$ given by

$$Q_t(y) = F(x_{t-1}) + \nabla F(x_{t-1})^T y + \frac{y^T \nabla^2 F(x_{t-1}) y}{2}$$

where $y = x - x_{t-1}$. The above expansion provides an alternative interpretation of the estimator for $\nabla^2 F(x)$ used in LiSSA. Consider running gradient descent on the above quadratic $Q_t$, and let $y^{(i)}_t$ be the $i^{th}$ step in this process. By definition we have that

$$y^{(i+1)}_t = y^{(i)}_t - \nabla Q_t(y^{(i)}_t) = (I - \nabla^2 F(x_{t-1})) y^{(i)}_t - \nabla F(x_{t-1}).$$

It can be seen that the above expression corresponds exactly to the steps taken in LiSSA (Algorithm 3, line 8) with the difference being that we use a sample of the Hessian instead of the true Hessian. Therefore LiSSA can also be interpreted as doing a partial stochastic gradient descent on the quadratic $Q_t$. It is partial because we have a precise estimate of gradient of the function $F$ and a stochastic estimate for the Hessian. We believe that this nuance is essential for the linear convergence guarantees we show for LiSSA.

While LiSSA presents a particular method for minimizing the quadratic $Q_t$, the above interpretation allows for any method that is suited to minimizing quadratic functions to be used as a subsolver. In particular, consider the following definition

**Definition 2.5.1 (Quadratic Solver).** Given a quadratic function $Q(y) = \frac{y^T A y}{2} + b^T y$ and a probability value $\delta \in (0, 1)$, a randomized algorithm is a $T^Q(\delta)$ time quadratic solver if for any $\varepsilon > 0$, given $y_0$, with probability at least $1 - \delta$, it produces a point $y$ such that

$$\|y - y^*\| \leq \varepsilon \|y_0 - y^*\|,$$  \hspace{1cm} (2.5.1)

where $y^* = \text{argmin}_y Q(y)$, in total time $T^Q(\delta) \log(\varepsilon^{-1})$.

Given a Quadratic Solver ALG, LiSSA-Quad, as described in Algorithm 2 generically implements the above idea, modifying LiSSA by replacing the inner loop with the given
Algorithm 2 LiSSA-Quad

**Input:** $T, F(x) = \sum_{k=1}^{n} f_k(x), \text{ALG}, \varepsilon > 0, \delta \in (0, 1), x_0$

**for** $t = 1$ **to** $T$ **do**

\[ Q_t(y) = \nabla F(x_{t-1})^T y + \frac{y^T \nabla^2 F(x_{t-1}) y}{2} \]

\[ x_t = \text{ALG}(Q_t, \varepsilon, \delta / T) \]

**end for**

**return** $\text{argmin}_t F(x_t)$

algorithm ALG. The following is a meta-theorem about the convergence properties of LiSSA-Quad.

**Theorem 2.5.2.** Given a GLM function $F(x)$ which is $\alpha$-strongly convex, let $x^*$ be the minimizer of the function and $\{x_t\}$ be defined as in Algorithm 2. Suppose the algorithm ALG is $T^Q(\delta) \text{ time quadratic solver}$ and suppose $x_0$ is such that

\[ \|x_0 - x^*\| \leq \frac{1}{2 \left( \frac{2L_3}{\alpha} + 1 \right)} . \]

Further given $\delta \in (0, 1)$, set the parameters in Algorithm 2 as follows: $T = \log \log (1/\varepsilon)$, where $\varepsilon$ is the final error guarantee one wishes to achieve. Then we have that after $T$ steps, with probability at least $1 - \delta$,

\[ \min_{t \in \{1...T\}} \|x_t - x^*\| \leq \varepsilon. \]

In particular, LiSSA-Quad(ALG) produces a point $x$ such that

\[ \|x - x^*\| \leq \varepsilon \]

in total time $O(T^Q(\delta / T) \log (1/\varepsilon) \log \log (1/\varepsilon))$ with probability at least $1 - \delta$.

We now provide a proof for Theorem 2.5.2

**Proof of Theorem 2.5.2** We run the algorithm ALG to achieve accuracy $\varepsilon$ on each of the intermediate quadratic functions $Q_t$, and we set $\delta_{\text{ALG}} = \delta / T$ which implies via a union
bound that for all $t \leq T$,
\[
\|x_{t+1} - x^*\| \leq \varepsilon \|x_t - x^*\| 
\] (2.5.2)
with probability at least $1 - \delta$, where $x_t^* = \arg\min_x Q_t(x)$. Assume that for all $t < T$, $\|x_t - x^*\| \geq \varepsilon$ (otherwise the theorem is trivially true). Using the analysis of Newton’s method as before, we get that for all $t < T$,
\[
\|x_{t+1} - x^*\| \leq \|x_t^* - x^*\| + \|x_{t+1} - x_t^*\|
\leq \|x_t^* - x^*\| + \varepsilon \|x_t - x_t^*\|
\leq (1 + \varepsilon) \|x_t^* - x^*\| + \varepsilon \|x_t - x^*\|
\leq 2 \|x_t - \nabla^{-2} F(x_t)\nabla F(x_t) - x^*\| + \varepsilon \|x_t - x^*\|
\leq \frac{2L_3}{\alpha} \|x_t - x^*\|^2 + \varepsilon \|x_t - x^*\|
\leq \left(\frac{2L_3}{\alpha} + 1\right) \|x_t - x^*\|^2
\]
where the first and the third inequalities follow from the triangle inequality, the second inequality follows from the fact that ALG is an $\varepsilon$-accuracy quadratic solver and the fourth inequality follows from the analysis in the proof of Theorem 2.3.3 and Equation (2.5.2).
Since we know that $\|x_0 - x^*\| \leq \frac{1}{2(\frac{2L_3}{\alpha} + 1)}$, applying the above inductively and using $T = \log(\log(1/\varepsilon))$ prescribed by the theorem statement, we get that $\|x_T - x^*\| \leq \varepsilon$. 

Note that for GLM functions, the $\nabla Q_t(y)$ at any point can be computed in time linear in $d$. In particular, a full gradient of $Q_t$ can be computed in time $O(nd)$ and a stochastic gradient (corresponding to a stochastic estimate of the Hessian) in time $O(d)$. Therefore, a natural choice for the algorithm ALG in the above are first-order algorithms which are linearly convergent, for example SVRG, SDCA, and Acc-SDCA. Choosing a first-order algorithm FO gives us a family of algorithms LiSSA-Quad(FO), each with running time comparable to the running time of the underlying FO, up to logarithmic factors. The
following corollary summarizes the typical running time guarantees for LiSSA-Quad(FO) when FO is Acc-SVRG.

**Corollary 2.5.3.** Given a GLM function $F(x)$, if ALG is replaced by Acc-SVRG [96], then under a suitable setting of parameters, LiSSA-Quad produces a point $x$ such that

$$F(x) - F(x^*) \leq \varepsilon$$

with probability at least $1 - \delta$, in total time $\tilde{O}(n + \min\{\sqrt{\kappa_l n}, \tilde{\kappa}d\})d \log(1/\varepsilon) \log \log(1/\varepsilon)$.

Here the $\tilde{O}$ above hides logarithmic factors in $\kappa, d, \delta$, but not in $\varepsilon$. Note that the above running times depend upon the condition number $\tilde{\kappa}$ which as described in Section 2.2 can potentially provide better dependence compared to its global counterpart. In practice this difference could lead to faster running time for LiSSA-Quad as compared to the underlying first-order algorithm FO.

In the next chapter we provide a faster algorithm for regression (Theorem 3.1.5) which in turn provides a faster algorithm for solving quadratics which can be used in LiSSA-Quad. This algorithm whose guarantees are formally summarized in Theorem 3.1.10 improves over Corollary 2.5.3 by replacing the term $\sqrt{\kappa \cdot n}$ by $\sqrt{\kappa \cdot d}$ thereby significantly improving the runtime when $n >> d$.

### 2.6 Conclusion

In this chapter we proposed a linear time second-order stochastic optimization algorithm LiSSA which made use of the Neumann series for the matrix inverse to create an unbiased estimator. We provided convergence guarantees for our method akin to those of known fast gradient based methods. We showed that the method can also be implemented in time proportional to the sparsity of the underlying data. Further we provided an experimental evaluation of LiSSA, pitting it against the state-of-the-art gradient based methods. Finally
we formulated a generic reduction from solving systems of linear equations (or a convex quadratic) to convex optimization via Newton’s method. While the focus in this chapter was to provide a practical second-order algorithm, in the next chapter we focus on improving the best known run-time guarantee in theory for GLM minimization. We achieve this by first focusing on linear regression as a sub-problem and then using that result as a sub-routine in LiSSA-Quad.
Chapter 3

Leverage Score Sampling for Faster Accelerated Regression and GLM

3.1 Introduction

Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, the regression problem $\min_{x \in \mathbb{R}^d} \frac{1}{2} \|Ax - b\|_2^2$ is one of the most fundamental problems in optimization and a prominent tool in machine learning. It is one of the simplest empirical risk minimization (ERM) problems and a prominent proving ground for developing new provably efficient algorithms for solving large scale learning problems.

Regression is long known to be solvable directly by fast matrix multiplication in $O(nd^{\omega-1})$ time where $\omega < 2.373$\footnote{Through this chapter, we use $\tilde{O}$ to hide factors polylogarithmic in $n,d,\lambda_{\max}(A^T A),1/\lambda_{\min}(A^T A)$, and $L_3$ (c.f. Definition 3.1.3 and Assumptions A3.1.7 and A2.2.3).} is the matrix multiplication constant and recent work has improved the running time to $\tilde{O}((m\|A\| + d^{\omega}) \log(1/\epsilon))$, i.e. linear time plus the time needed to solve a nearly square linear system \cite{50,94,107,52,51}. However, for large $A$ even a super-quadratic running time of $\Omega(d^{\omega})$ can be prohibitively expensive. Consequently, over the past decade improving this running time under mild regularity assumptions on $A$ has been an active area of research.
In this chapter we improve the best known running time for solving regression under standard regularity assumptions. Formally the problem we consider is as follows.

**Definition 3.1.1 (The Regression Problem).** Given $A \in \mathbb{R}^{n \times d}$ with rows $a_1, \ldots, a_n$ and $b \in \mathbb{R}^n$, we consider the regression problem $\min_{x \in \mathbb{R}^d} f_{A,b}(x)$ where

$$f_{A,b}(x) = \frac{1}{2} \|Ax - b\|_2^2 = \sum_{i=1}^n \frac{1}{2} (a_i^T x - b_i)^2.$$ 

The central problem of this chapter is to get faster regression algorithms defined as follows.

**Definition 3.1.2 (Regression Algorithm).** We call an algorithm a $T(A)$-time regression algorithm if for any $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^d$, and $\varepsilon \in (0, \frac{1}{2})$ with high probability (w.h.p) in $n$ in time $O(T(A) \log \varepsilon^{-1})$ the algorithm outputs a vector $y$ such that

$$f_{A,b}(y) - \min_x f_{A,b}(x) \leq \varepsilon \cdot \left( f_{A,b}(x_0) - \min_x f_{A,b}(x) \right).$$ (3.1.1)

Note that if $x^*$ is a minimizer of $f_{A,b}(x)$ then the guarantee (3.1.1) is equivalent to the following

$$\|y - x^*\|_{A^T A}^2 \leq \varepsilon \|x_0 - x^*\|_{A^T A}^2$$ (3.1.2)

where $\|x\|_M^2 \triangleq x^T M x$ for $M \succeq 0$. The goal of this chapter is to provide regression algorithms with improved running times depending on $n$, $d$, and the following regularity parameters.

**Definition 3.1.3. (Regularity Parameters)** We let $\lambda_{\min}(A^T A)$ and $\lambda_{\max}(A^T A)$ denote the smallest and largest eigenvalues of $A^T A$. We let $\kappa(A^T A) = \lambda_{\max}(A^T A) / \lambda_{\min}(A^T A)$ denote the condition number of $A^T A$ and let $\kappa_{\text{sum}}(A^T A) = \text{Tr}(A^T A) / \lambda_{\min}(A^T A)$ denote the total condition number of $A^T A$. We let $s(A)$ denote the maximum number of non-zero entries in a row of $A$. Occasionally, we drop the terms in parenthesis when they are clear from context.
Regression Running Times

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Matrix Multiplication</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Gradient Descent</td>
<td>$\tilde{O}(\text{nnz}(A)\kappa \log(1/\varepsilon))$</td>
</tr>
<tr>
<td>Fast Matrix Multiplication [142]</td>
<td>$O(n^\omega^\varepsilon)$ where $\omega &lt; 2.373$</td>
</tr>
<tr>
<td>Accelerated Gradient Descent [111]</td>
<td>$\tilde{O}(\text{nnz}(A)\sqrt{\kappa} \log(1/\varepsilon))$</td>
</tr>
<tr>
<td>Row Sampling [94, 52], Subspace Embeddings [107]</td>
<td>$\tilde{O}(\text{nnz}(A) + \kappa_{\text{sum}} s) \log(1/\varepsilon))$</td>
</tr>
<tr>
<td>SAG [124, 54], SVRG [84]</td>
<td>$\tilde{O}(\text{nnz}(A) + s \sqrt{n \kappa_{\text{sum}}} \log(1/\varepsilon))$</td>
</tr>
<tr>
<td>App. Prox. Point [64], Catalyst [96], Katyusha [11]</td>
<td>$\tilde{O}(\text{nnz}(A) + s \sqrt{d \kappa_{\text{sum}}} \log(1/\varepsilon))$</td>
</tr>
<tr>
<td>This Paper</td>
<td>$\tilde{O}(\text{nnz}(A) + s \sqrt{d \kappa_{\text{sum}}} \log(1/\varepsilon))$</td>
</tr>
</tbody>
</table>

Table 3.1: History of improvements to the running time for regression in terms of the problem parameters nnz(A), n, d, $\kappa \triangleq \kappa(A^\top A)$, $\kappa_{\text{sum}} \triangleq \kappa_{\text{sum}}(A^\top A)$, and $s \triangleq s(A)$.

In this chapter we first provide an $\tilde{O}((n + \sqrt{d \cdot \kappa_{\text{sum}}})s \log(1/\varepsilon))$ time algorithm for solving regression, improving upon the previous best running time of $\tilde{O}((n + \sqrt{n \cdot \kappa_{\text{sum}}})s \log(1/\varepsilon))$. (See Table 3.1 for the history of running time improvements to this problem.). Further we provide extensions of our results to GLMs under two different regularity assumptions.

### 3.1.1 Previous Results

Classic iterative methods such as gradient descent and accelerated gradient descent [111] solve the regression problem with running times of $O(n \cdot s(A) \cdot \kappa(A^\top A))$ and $O(n \cdot s(A) \cdot \sqrt{\kappa(A^\top A)})$ respectively. While these running times are super-linear whenever $\kappa(A^\top A)$ is super constant there has been a flurry of recent papers showing that using sampling techniques faster running times can be achieved. These often yield nearly linear running times when $n$ is sufficiently larger than $d$ [129, 84, 130, 11].

Using recent advances in accelerated coordinate descent [18, 116] coupled with proximal point methods [64, 96] the previous fastest iterative algorithm is as follows:
**Theorem 3.1.4** (Previous Best Regression Running Times). *Given* \(A \in \mathbb{R}^{n \times d}\), *there is a* \(\mathcal{T}(A)\)-time regression algorithm with

\[
\mathcal{T}(A) = \tilde{O}\left( n + \frac{\sum_{i \in [n]} \|a_i\|^2}{\sqrt{\lambda_{\min}(A^\top A)}} \cdot s(A) \right) = \tilde{O}\left( (n + \sqrt{n \cdot \kappa_{\text{sum}}(A^\top A)}) \cdot s(A) \right).
\]

The inequality in this theorem follows directly from Cauchy Schwartz, as

\[
\sum_{i \in [n]} \|a_i\|_2 \leq \sqrt{n \cdot \text{Tr}(A^\top A)} = \sqrt{n \cdot \kappa_{\text{sum}}(A^\top A) \cdot \lambda_{\min}(A^\top A)}.
\]

We provide a generalization of the above theorem as Theorem 3.2.1 which is more useful to our analysis and provide a proof in Section 3.6.1.

### 3.1.2 Main Results

#### Results for Regression

The results in this chapter are motivated by the natural question, *Can this running time of Theorem 3.1.4 be further improved?* Despite the running time lower bound of \(\sqrt{n \cdot \kappa_{\text{sum}}(A^\top A)}\) shown in [143] in this chapter we give an affirmative answer improving the \(\sqrt{n \cdot \kappa_{\text{sum}}(A^\top A)}\) term in Theorem 3.1.4 to \(\sqrt{d \cdot \kappa_{\text{sum}}(A^\top A)}\). The main result of this chapter is the following:

**Theorem 3.1.5** (Improved Regression Running Time). *Given* \(A \in \mathbb{R}^{n \times d}\), *Algorithm 7* is a \(\mathcal{T}(A)\)-time regression algorithm that succeeds w.h.p in \(n\) where

\[
\mathcal{T}(A) = \tilde{O}\left( \text{nnz}(A) + \left( d + \frac{\sum_{i \in [n]} \|a_i\|^2 \cdot \sqrt{\sigma_i(A^\top A)}}{\sqrt{\lambda_{\min}(A^\top A)}} \right) \cdot s(A) \right)
\]

and \(\sigma_i(A) = \|a_i\|^2_{(A^\top A)^{-1}}\) *for all* \(i \in [n]\).

\[\text{Their lower bound involves a function with } d \gg n. \text{ However, } d \ll n \text{ is more common as we explain.}\]
Up to polylogarithmic factors Theorem [3.1.5] is an improvement over Theorem [3.1.4] as \( \sigma_i(A) \in [0, 1] \). This improvement can be substantial as \( \sigma_i(A) \) can be as small as \( O(d/n) \), e.g. if \( A \) is an entry-wise random Gaussian matrix. Compared to Theorem [3.1.4] whose second term in running time grows as \( n \), our second term is always independent of \( n \) due to the following:

\[
\frac{\sum_{i \in [n]} \|a_i\| \cdot 2 \sigma_i(A)}{\sqrt{\lambda_{\min}(A^\top A)}} \leq \sqrt{\frac{\sum_{i \in [n]} \|a_i\|^2 \sum_{i \in [n]} \|a_i\|^2 (A^\top A)^{-1}}{\lambda_{\min}(A^\top A)}} = \sqrt{\frac{\text{Tr}(A^\top A) \cdot \text{Tr}(A(A^\top A)^{-1}A^T)}{\lambda_{\min}(A^\top A)}} \leq \sqrt{d \kappa_{\text{sum}}(A^\top A)}. \tag{3.1.3}
\]

Therefore in Theorem [3.1.5] we have \( \mathcal{T}(A) = \tilde{O}((n + \sqrt{d \cdot \kappa_{\text{sum}}(A^\top A)}) \cdot s(A)) \).

This improvement from \( n \) to \( d \) can be significant as \( n \) (the number of samples) is in some cases orders of magnitude larger than \( d \) (the number of features). For example, in the LIBSVM dataset\(^3\) in 87 out of 106 non-text problems, we have \( n \geq d \), 50 of them have \( n \geq d^2 \) and in the UCI dataset\(^4\) in 279 out of 301 non-text problems, we have \( n \geq d \), 195 out of them have \( n \geq d^2 \).

**Results for GLM functions**

Furthermore, in Section [3.4] we show how to extend our results to GLM functions (defined in Definition [2.1.1]) which are more general than regression. We restate the definition here for concreteness.

**Definition 3.1.6 (GLM).** Given a matrix \( A \in \mathbb{R}^{n \times d} \) with rows \( a_1, \ldots, a_n \) and functions \( \{\psi_1, \ldots, \psi_n\} \in \mathbb{R} \rightarrow \mathbb{R} \) such that each \( \psi_i : \mathbb{R} \rightarrow \mathbb{R} \) is convex and twice differentiable, define \( F(x) : \mathbb{R}^d \rightarrow \mathbb{R} \) as

\[
F(x) \triangleq \sum_{i=1}^n f_i(x) = \sum_{i=1}^n \psi_i(a_i^T x).
\]

\(^3\)https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
\(^4\)http://archive.ics.uci.edu/ml/datasets.html
Further given a convex function $R(x) : \mathbb{R}^d \to \mathbb{R}$ (referred to as the regularizer), we wish to minimize

$$\min_{x \in \mathbb{R}^d} F(x) + R(x)$$

Note that we removed the factor of $\frac{1}{n}$ in the definition of $F(x)$ above as compared to Definition 2.1.1 for convenience of notation. This makes no difference to the results in this chapter as the measure of error in our theorems is either relative multiplicative error or distance from the solution in parameter space both of which are unaffected under this change.

As in Chapter 2 we will assume that the regularizer $R(x) = \lambda \|x\|^2$. Further note that due to the structure of the regularizer we will assume that $A$ is redefined as $A = [A, \sqrt{\lambda}I]$ and $b$ is redefined as $b = [b, 0]$. Under this redefinition without loss of generality, we can assume $R(x) = 0$ in Definition 3.1.6.

We will prove two results under different third-order regularity assumption on the GLM function. The first assumption requires the following per-component third-order bound.

**Assumption 3.1.7.** Each $\psi_i : \mathbb{R} \to \mathbb{R}$ is twice differentiable and satisfies

$$\forall x \in \mathbb{R}^d \quad \frac{1}{L_3} \leq \psi''_i(x) \leq L_3$$

The following is our main theorem regarding GLM functions under A3.1.7.

**Theorem 3.1.8.** Given a GLM function (Definition 3.1.6) satisfying A3.1.7 and an initial point $x_0$, there exists an algorithm that produces a point $x'$ such that

$$F(x') - \min_{x \in \mathbb{R}^d} F(x) \leq \varepsilon \left( F(x_0) - \min_{x \in \mathbb{R}^d} F(x) \right)$$

which succeeds w.h.p in $n$ in total time

$$\tilde{O} \left( \left( \text{nnz}(A) + \left( dL_3^5 + \sum_{i=1}^n \frac{\|a_i\|_2 \sqrt{\sigma_i L_3^2}}{\sqrt{\lambda_{\min}(A^T A)}} \right) s(A) \right) \log \left( \frac{1}{\varepsilon} \right) \right)$$
where $\sigma_i \triangleq \|a_i\|^2_{[A^TA]^{-1}}$ are the leverage scores with respect to $a_i$.

A\ref{assumption:erm} is satisfied by many ERM problems where the $f_i(x)$ are regularized directly. For example, the $\delta$-regularized logistic function $f(x) = \log(1 + \exp(-x)) + \frac{\delta}{2}x^2$ satisfies $\delta \leq f''(x) \leq 1/4 + \delta$ for all $x$ and therefore under appropriate rescaling yields $L_3 = 1/\sqrt{\delta}$ which may be small in many instances.

Further we propose the following different regularity assumption which bounds the norm of the third derivative of the GLM function $F(x)$. This is the same assumption under which we showed our results in Chapter\ref{chapter:glm}.

**Assumption 3.1.9.** $\nabla^2 F(x)$ is $L_3$-Lipschitz with respect to the $\| \cdot \|$ norm, i.e.

$$\forall \ x, y \: \| \nabla^2 F(x) - \nabla^2 F(y) \| \leq L_3 \| x - y \|$$

Under the above assumption we show the following theorem.

**Theorem 3.1.10.** Consider a GLM function $F(x)$ (Definition\ref{definition:glm}) satisfying A\ref{assumption:glm} Further suppose the GLM function is $\alpha$ strongly convex and define $x^* = \arg\min_{x \in \mathbb{R}^d} F(x)$. Given an initial point $x_0$, such that

$$\| x_0 - x^* \| \leq \frac{1}{2 \left( \frac{2L_3}{\alpha} + 1 \right)},$$

and $\varepsilon \in (0, 1)$, there exists an algorithm that produces a point $x'$ such that

$$\| x' - x^* \| \leq \varepsilon$$

which succeeds w.h.p in $n$ in total time

$$\tilde{O} \left( \left( \text{nnz}(A) + \left( d + \sqrt{d \cdot \max_{x \in \mathbb{R}^d} \sum_{i=1}^m \psi''(a_i^T x) \|a_i\|^2} \right) s(A) \right) \log \left( \frac{1}{\varepsilon} \right) \log \left( \log \left( \frac{1}{\varepsilon} \right) \right) \right).$$
Discussion of the results

Note that both Theorems 3.1.8 and 3.1.10 interpolate our regression results, i.e. it recovers our results for regression in the special case of $L_3 = 1$. To better understand the bound in Theorem 3.1.8, note that following the derivation in Equation (3.1.3) we have that the running time in Theorem 3.1.8 is bounded by

$$\tilde{O} \left( \left( \text{nnz}(A) + \sum_{i=1}^{n} L_3^2 \sqrt{dn} \kappa_{\text{sum}}(A^T A) \right) s(A) \log \left( \frac{1}{\varepsilon} \right) \right).$$

The best known bound for GLM problems as defined in Definition 3.1.6 given by [64, 96, 12] is

$$\tilde{O} \left( \left( \text{nnz}(A) + \sum_{i=1}^{n} L_3 \sqrt{n \kappa_{\text{sum}}(A^T A)} \right) s(A) \log \left( \frac{1}{\varepsilon} \right) \right) \quad (3.1.4)$$

In this case Theorem 3.1.8 should be seen as implying that under A3.1.7, the effective dependence on the number of examples on the running time for GLM functions can be reduced to at most $dL_3^5$.

The key difference between the results derived in Theorem 3.1.8 and 3.1.10 is the requirement of a warm start in Theorem 3.1.10. Apriori (as remarked in Chapter 2), while the time to achieve the warm start does not depend on $\varepsilon$, however the best known algorithm to achieve such a warm start would be the guarantee given by accelerated first-order methods [64, 96, 12] as detailed in (3.1.4). Thus while Theorem 3.1.10 achieves a faster run-time under a more standard assumption, it is applicable only when $\varepsilon \rightarrow 0$.

Bridging this gap between Theorems 3.1.8 and 3.1.10 is an intriguing question which is left for future work.

### 3.1.3 Our Approach

Our algorithm follows from a careful combination and analysis of a recent suite of advances in numerical linear algebra. First, we use the previous fastest regression algorithm,
Theorem 3.1.4, which is the combination of recent advances in accelerated coordinate descent [18, 116] and proximal point methods [64, 96]. Then, we show that if we have estimates of the leverage scores of the rows of $A$, a natural recently popularized measure of importance, [136, 94, 52], we can use concentration results on leverage score sampling and preconditioning to obtain a faster regression algorithm. (See Section 3.2.)

A powerful well known fact is that given a regression algorithm leverage scores can be estimated in nearly linear time plus the time needed to solve $O(1)$ regression problems [136]. Consequently, to achieve the improved running time when we do not have leverage scores we are left with a chicken and egg problem. Fortunately, recent work [94, 52] has shown that such a problem can be solved in several ways. We show that the technique in [94] carefully applied can be used to obtain our improved running time for both estimating leverage scores and solving regression problems with little overhead. (See Section 3.3.)

For application to a broader class of GLM problems under A3.1.7 most parts of the regression procedure generalize naturally. The key ingredient is the generalization of preconditioning to the case when we are sampling non-quadratic functions. For this, we prove concentration results on sampling from GLM functions inspired from [65] to show that it suffices to solve the problem on a sub-sample of the components that may be of intrinsic interest (See Section 3.4). The application to GLM functions under 3.1.9 follows from the analysis LiSSA-Quad meta algorithm (Algorithm 2) by using the regression routine from Theorem 3.1.5 as a quadratic sub-solver.

In summary our algorithms are a careful blend of accelerated coordinate descent and concentration results coupled with the iterative procedure in [94] and the Johnson Lindenstrauss machinery of [136] to compute leverage scores. The algorithms we provide are fairly straightforward, but it provides a substantial running time improvement that we think is of intrinsic interest.

Finally, we remark that there is another way to achieve the $\sqrt{d \cdot \kappa_{\text{sum}}(A)}$ improvement over $\sqrt{n \cdot \kappa_{\text{sum}}(A)}$. One could use subspace embeddings [50, 107, 51] and preconditioning
to reduce the regression problem to a regression problem on a $\tilde{O}(d) \times d$ matrix and then apply Theorem 3.1.4 to solve the $\tilde{O}(d) \times d$ regression problem. While this works, it has three shortcomings relevant to our approach. Firstly note that even if the original rows of the matrix are $s$ sparse, the the rows of the sketched matrix might become $\Omega(d)$ sparse, and the final running time would have an additional $\tilde{O}(d^2)$ term our method does not. Second, it is unclear if this approach yields our more fine-grained running time dependence on leverage scores that appears in Theorem 3.1.5 which we believe to be significant. Thirdly it is unclear how to extend the approach to the GLM setting under 3.1.7 for which we can prove our running time bounds without the requirement of a warm start.

### 3.1.4 Chapter Organization

The chapter is organised as follows. We prove Theorem 3.1.5 in Sections 3.2 and 3.3. We first provide the algorithm for regression given leverage score estimates in Section 3.2 and further provide the algorithm to compute the estimates in Section 3.3. Note that the algorithm for computing leverage scores makes use of the algorithm for regression given leverage scores as a sub-routine. In Section 3.4 we provide Algorithm 4 to solve GLM problems under A3.1.7. In this section we provide the proof of Theorem 3.1.8. Further in Section 3.5 we provide the proof of Theorem 3.1.10. Finally in Section 3.6 we provide the proofs of lemmas/theorems which were deferred from the rest of the sections. This section provided for the sake of completeness is a collection of proofs for existing lemmas/theorems which we have restated in a form convenient for our use in the main sections.

### 3.2 Regression Algorithm Given Leverage Score Estimates

The regression algorithm we provide in this chapter involves two steps. First we find which rows of $A$ are important in terms of leverage score. Second, we use these leverage scores
to sample the matrix and solve the regression problem on the sampled matrix using
the following theorem which is a generalization of Theorem 3.1.4 that is useful for our analysis.

**Theorem 3.2.1** (Previous Best Regression Running Time). Let $A$ and $B$ be matrices with the
same number of columns. Suppose that $B$ has $n$ rows and $(\frac{5}{6}) B^\top B \leq A^\top A \leq (\frac{6}{5}) B^\top B$, then there is a $T(A)$-time regression algorithm with
\[
T(A) = \tilde{O} \left( \text{nnz}(A) + \left( n + \frac{\sum_{i \in \{n\}} \| b_i \|_2}{\lambda_{\min}(B^\top B)} \right) \cdot s(B) \right).
\]

Theorem 3.2.1 is a consequence of results on accelerated coordinate descent [18, 116] and approximate proximal point [64, 96]. We defer the proof to Section 3.6.1. In the rest of the section we define leverage scores and analyze the second step of our algorithm.

**Definition 3.2.2** (Leverage Score). For $A \in \mathbb{R}^{n \times d}$ with rows $a_1, \ldots, a_n \in \mathbb{R}^d$ we denote the
leverage score of row $i \in [n]$ by $\sigma_i(A) \triangleq a_i^\top (A^\top A)^+ a_i$.

Note that $\sigma_i(A) \in (0, 1]$ for all $i \in [n]$ and $\sum_{i \in [n]} \sigma_i(A) = \text{rank}(A)$. The following
lemma shows that sampling rows of $A$ according to overestimates of leverage scores yields
a good approximation to $A$ after appropriate re-scaling [52, 136]:

**Lemma 3.2.3** (Leverage Score Sampling ([52]) ). Let $A \in \mathbb{R}^{n \times d}$, let $\delta \in (0, \frac{1}{2})$, and let $u \in \mathbb{R}^n$ be overestimates of leverage scores of $A$; i.e. $u_i \geq \sigma_i(A)$ for all $i \in [n]$. Define $p_i \triangleq \min \{ 1, k \delta^{-2} u_i \log n \}$ for a sufficiently large constant $k > 0$ and let $H \in \mathbb{R}^{n \times n}$ be a
random diagonal matrix where $H_{ii} = \frac{1}{p_i}$ independently with probability $p_i$ and $H_{ii} = 0$ otherwise. With high probability in $n$, $\text{nnz}(H) = O(d \cdot \delta^{-2} \cdot \log n)$ and $(1 - \delta) \cdot A^\top A \preceq A^\top H A \preceq (1 + \delta) \cdot A^\top A$.

Algorithm 1 outlines the procedure to solve regression given overestimates of leverage score.
Algorithm 1: SolveUsingLS\(_{A,u}(x_0, \varepsilon, u)\)

Let \( p_i = \min \{1, k' \cdot u_i \log n\} \) where \( k' \) is a sufficiently large absolute constant.

\[ \begin{align*}
\text{repeat} & \\
& \text{Let } H \in \mathbb{R}^{n \times n} \text{ be a diagonal matrix where independently for all } i \in [n] \text{ we let } H_{ii} = \frac{1}{p_i} \text{ with probability } p_i \text{ and 0 otherwise.} \\
& \text{Let } B = \sqrt{HA}. \\
\text{until } & \sum_{i \in [n]} \|b_i\|_2 \leq 2 \cdot \sum_{i \in [n]} \sqrt{k' \cdot u_i \log n} \cdot \|a_i\|_2; \\
\text{Invoke Theorem 3.2.1 on } A \text{ and } B \text{ to find } y \text{ such that} \\
& f_{A,b}(y) - \min_{x} f_{A,b}(x) \leq \varepsilon \cdot \left( f_{A,b}(x_0) - \min_{x} f_{A,b}(x) \right). \\
\text{Output: } y.
\end{align*} \]

Theorem 3.2.4. If \( u \in \mathbb{R}^n \) satisfies \( \sigma_i(A) \leq u_i \leq 4 \cdot \sigma_i(A) + [n \cdot \kappa(A^\top A)]^{-1} \) for all \( i \in [n] \) then SolveUsingLS\(_{A,u} \) is a \( \mathcal{T}(A) \)-time regression algorithm where

\[ \mathcal{T}(A) = \tilde{O}\left( \text{nnz}(A) + \left( d + \frac{\sum_{i \in [n]} \sqrt{\sigma_i(A) \cdot \|a_i\|_2}}{\sqrt{\lambda_{\text{min}}(A^\top A)}} \right) \cdot s(A) \right). \]

Proof. Let \( k' = \delta^{-2} \cdot k \) for \( \delta = \frac{1}{10} \). Applying Lemma [3.2.3] yields with high probability in \( n \) that

\[ \left( \frac{5}{6} \right) A^\top A \preceq A^\top HA \preceq \left( \frac{6}{5} \right) A^\top A \]  \hspace{1cm} (3.2.1)

where \( A^\top HA = \sum_{i \in [n]} b_i b_i^\top \), \( b_i = \frac{1}{\sqrt{p_i}} a_i \) and \( p_i = \min \{1, k' \cdot u_i \log n\} \). Note that

\[ \mathbb{E}\left[ \sum_{i \in [n]} \|b_i\|_2^2 \right] = \sum_{i \in [n]} \frac{p_i}{\sqrt{p_i}} \|a_i\|_2 \leq \sum_{i \in [n]} \sqrt{k' u_i \log n \cdot \|a_i\|_2}. \]

Consequently, by Markov’s inequality, with probability at least 1/2

\[ \sum_{i \in [n]} \|b_i\|_2 \leq 2 \cdot \sum_{i \in [n]} \sqrt{k' u_i \log n \cdot \|a_i\|_2}. \]

Therefore the loop in the algorithm terminates with high probability in \( n \) in \( O(\log n) \) iterations. Consequently, the loop takes only \( O(\text{nnz}(A) + n \log n) \)-time and since we only
sampled $O(\log n)$ many independent copies of $A^T HA$, the guarantee (3.2.1) again holds with high probability in $n$.

Using the guarantee (3.2.1) and Theorem 3.2.1 on $A$ and $B \doteq \sqrt{HA}$, we can produce a $y$ we need in time $O(\log(\varepsilon^{-1}))$ times

$$\tilde{O}\left(\left(nnz(A) + \left( d\log n + \frac{1}{\sqrt{\lambda_{\min}(A^T A)} \sum_{i \in [n]} \|b_i\|_2}\right) \cdot s(A) \right)\right)$$

where we used that $B$ has at most $O(d \log n)$ rows with high probability in $n$. Since we know

$$\sum_{i \in [n]} \|b_i\|_2 \leq 2 \sum_{i \in [n]} \sqrt{k' \cdot u_i \log n \cdot \|a_i\|_2},$$

all that remains is to bound $\sum_{i \in [n]} \sqrt{u_i} \|a_i\|_2$. However, $A^T A \leq \lambda_{\max}(A^T A) I$ and therefore

$$I \leq \lambda_{\max}(A^T A)(A^T A)^{-1} \text{ and } \|a_i\|_2 \leq \sqrt{\lambda_{\max}(A^T A) \cdot \sigma_i(A)}.$$

Consequently, Cauchy Schwartz and $\lambda_{\min}(A^T A) \leq \text{Tr}(A^T A)$ yields

$$\frac{1}{\sqrt{n}} \sum_{i \in [n]} \|a_i\|_2 \leq \sqrt{\sum_{i \in [n]} \|a_i\|_2^2} \leq -\frac{1}{\sqrt{\lambda_{\min}(A^T A)}} \sum_{i \in [n]} \|a_i\|_2^2 \leq \sqrt{\kappa(A^T A)} \sum_{i \in [n]} \sigma_i(A) \cdot \|a_i\|_2.$$

Since $\sqrt{a + b} \leq \sqrt{a} + \sqrt{b}$ this yields

$$\sum_{i \in [n]} \sqrt{u_i} \cdot \|a_i\|_2 \leq 2 \sum_{i \in [n]} \sqrt{\sigma_i(A) \cdot \|a_i\|_2} + \frac{1}{\sqrt{n \cdot \kappa(A^T A)}} \sum_{i \in [n]} \|a_i\|_2 \leq 3 \sum_{i \in [n]} \sqrt{\sigma_i(A) \cdot \|a_i\|_2}$$

which in turn yields the result as $\tilde{O}$ hides factors poly-logarithmic in $n$ and $d$. □
3.3 Regression Algorithm Without Leverage Score Estimates

In the previous section we showed that we can solve regression in our desired running time provided we have constant factor approximation to leverage scores. Here we show how to apply this procedure repeatedly to estimate leverage scores as well. We do this by first adding a large multiple of the identity to our matrix and then gradually decreasing this multiple while maintaining estimates for leverage scores along the way. This is a technique introduced in [94] and we leverage it tailored to our setting.

A key technical ingredient for this algorithm is the following well-known result on the reduction from leverage score computation to regression with little overhead. Formally, Lemma 3.3.1 states that you can compute constant multiplicative approximations to all leverage scores of a matrix in nearly linear time plus the time needed to solve $O(1)$ regression problems. Algorithm 2 details the procedure for computing leverage scores.

**Algorithm 2: ComputeLS($A$, $\delta$, $A$)**

Let $k = c \log(n)$ and $\varepsilon = \frac{\delta^2}{(18nd \log n \cdot \kappa(A^\top A))^\gamma}$ where $c$ is some large enough constant.

for $j = 1, \ldots, k$ do

Let $v_j \in \mathbb{R}^n$ be a random Gaussian vector, i.e. each entry follows $N(0, I)$.

Use algorithm $\mathcal{A}$ to find a vector $y_j$ such that

$$f_{A, v_j}(y_j) - \min_x f_{A, v_j}(x) \leq \varepsilon(f_{A, v_j}(0) - \min_x f_{A, v_j}(x)) .$$

end

Let $\tau_i = \frac{1}{k} \sum_{j=1}^k (e_i^\top A^\top y_j)^2$ for all $i = 1, \ldots, n$.

Output: $\frac{\tau_i}{1 - \delta/3} + \frac{\delta}{2n \cdot \kappa(A^\top A)}$.

**Lemma 3.3.1 (Computing Leverage Scores).** For $A \in \mathbb{R}^{n \times d}$, let ALG be a $T(A)$-time algorithm for regression on $A$. For $\delta \in (\frac{1}{n}, \frac{1}{2})$, in time $O((\text{nnz}(A) + T(A) \log \varepsilon^{-1})\delta^{-2} \log n)$ where we set $\varepsilon = \delta^2(18n \cdot d \cdot \log n \cdot \kappa(A^\top A))^{-2}$, with high probability in $n$, the algorithm
ComputeLS \((A, \delta, A)\) outputs \(\tau \in \mathbb{R}^n\) such that \(\sigma_i(A) \leq \tau_i \leq (1 + \delta)\sigma_i(A) + \delta \cdot [n \cdot \kappa(A^\top A)]^{-1}\) for all \(i \in [n]\).

We provide the proof of Lemma 3.3.1 at the end of this section. Combining the algorithm for estimating leverage scores ComputeLS (Algorithm 2) with our regression algorithm given leverage scores SolveUsingLS (Theorem 3.2.4) yields our solver (Algorithm 3). We first provide a technical lemma regarding invariants maintained by the algorithm (Lemma 3.3.2). The proof of Lemma 3.3.2 is provided at the end of the chapter.

**Algorithm 3: Solve\(_A(x_0, b, \varepsilon)\)**

1. Let \(A_\eta = \left( \begin{array}{c} A \\ \sqrt{\eta} I \end{array} \right), \eta = \lambda_{\text{max}}(A^\top A)\) and \(u_i = \begin{cases} \frac{1}{\eta} ||a_i||^2 & \text{if } 1 \leq i \leq n \\ 1 & \text{if } n + 1 \leq i \leq n + d \end{cases}\)
2. repeat
   1. \(u \leftarrow 2 \cdot \text{ComputeLS}(A_\eta, \frac{1}{4}, A)\) for algorithm \(A\) given by SolveUsingLS\(_{A_\eta,u}\).
   2. \(\eta \leftarrow \frac{3}{4} \cdot \eta\).
3. until \(\eta > \frac{1}{10} \lambda_{\text{min}}(A^\top A)\);
4. Set \(\eta \leftarrow 0.\) Let \(\bar{b} = \left( \begin{array}{c} b \\ \vec{0} \end{array} \right) \in \mathbb{R}^{n+d}.
5. Apply algorithm SolveUsingLS\(_{A_0,u}\) to find \(y\) such that
   \[ f_{A_0,b}(y) - \min_x f_{A_0,b}(x) \leq \varepsilon (f_{A_0,b}(x_0) - \min_x f_{A_0,b}(x)). \]
6. Output: \(y\)

**Lemma 3.3.2.** In the algorithm Solve\(_{A, \varepsilon}\) (See Algorithm 3) the following invariant is satisfied

\[ \sigma_i(A_\eta) \leq u_i \leq 4 \cdot \sigma_i(A_\eta) + [n \cdot \kappa(A_\eta^\top A_\eta)]^{-1}. \] (3.3.1)

We now prove Theorem 3.1.5 using Lemma 3.3.2 and Algorithm 3.

**Proof of Theorem 3.1.5** Lemma 3.3.2 shows that \(u\) is always a good enough estimate of \(\sigma_i(A_\eta)\) throughout the algorithm to invoke SolveUsingLS with Theorem 3.2.4. In particular, this holds at the last step when \(\eta\) is set to 0 and thus the output of the algorithm is as desired by Theorem 3.1.5.
During the whole algorithm, ComputeLS($A_{\eta, \frac{1}{4}, A}$) is called $\Theta(\log(n(A^T A)))$ times. Each time ComputeLS is called, SolveUsingLS is called $\Theta(\log(n))$ many times. All that remains is to bound the running time of SolveUsingLS. However, for $\lambda \geq 0$ and $i \in [n]$ we have $\sigma_i(A_A) \leq \sigma_i(A_0)$ and since $A_A^T A_A \geq \lambda I$ we $\lambda \leq \lambda_{\min}(A_A^T A_A)$. Furthermore, since $\lambda_{\min}(A_A^T A_A) \geq \lambda_{\min}(A^T A)$ we have that the running time follows from the following:

$$\frac{\sum_{i \in [n+d]} \sqrt{\sigma_i(A_A^T)} \cdot \|a_i\|_2}{\sqrt{\lambda_{\min}(A_A^T A_A)}} \leq \sum_{i \in [n]} \frac{\sqrt{\sigma_i(A_A^T)} \cdot \|a_i\|_2}{\sqrt{\lambda_{\min}(A^T A)}} + \sum_{i \in [d]} \frac{\sqrt{\lambda}}{\sqrt{\lambda}}.$$

We finish this section by providing the proofs for Lemmas 3.3.1 and 3.3.2.

**Proof of Lemma 3.3.1** Let $y_j^* = (A^T A)^{-1} A^T v_j$ be the minimizer of $f_{A,v_j}(x)$. (3.1.2) shows that

$$\|A y_j - A y_j^*\|_2^2 \leq \varepsilon \cdot v_j^T A (A^T A)^{-1} A^T v_j.$$

Using $v_j \sim N(0, I)$, we have that

$$v_j^T A (A^T A)^{-1} A^T v_j \leq 2d \cdot \log(n)$$

with probability $1 - n^{-\Theta(1)}$. Hence, we have that

$$|e_i^T A y_j - e_i^T A y_j^*| \leq \|A y_j - A y_j^*\|_2 \leq \sqrt{2d \cdot \log(n)}.$$

Using this and

$$|e_i^T A y_j^*| \leq \sqrt{e_i^T A (A^T A)^{-1} A^T e_i} \sqrt{v_j^T A (A^T A)^{-1} A^T v_j} \leq \sqrt{2d \cdot \log(n)},$$

we have that

$$\left| (e_i^T A y_j)^2 - (e_i^T A y_j^*)^2 \right| \leq 6\sqrt{d \cdot \log(n)}.$$
Using the definition of $\varepsilon$, we have that

$$\left| \frac{1}{k} \sum_{j=1}^{k} (e_i^T A y_j)^2 - \frac{1}{k} \sum_{j=1}^{k} (e_i^T A y_j^*)^2 \right| \leq 6\sqrt{\varepsilon d} \cdot \log(n) \leq \frac{3}{3n \cdot \kappa(A^T A)} \tag{3.3.2}$$

Also, we note that

$$\frac{1}{k} \sum_{j=1}^{k} (e_i^T A y_j^*)^2 = \frac{1}{k} \sum_{j=1}^{k} (e_i^T A (A^T A)^{-1} A^T v_j)^2.$$ 

Since $v_j \sim N(0, I)$ and $k = c \log(n)/\delta^2$ where $c$ is some large enough constant, Johnson-Lindenstrauss lemma shows that, with high probability in $n$ for all $i \in [n]$

$$\left( 1 - \frac{\delta}{3} \right) \sigma_i(A) \leq \frac{1}{k} \sum_{j=1}^{k} (e_i^T A y_j^*)^2 \leq \left( 1 + \frac{\delta}{3} \right) \sigma_i(A)$$

Combining this with (3.3.2) gives the result.

Finally, to check the success probability of this algorithm, we note that we solved $O(\delta^{-2} \log n)$ many regression problems and each one has success probability $1 - n^{-\Theta(1)}$. Also, the Johnson–Lindenstrauss lemma succeed with probability $1 - n^{-\Theta(1)}$. This gives the result. \qed

**Proof of Lemma 3.3.2** Note that $A_n^T A_n = A^T A + \eta I$. Consequently, since initially $\eta = \lambda_{\text{max}}(A^T A)$ we have that initially $\eta I \preceq A_n^T A_n \preceq 2\eta I$. Consequently, we have that initially $\sigma_i(A_n) \leq u_i \leq 2\sigma_i(A_n)$ and therefore satisfies the invariant (3.3.1).

Now, suppose at the start of the repeat loop, $u$ satisfies the invariant (3.3.1). In this case the the assumptions needed to invoke SolveUsingLS by Theorem 3.2.4 are satisfied. Hence, after the line $u \leftarrow 2 \cdot \text{ComputeLS}(A_n, \frac{1}{4}, A)$, by Lemma 3.3.1 we have that for all $i \in [n]$

$$2\sigma_i(A_n) \leq u_i \leq 2 \left( 1 + \frac{1}{4} \right) \sigma_i(A_n) + \frac{2}{4n \cdot \kappa(A_n^T A_n)}.$$
Now, letting $\eta' = \frac{3}{4} \eta$ we see that $(3/4)\sigma_i(A_{\eta}) \leq \sigma_i(A_{\eta'}) \leq (4/3)\sigma_i(A_{\eta})$ and direct calculation shows that invariant (3.3.1) is still satisfied after changing $\eta$ to $\eta'$.

All the remains is to consider the last step when we set $\eta = 0$. When this happens $\eta < \frac{1}{10} \lambda_{\min}(A^T A)$. and therefore $\sigma_i(A_{\eta})$ is close enough to $\sigma_i(A)$ and the invariant (3.3.1) is satisfied.

### 3.4 Extension for GLMs (under Assumption A3.1.7)

In this section we consider the GLM problem (cf. Definition 3.1.6) under A3.1.7. We propose Algorithm 4 as the main sub-routine to solve the GLM problem. Theorem 3.4.2 provides the error guarantee and bounds the running time of Algorithm 4. Note that Theorem 3.4.2 provides a constant factor decrease in the error which can be repeated via a standard reduction to provide $\varepsilon$ error as required by Theorem 3.1.8. We formally provide the reduction and the proof of Theorem 3.1.8 and then provide the proof of Theorem 3.4.2 in Section 3.4.1.

Before describing the algorithm, we first provide a generalization of Theorem 3.1.4 for the GLM setting.

**Theorem 3.4.1 (Acc. Coordinate Descent for ERM).** Consider the GLM problem (cf. Definition 3.1.6) with $\psi_i$ such that $\forall \ x \ \psi_i''(x) \in [\mu_i, L_i]$ and $\lambda$ such that $\forall \ x \ \nabla^2 F(x) \geq \lambda I$. Given a point $x_0$, there exists an algorithm $A$ which produces a point $x'$ w.h.p in $n$ such that

$$F(x') - \min_{x \in \mathbb{R}^d} F(x^*) \leq \varepsilon (F(x_0) - \min_{x \in \mathbb{R}^d} F(x^*))$$

in total time proportional to

$$\tilde{O}
\left(
\left(
\sum_{i=1}^{n} \sqrt{\frac{L_i}{\mu_i}} + \sum_{i=1}^{n} \frac{\|a_i\|_2}{\lambda} \sqrt{\frac{L_i}{\lambda}}
\right) s(A) \log(\varepsilon^{-1})
\right)$$

60
The proof of Theorem 3.4.1 is a direct consequence of [18] and is deferred to Section 3.6.2. We will use Algorithm \( \mathcal{A} \) guaranteed by Theorem 3.2.1 as a subroutine in the Algorithm 4 which we describe next.

Algorithm 4 takes as input, estimates of leverage scores of the matrix \( \mathbf{A} \mathbf{A}^\top \) and creates an estimator of the true function by sampling component functions according to the probability distribution given by the leverage scores and appropriate re-scaling. Further, it reformulates the estimator as a sum of variance reduced components akin to [84]. The algorithm then approximately minimizes the estimator using an off-the-shelf ERM minimizer \( \mathcal{A} \) (Theorem 3.4.1). This step can be seen as analogous to the preconditioned iteration in the case of linear regression.

**Algorithm 4: ERMSolve**

\[
\text{Define for } k = 1 \rightarrow n, p_k = \frac{1}{\sum_j \tau_j}.
\]

Let \( \mathcal{D}(j) \) be the distribution over \([1, \ldots, n]\) such that \( \forall k \), \( \Pr_{j \sim \mathcal{D}(j) = k} = p_k \)

Define for \( k = 1 \rightarrow n, \tilde{f}_k(x) = \frac{1}{p_k} \left[ f_k(x) - \nabla f_k(x) \right] + \nabla F(x) \)

Sample \( m \) integers \( i_1 \ldots i_m \in [n] \) independently from \( \mathcal{D} \).

\[
\text{if } \sum_{t=1}^m \frac{|a_t|^2}{\sqrt{n_t}} \leq 10m \sum_{k=1}^n \| a_k \| 2 \sqrt{p_k} \text{ then}
\]

Set \( F_m(x) = \frac{1}{m} \sum_{t=1}^m \tilde{f}_{i_t}(x) \).

Use Theorem 3.4.1 to find \( x' \) such that

\[
F_m(x') - \min F_m(x) \leq \frac{1}{512L_3^3} \left( F_m(x_0) - \min F_m(x) \right)
\]

Output: \( x' \)

The following is the main theorem we prove regarding Algorithm 4. The proof is provided in Section 3.4.1.

**Theorem 3.4.2.** Given a GLM function \( F(x) \) (Definition 3.1.6) satisfying A3.1.7 and numbers \( u_i \) which are over estimates of leverage scores i.e. \( u_i \geq \sigma_i \), set parameters such that

\[
\tau_i = \min \{1, 20u_i \log(d)\}, m = 160 \left( \sum_j \tau_j \cdot L_4^4 \right)
\]

then we have that Algorithm 4 produces
a point \( x' \) such that

\[
F(x') - \min_{x \in \mathbb{R}^d} F(x) \leq \frac{1}{2} \left( F(x_0) - \min_{x \in \mathbb{R}^d} F(x) \right)
\]

with probability at least \( 1/2 \). Further Algorithm 4 can be implemented in total time

\[
\tilde{O} \left( \left( mL_3 + \sum_{i=1}^{n} \frac{\|a_i\|_2 \sqrt{n}L_3^3}{\sqrt{\lambda_{\min}(A^\top A)}} \right) s(A) \right).
\]

Using the above theorem we now provide the proof of Theorem 3.1.8. Before providing the proof, we provide Lemma 3.4.3 which reduces the problem of achieving \( \varepsilon \) accuracy with high probability to the problem of achieving an accuracy \( c \) with probability at least \( \delta \) for some constants \( c, \delta \). Note that a naive reduction suffers an additional \( \log \log(1/\varepsilon) \) term which we avoid. The reduction helps provide a concise proof of Theorem 3.1.8 based on Theorem 3.4.2.

**Lemma 3.4.3.** Consider being given a function \( F : \mathbb{R}^d \rightarrow \mathbb{R} \) and define \( x^* = \arg\min_x F(x) \). Let ALG be an algorithm such that given any point \( x_0 \) the algorithm runs in time \( T \) and produces a point \( x' \) such that

\[
F(x') - F(x^*) \leq c (F(x_0) - F(x^*))
\]

with probability at least \( 1 - \delta \) for given universal constants \( c, \delta \in [0, 1] \). Further suppose there exists a procedure \( \mathcal{P} \) which given a point \( x \) can produce an estimate \( m \) in time \( T' \) such that \( F(x) - F(x^*) \in [m/r, rm] \) for some given \( r \geq 1 \). Then there exists a procedure that given a point \( x_0 \) outputs a point \( x' \) such that

\[
F(x') - F(x^*) \leq \varepsilon (F(x_0) - F(x^*))
\]
and the expected running time of the procedure is bounded by $O((\mathcal{T} + \mathcal{T}') \log(r) \log(\varepsilon^{-1}))$
where $O()$ hides constant factors in $c, \delta$. Moreover for any $\gamma$ we have a procedure that
produces a point $x'$ such that

$$F(x') - F(x^*) \leq \varepsilon (F(x_0) - F(x^*))$$

with probability at least $1 - \gamma$ with a total running time of

$$O \left((\mathcal{T} + \mathcal{T}') \log(r) \log(\varepsilon^{-1}) \log(\gamma^{-1})\right)$$

We first use Lemma 3.4.3 to prove Theorem 3.1.8 and provide the proof of Lemma 3.4.3
in Section 3.6.3.

Proof of Theorem 3.1.8. We make use of Lemma 3.4.3 plugging in Algorithm 4 as the
procedure ALG. Note that $c, \delta$ are both $1/2$ as guaranteed by Theorem 3.4.2. Moreover
since $F(x)$ is such that that $\forall \ x \ L_3\lambda_{\min}(A^\top A) \leq \nabla^2F(x) \leq L_3\lambda_{\max}(A^\top A)$, we can
use $\|\nabla F(x)\|_2^2$ as an estimator for $F(x) - F(x^*)$. The corresponding $r$ for it is bounded by
$L_3^2\kappa(A^\top A)$.

Finally note that the running time guaranteed in Theorem 3.4.2 depends on the quality
of the estimates of the leverage scores input to it. We invoke Lemma 3.3.1 for computing
accurate estimates of leverage scores. Putting together the above arguments finishes the
proof for Theorem 3.1.8.

3.4.1 Proof of Theorem 3.4.2

Proof. For convenience we restate the definitions provided in Algorithm 4. Given parameters
$\{\tau_1 \ldots \tau_n\}$ we define a probability distribution $\mathcal{D}$ over $\{1, \ldots n\}$ such that

$$\forall \ k \in [n] \quad p_k \triangleq Pr_{j \sim \mathcal{D}}(j = k) \triangleq \frac{\tau_k}{\sum \tau_k}.$$  \hspace{1cm} (3.4.1)
We define approximations to \( f_k \) for \( k \in [n] \) as

\[
\tilde{f}_k(x) \triangleq \frac{1}{p_k} \left[ f_k(x) - \nabla f_k(x)_{\mathbf{x}}^T x \right] + \nabla F(x_0)^T x.
\] (3.4.2)

Further we sample \( m \) integers \( \{i_1, \ldots, i_m\} \) independently from \( D \) and we define the approximation

\[
F_m(x) \triangleq \frac{1}{m} \sum_{t=1}^m \tilde{f}_{i_t}(x).
\] (3.4.3)

Define \( x^* \triangleq \text{argmin}_x F(x) \). To prove the theorem we will prove two key properties. Firstly the choice of the sample size \( m = \Omega(\sum_{k=1}^n \tau_k L^4_d) \) is sufficient to ensure that approximately minimizing \( F_m(x) \) makes constant multiplicative factor progress on \( F \). Secondly we will bound the running time of the coordinate descent procedure (\( A \) from Theorem 3.2.1).

Consider the random matrix

\[
\tilde{A}^\top \tilde{A} \triangleq \frac{1}{m} \sum_{t=1}^m a_{i_t} a_{i_t}^\top / p_{i_t}.
\] (3.4.4)

and define the event \( \mathcal{E}_1 \) to be the following event.

\[
\mathcal{E}_1 \triangleq \left\{ 0.5 \mathbf{A}^\top \mathbf{A} \preceq \tilde{\mathbf{A}}^\top \tilde{\mathbf{A}} \preceq 2 \mathbf{A}^\top \mathbf{A} \right\}.
\] (3.4.5)

We use the following concentration inequality Lemma 3.4.4 (proved in Section 3.6.4).

**Lemma 3.4.4.** Given an error parameter \( 0 \leq \varepsilon \leq 1 \), let \( u \) be a vector of leverage score overestimates, i.e. \( \sigma_i(\mathbf{A}) \leq u_i \) for all \( i \). Let \( \alpha = \varepsilon^{-2} \) be a sampling rate parameter and \( c \) be a fixed constant. For each row we define a number \( \gamma_i = \min\{1, \alpha u_i \log(d)\} \) and a probability \( p_i = \frac{\gamma_i}{\sum_i \gamma_i} \). Let \( Y_j \) be a random variable which is sampled by picking a vector \( a_{i_j} \) with probability \( p_i \) and setting \( Y_j = \frac{a_{i_j} a_{i_j}^\top}{p_i} \). Now consider the random variable \( Y = \frac{1}{m} \sum_j Y_j \).

We have that as long as \( m \geq \sum_i \gamma_i \) then

\[
\Pr((1 - \varepsilon) \mathbf{A}^\top \mathbf{A} \preceq Y \preceq (1 + \varepsilon) \mathbf{A}^\top \mathbf{A}) \geq 1 - d^{-c/3}
\]
The above lemma ensures

\[ Pr(\mathcal{E}_1) \geq 1 - 1/d \]

The following lemma bounds the number of samples required for exact minimization of \( F_m(x) \) to lead to constant decrease in error under the event \( \mathcal{E}_1 \). We provide the proof of Lemma 3.4.5 after finishing the present proof.

**Lemma 3.4.5.** Consider a GLM function \( F(x) = \sum f_i(x) \) as defined in Definition 3.1.6 and satisfying A3.1.7. Let \( F_m \) be as defined in (3.4.3) and \( \tilde{\mathbf{A}} \) be as defined in (3.4.4). Let

\[ x_m = \arg\min_{x \in \mathbb{R}^d} F_m(x). \]

Let \( \mathcal{E}_1 = \{ 0.5\mathbf{A}^\top \mathbf{A} \leq \tilde{\mathbf{A}}^\top \tilde{\mathbf{A}} \leq 2\mathbf{A}^\top \mathbf{A} \} \) and let \( Pr(\mathcal{E}_1) \geq p \). Then if we set \( m \geq 160(\sum_j \tau_j) \cdot L_3^4 \), we have that

\[ Pr \left( F(x_m) - F(x^*) \leq O \left( \frac{1}{4} \left( F(x_0) - F(x^*) \right) \right) \right) \geq p - \frac{1}{10}. \tag{3.4.6} \]

For the rest of the proof we will assume that the event \( \mathcal{E}_1 \) and the property (3.4.6) holds. Lemma 3.4.5 implies that this happens with probability at least 7/10. An application of Markov’s inequality gives us that the condition in the if statement in Algorithm 4 i.e.

\[ \sum_{t=1}^{m} \left\| a_t \right\|_2 \leq 10m \sum_{k=1}^{n} \left\| a_k \right\|_2 \sqrt{p_k} = 10E \left[ \sum_{t=1}^{m} \frac{\left\| a_t \right\|_2}{\sqrt{p_t}} \right] \] \tag{3.4.7} \]

happens with probability at least 9/10. Putting the above together via a union bound gives us that with probability at least 6/10 all three of the following happen: \( \mathcal{E}_1 \), Condition (3.4.6) and the execution of the if loop (i.e. Condition 3.4.7 is met). We now show that under the above conditions we get sufficient decrease in error. Firstly note that by definition we have that

\[ F_m(x') - F_m(x_m) \leq \frac{1}{512L_3^4} \left( F_m(x_0) - F_m(x_m) \right). \tag{3.4.8} \]
Note that if event $E_1$ happens then

$$\forall \ x \quad \frac{1}{2L_3} A^\top A \leq \nabla^2 F_m(x) \leq 2L_3 A^\top A \quad .$$  \hspace{1cm} (3.4.9)

Now consider the RHS of (3.4.8)

$$F_m(x_0) - F_m(x_m) \leq L_3 \|x_0 - x_m\|_{A^\top A}^2 \leq 2L_3(\|x_0 - x^*\|_{A^\top A}^2 + \|x_m - x^*\|_{A^\top A}^2)$$

$$\leq 4L_3^2(F(x_0) - F(x^*) + F(x_m) - F(x^*))$$

$$\leq 5L_3^2(F(x_0) - F(x^*))$$  \hspace{1cm} (3.4.10)

The first inequality follows from (3.4.9), second from triangle inequality, third by noting that $F$ is $\frac{1}{L_3}$ strongly convex in $A^\top A$ norm (Assumption A3.1.7) and the fourth from Lemma 3.4.5. Further,

$$F(x') - F(x_m) \leq \nabla F(x_m)^\top (x' - x_m) + \frac{L_3}{2}\|x' - x_m\|_{A^\top A}^2$$

$$\leq \|\nabla F(x_m)\|_{[A^\top A]^{-1}}\|(x' - x_m)\|_{A^\top A} + \frac{L_3}{2}\|x' - x_m\|_{A^\top A}^2$$

$$\leq \sqrt{2L_3(F(x_m) - F(x^*))}\|(x' - x_m)\|_{A^\top A} + \frac{L_3}{2}\|x' - x_m\|_{A^\top A}^2$$

$$\leq \sqrt{2L_3(F(x_m) - F(x^*))}\sqrt{4L_3(F_m(x') - F_m(x_m))}$$

$$+ 2L_3^2(F_m(x') - F_m(x_m))$$

$$\leq \frac{1}{8L_3}\sqrt{(F(x_m) - F(x^*))}\sqrt{(F_m(x_0) - F_m(x_m))}$$

$$+ \frac{1}{256L_3^2}(F_m(x_0) - F_m(x_m))$$

$$\leq \frac{1}{3}(F(x_0) - F(x^*))$$  \hspace{1cm} (3.4.11)

The first and third inequality follow by noting that $F$ is $L_3$ smooth and $1/L_3$ strongly convex in $A^\top A$ norm. Fourth inequality follows by noting that if event $E_1$ holds, $F_m$ is $1/2L_3$ strongly convex in $A^\top A$ norm. Fifth inequality follows from (3.4.8) and sixth inequality
from (3.4.10) and Lemma 3.4.5 (3.4.11) together with (3.4.6) implies that with probability at least \( \frac{6}{10} \), we have that

\[
F(x') - F(x^*) \leq \frac{1}{2}(F(x_0) - F(x^*))
\]

We will now bound the running time of the procedure via Theorem 3.4.1. Define \( L_{i_t} \) and \( \mu_{i_t} \) to be respectively the smoothness and strong convexity parameters of the components \( \tilde{f}_{i_t} \). Note that \( L_{i_t} \leq \frac{L_3}{m p_{i_t}} \) and \( \mu_{i_t} \geq \frac{1}{L_3^2 m^2 p_{i_t}} \). Note that event \( E_1 \) gives us that \( \forall x \ \nabla^2 F_m(x) \geq \frac{1}{2L_3} \lambda_{\min}(A^\top A) \). A direct application of Theorem 3.4.1 using the bounds on \( L_{i_t} \) and \( \mu_{i_t} \) gives us that the total running time is bounded by

\[
\tilde{O} \left( \left( \sum_{i=1}^{m} L_{i_t} + \sum_{i=1}^{m} a_i 2 \sqrt{\frac{L_3}{mp_{i_t}}} \cdot \frac{L_3}{\lambda_{\min}(A^\top A)} \right) s(A) \log(\varepsilon^{-1}) \right)
\]

\[
\leq \tilde{O} \left( mL_3 + \sum_{i=1}^{n} \frac{\|a_i\|_2 \sqrt{\tau_i L_3^2}}{\sqrt{\lambda_{\min}(A^\top A)}} \right)
\]

The inequality follows from Condition (3.4.7) and the definitions of \( p_k, m \).

Proofs of ERM Sampling (Lemma 3.4.5)

**Proof of Lemma 3.4.5** Consider the definitions in (3.4.1), (3.4.2), (3.4.3). Note the following easy observation.

\[
F(x) = \mathbb{E}_{k \sim D} \tilde{f}_k(x)
\]

Consider the following Lemma 3.4.6 which connects the optima of two convex functions \( F \) and \( G \).

**Lemma 3.4.6.** Let \( F(x), G(y) : \mathbb{R}^d \rightarrow \mathbb{R} \) be twice differentiable and strictly convex. Define

\[
x^* = \arg \min_x F(x) \text{ and } y^* = \arg \min_y G(y)
\]
Then we have that
\[ F(y^*) - F(x^*) = \| \nabla G(x^*) \|^2 \mathbb{H}_G^{-1} \mathbb{H}_F \mathbb{H}_G^{-1} \cdot \]

where \( H_F \triangleq \int_0^1 \nabla^2 F(t.y^* + (1-t)x^*) dt \) and \( H_G \triangleq \int_0^1 \nabla^2 G(t.y^* + (1-t)x^*) dt \).

We wish to invoke Lemma 3.4.6 by setting \( F = F(x), G = F_m(x) \). In this setting we have that
\[ H_F \triangleq \int_0^1 \nabla^2 F(t.x_m + (1-t)x^*) dt \quad \text{and} \quad H_G \triangleq \int_0^1 \nabla^2 F_m(t.x_m + (1-t)x^*) dt \]

Firstly note that the definition of \( F \) and A3.1.7 gives us that
\[ H_F \leq L_3 \cdot A^\top A \tag{3.4.12} \]

Using Definition 3.4.4 and A3.1.7 gives us that
\[ H_G \leq L_3 \cdot \tilde{A}^\top \tilde{A} \]

Combining the above two and noting that the event \( E_1 \) happens with probability at least \( p \) we get that
\[ H_G^{-1} A^\top A H_G^{-1} \leq 4L_3^2[A^\top A]^{-1} \text{w.p. } p \tag{3.4.13} \]

Also note that for any fixed matrix \( R \), we have that
\[ \mathbb{E}[\| \nabla F_m(x^*) \|^2_R] = \frac{\mathbb{E}_{k \sim \mathcal{D}}[\| \nabla f_{\tilde{k}}(x^*) \|^2_R]}{m} \]

which implies via Markov’s inequality that with probability at least \( 9/10 \) we have that
\[ \| \nabla F_m(x^*) \|^2_R \leq \frac{10\mathbb{E}_{k \sim \mathcal{D}}[\| \nabla f_{\tilde{k}}(x^*) \|^2_R]}{m} \tag{3.4.14} \]
Putting (3.4.13) and (3.4.14) together and using a union bound we get that

\[
\|F_m(x^*)\|_{H^{-1}_G A H^{-1}_C}^2 \leq \frac{40L_3^2 \mathbb{E}_{k \sim D}[\|\nabla \tilde{f}_k(x^*)\|_{[A^\top A]^{-1}}^2]}{m} \text{ w.p. } p - \frac{1}{10}
\]

Using Lemma 3.4.6 and (3.4.12) we get that with probability at least \( p - \frac{1}{10} \)

\[
F(x_m) - F(x^*) \leq \frac{40L_3^2 \mathbb{E}_{k \sim D}[\|\nabla \tilde{f}_k(x^*)\|_{[A^\top A]^{-1}}^2]}{m} \tag{3.4.15}
\]

We will now connect \( \mathbb{E}_{k \sim D}[\|\nabla \tilde{f}_k(x^*)\|_{[A^\top A]^{-1}}^2] \) with the error at \( x_i \).

**Lemma 3.4.7.** Consider an ERM function \( F(x) = \sum_{i=1}^{m} f_i(x) \) where \( f_i(x) = \psi_i(a_i^\top x) \) with \( \psi_i \in [\frac{1}{L_3}, L_3] \). Define a distribution \( D(j) \) over \([n]\) such that \( Pr(j = k) = p_k = \frac{\tau_k}{\sum \tau_k} \) for numbers \( \tau_k = \min(1, 20 u_k \log(d)) \) where \( u_k \geq \sigma_i[A^6] \) are overestimates of leverage scores.

Given a point \( \bar{x} \) consider the variance reduced reformulation

\[
F(x) = \mathbb{E}_{k \sim D}[\tilde{f}_k(x)]
\]

where

\[
\tilde{f}_k(x) = \frac{1}{p_k} \left[ f_k(x) - \nabla f_k(\bar{x})^\top x \right] + \nabla F(\bar{x})^\top x
\]

Then we have that

\[
\mathbb{E}_{k \sim D} \left[ \|\nabla \tilde{f}_k(x^*)\|_{[A^\top A]^{-1}}^2 \right] \leq 2 \left( \sum_j \tau_j \right) \cdot L_3 \cdot \left( F(\bar{x}) - F(x^*) \right)
\]

Putting together (3.4.15) and Lemma 3.4.7 we get that

\[
F(x_m) - F(x^*) \leq 80 \left( \frac{\left( \sum_j \tau_j \right) \cdot L_3^4}{m} \cdot (F(x_0) - F(x^*)) \right) \text{ w.p. } p - \frac{1}{10}
\]

Lemma 3.4.5 now follows from the choice of \( m \).

\( ^6 \sigma_i \) are leverage scores defined in Definition 3.2.2
We finish this section with proofs of Lemma 3.4.6 and 3.4.7.

**Proof of Lemma 3.4.6.** For all \( t \in [0, 1] \) let \( z(t) = t \cdot y^* + (1 - t) \cdot x^* \) for \( t \in [0, 1] \) and \( H_F = \int_0^1 \nabla^2 F(z(t)) dt \). By Taylor series expansion we have that

\[
F(y^*) = F(x^*) + \nabla F(x^*)^\top(y^* - x^*) + \int_0^1 \frac{1}{2} (y^* - x^*)^\top \nabla^2 F(z(t))(y^* - x^*) dt
\]

\[
= F(x^*) + \frac{1}{2} \|y^* - x^*\|_{H_F}^2.
\]

Here we used that \( \nabla F(x^*) = 0 \) and \( \nabla^2 F(z(t)) \geq 0 \) by the convexity of \( F \). We also have by definition that

\[
\nabla G(y^*) = \vec{0}
\]

and therefore

\[
\nabla G(y^*) - \nabla G(x^*) = \int_0^1 \nabla^2 G(z(t))(y^* - x^*) dt
\]

and

\[
(y^* - x^*) = -H_G^{-1} \nabla G(x^*)
\]

where \( H_G = \int_0^1 \nabla^2 G(z(t)) \). We now have that

\[
F(y^*) - F(x^*) = \frac{1}{2} \|y^* - x^*\|_{H_F}^2 = \|\nabla G(x^*)\|_{H_G^{-1} \cdot H_F \cdot H_G^{-1}}^2
\]  

(3.4.16)

\[\square\]

**Proof of Lemma 3.4.7.** For the purpose of this proof it will be convenient to perform a change of basis. Define the function

\[
G(x) = \mathbb{E}_{k \sim \mathcal{D}} g_i(x) \quad \text{where} \quad g_i(x) = \frac{1}{p_i} f_k((A^\top A)^{-1/2} x)
\]
Note that $G(x) = F((A^\top A)^{-1/2}x)$. We will first note that

$$\nabla^2 g_i(x) = \frac{1}{p_i} \cdot \left[(A^\top A)^{-1/2} a_i (A^\top A)^{-1/2} \cdot \psi''(a_i (A^\top A)^{-1/2} x)\right]$$

and now by the cyclic property of trace and the fact that $\psi'' \leq L_3$ we have

$$\tau(\nabla^2 g_i(x)) = \frac{\sum_j \tau_j}{\tau_i} \cdot a_i (A^\top A)^{-1} a_i \cdot L_3$$

Note that $a_i (A^\top A)^{-1} a_i \leq 1$. Now either $\tau_i = 1$ or $\tau_i \geq 20a_i (A^\top A)^{-1} a_i \log(d)$. In both cases we see that RHS above $\leq 1$. Therefore we get that $g_i$ is $(\sum_j \tau_j)L_3$ smooth. We now have the following lemma.

**Lemma 3.4.8.** Let $\mathcal{D}$ be any distribution over $[n]$ and define $G = \mathbb{E}_{i \sim \mathcal{D}}[g_i(x)]$ for component convex functions $g_i$ each of which is $L$ smooth. Let $x^* \doteq \arg\min G(x)$. We have that

$$\mathbb{E}_{i \sim \mathcal{D}}\|\nabla g_i(x) - \nabla g_i(x^*)\|_2^2 \leq 2L(G(x) - G(x^*))$$

The proof of the above Lemma is identical to the proof of Equation 8 in [84] and we provide the proof for completeness. Assuming the Lemma, note that

$$2(\sum_j \tau_j)L_3(F(\bar{x}) - F(x^*)) = 2(\sum_j \tau_j) \cdot L_3 \cdot (G((A^\top A)\bar{x}) - G((A^\top A)x^*))$$

$$\geq \mathbb{E}_{i \sim \mathcal{D}}\|\nabla g_i((A^\top A)\bar{x}) - \nabla g_i((A^\top A)x^*)\|_2^2$$

$$= \mathbb{E}_{i \sim \mathcal{D}}\|(A^\top A)^{-1/2} \frac{1}{p_i} (\nabla f_i(\bar{x}) - \nabla f_i(x^*))\|_2^2$$

$$= \mathbb{E}_{i \sim \mathcal{D}}\|\nabla \tilde{f}_i(x^*)\|_{(A^\top A)^{-1}}^2 - \|\nabla F(\bar{x})\|_{[A^\top A]^{-1}}^2$$

$$- 2\mathbb{E}_{i \sim \mathcal{D}} \left[ \frac{1}{p_i} (\nabla f_i(x^*) - \nabla f_i(\bar{x}))^\top [A^\top A]^{-1} \nabla F(\bar{x}) \right]$$

$$\geq \mathbb{E}_{i \sim \mathcal{D}}\|\nabla \tilde{f}_i(x^*)\|_{(A^\top A)^{-1}}^2$$

71
The first line follows by definition. The second line by noting that $g$ is $(\sum_j \tau_j) L_3$ smooth. The third and fourth line follows by definition. The fifth line follows by noting that $\nabla F(x^*) = 0$. 

Proof of Lemma 3.4.8. Let $x^* \doteq \text{argmin } g(x)$. Define auxiliary functions

$$h_i(x) \doteq g_i(x) - g_i(x^*) - \nabla g_i(x^*)^\top (x - x^*)$$

We know that $h_i(x^*) = \min h_i(x)$ since $\nabla h_i(x^*) = 0$. Using smoothness of $h$ and that $h_i(x^*) = 0$, we now have that

$$\|\nabla h_i(x)\|_2^2 \leq 2Lh_i(x)$$

A simple substitution gives us that for all $i$

$$\|\nabla g_i(x) - \nabla g_i(x^*)\|_2^2 \leq 2L (g_i(x) - g_i(x^*) - \nabla g_i(x^*)^\top (x - x^*))$$

Taking expectations and using the fact that $g(x^*) = 0$ gives us that

$$\mathbb{E}_{i \sim D} \|\nabla g_i(x) - \nabla g_i(x^*)\|_2^2 \leq 2L(g(x) - g(x^*))$$

3.5 Extensions to GLM (under Assumption 3.1.9)

In this section we provide a proof of Theorem 3.1.10 which follows as a direct corollary of Theorem 2.5.2 and Theorem 3.1.5. We can use the regression algorithm given by Theorem 3.1.5 as the quadratic solver in Algorithm 2. Note that the quadratics $Q_t$ in Algorithm 2 are
of the form
\[
\frac{1}{2} \cdot y^\top \left( \sum_{i=1}^{\text{nnz}(A)} \psi_i''(a_i^\top x) a_i a_i^\top \right) y + b^\top y
\]
Using the regression algorithm guaranteed by Theorem 3.1.5 to solve the above quadratic we immediately get that each such quadratic can be minimized to relative accuracy \( \varepsilon \) in time proportional to
\[
\tilde{O}\left( \text{nnz}(A) + \left( d + \frac{\sum_{i=1}^{\text{nnz}(A)} \sqrt{\psi_i''(a_i^\top x)^2} \cdot \sigma_i(SA)}{\sqrt{\lambda_{\text{min}}((SA)^T(SA))}} \right) \cdot s(A) \right) \cdot \log \left( \frac{1}{\varepsilon} \right),
\]
where \( S \) is an \( n \times n \) diagonal matrix with entries \( S_{ii} = \sqrt{\psi_i''(a_i^\top x)} \). Note that since \( F \) is \( \alpha \) strongly convex we get that \( \lambda_{\text{min}}((SA)^T(SA)) \geq \alpha \). Further using Cauchy-Schwartz we get that
\[
\tilde{O}\left( \text{nnz}(A) + \left( d + \frac{\sum_{i=1}^{\text{nnz}(A)} \sqrt{\psi_i''(a_i^\top x)^2} \cdot \sigma_i(SA)}{\sqrt{\lambda_{\text{min}}((SA)^T(SA))}} \right) \cdot s(A) \right) \leq \tilde{O}\left( \text{nnz}(A) + \left( d + \sqrt{\frac{d \cdot \max_{x \in \mathbb{R}^d} \sum_{i=1}^{\text{nnz}(A)} \psi_i''(a_i^\top x)^2}{\alpha}} \right) s(A) \right).
\]
The proof now follows from Theorem 2.5.2.

### 3.6 Supplementary Results

#### 3.6.1 Previous Best Regression Algorithm

First we give the theorems encapsulating the results we use and then use them to prove Theorem 3.2.1 in the case when \( A = B \). We then prove the case when \( A \neq B \). Theorem 3.6.1 describes the fastest coordinate descent algorithm known by [18]. Theorem 3.6.2 describes the reduction [64] to from regression to coordinate decent via proximal point.

**Theorem 3.6.1** (Corollary of Thm 5.1 of [18]). Let \( f : \mathbb{R}^n \to \mathbb{R} \) be a twice differentiable \( \sigma \)-strongly convex function for \( \mu > 0 \). Further suppose that for all \( x \in \mathbb{R}^n \) and \( i \in [n] \) it is
the case that $\frac{\partial^2}{\partial x_i^2} f(x) \leq L_i$ for $i \in [n]$ and the partial derivative $\frac{\partial}{\partial x_i} f(x)$ can be computed in $O(s)$ time. Then there exists an algorithm which given any $\varepsilon > 0$ finds a $y \in \mathbb{R}^n$ such that

$$f(y) - \min_x f(x) \leq \varepsilon \left( f(x_0) - \min_x f(x) \right).$$

in expected running time $O(s \sum_i \sqrt{L_i/\mu}).$

**Theorem 3.6.2** (Corollary of Thm 4.3 of [64]). Given $A \in \mathbb{R}^{n \times d}$ with rows $a_1, \ldots, a_n$ and $c \in \mathbb{R}^n$. Consider the function $p(x) = \sum_{i=1}^n \phi_i(a_i^T x)$ where $\phi_i$ are convex functions. Suppose that $\lambda I \leq \nabla^2 p(x) \leq \lambda I$ for all $x \in \mathbb{R}^d$. Let $\kappa = L/\lambda$. Let dual problem

$$g_s(y) = \sum_{i=1}^n \phi_i^*(y_i) + \frac{1}{2\lambda} \|A^T y\|_2^2 - s^T A^T y.$$

Suppose that for any $s \in \mathbb{R}^d$, any $y_0 \in \mathbb{R}^n$ and any $0 \leq \varepsilon \leq \frac{1}{2}$, we can compute $y$ in expected running time $T_\varepsilon$ such that

$$g_s(y) - \min_y g_s(y) \leq \varepsilon (g_s(y_0) - \min_y g_s(y)). \tag{3.6.1}$$

Then, for any $x_0$ and any $\varepsilon \in (0, \frac{1}{2})$ we can find $x$ such that

$$p(x) - \min_x p(x) \leq \varepsilon \left( p(x_0) - \min_x p(x) \right)$$

in time $\tilde{O}(T_\varepsilon \log(1/\varepsilon))$ w.h.p. in $n$ where $\delta = \Theta(n^{-2} \kappa^{-4})$ and $\tilde{O}$ includes logarithmic factors in $n, \kappa$.

We note that although the guarantees of Thm 5.1 of [18] and Thm 4.3 of [64] are not stated in the form of Theorems 3.6.1 and 3.6.2. They can be easily converted to the form above by noticing that the expected running time of the procedure in Thm 4.3 of [64] using Theorem 3.6.1 is $\tilde{O}(T_\delta \log(1/\varepsilon))$ which can then be boosted to high probability in $n$ using Lemma 3.4.3. We now give the proof of Theorem 3.2.1.

74
**Proof of Theorem 3.2.1 when \( A = B \).** Let

\[
p(x) = \sum_{i=1}^{n} \phi_i(a_i^T x) \text{ where } \phi_i(x) = \frac{1}{2}(x - b_i)^2.
\]

Then, we have that \( \phi_i^*(y) = \frac{1}{2}y^2 + b_i y \) and hence

\[
g_s(y) = \sum_{i=1}^{n} \phi_i^*(y_i) + \frac{1}{2\lambda} ||A^T y||_2^2 - s^T A^T y = \frac{1}{2}||y||_2^2 + b^T y + \frac{1}{2\lambda} ||A^T y||_2^2 - s^T A^T y.
\]

Note that \( g_s(y) \) is \( \lambda \) strongly convex and

\[
\frac{d^2}{dy^2} g_s(y) = 1 + \frac{1}{\lambda} ||a_i||_2^2 \leq L_i.
\]

Hence, Theorem 3.6.1 finds \( y \) satisfying (3.6.1) in time

\[
O \left( s(A) \cdot \sum_{i=1}^{n} \sqrt{1 + \frac{1}{\lambda} ||a_i||_2^2 \log(\varepsilon^{-1})} \right) = O \left( \left( n + \frac{1}{\sqrt{\lambda}} \sum_{i=1}^{n} ||a_i||_2 \right) \cdot s(A) \cdot \log(\varepsilon^{-1}) \right).
\]

Hence, this shows that the primal can be solved in time

\[
O \left( \left( n + \frac{1}{\sqrt{\lambda}} \sum_{i=1}^{n} ||b_i||_2 \right) \cdot s \cdot \log(n \cdot \kappa) \cdot \log(\kappa \varepsilon^{-1}) \right)
\]

where we used \( A = B \) at the end. \( \square \)

**Proof of Theorem 3.2.1 for the case \( A \neq B \).** The proof involves two steps. First, we show that given any point \( x_0 \), we can find a new point \( x \) that is closer to the minimizer. Then, we bound how many steps it takes. To find \( x \), we consider the function

\[
f_{x_0}(x) = \frac{1}{2} ||Bx - Bx_0||_2^2 + \langle Ax_0 - c, Ax - Ax_0 \rangle.
\]
Let $z$ be the minimizer of $f_{x_0}$ and $x^*$ be the minimizer of $\frac{1}{2}\|Ax - b\|_2^2$. Note that

$$z = x_0 - (B^T B)^{-1} A^T \eta \text{ with } \eta = Ax_0 - b, \text{ and } x^* = (A^T A)^{-1} A^T b.$$ 

Hence, we have that

$$\frac{1}{2}\|Az - Ax^*\|_2^2 = \frac{1}{2}\|A(A^T A)^{-1} A^T \eta - A(B^T B)^{-1} A^T \eta\|_2^2 = \frac{1}{2}\eta^T (A^T A)^{-1} A^T \eta - \eta^T A^T (B^T B)^{-1} A^T \eta + \frac{1}{2}\|A(B^T B)^{-1} A^T \eta\|_2^2.$$ 

Using that $\frac{5}{6}B^T B \leq A^T A \leq \frac{6}{5}B^T B$, we have

$$\frac{1}{2}\|Az - Ax^*\|_2^2 \leq \frac{4}{10}\eta^T (A^T A)^{-1} A^T \eta = \frac{4}{10}\|Ax_0 - Ax^*\|_2^2. \quad \text{(3.6.2)}$$

However, it is difficult to reduce to the case when $A = B$ to minimize the function $f_{x_0}$ due to the extra linear term. To address this issue, we assume $B = [\tilde{B}; \sqrt{\frac{\lambda}{100}} I]$ by appending an extra identity term. Note that this only adds a small matrix $\frac{\lambda}{100} I$ and hence we still have $\frac{5}{6}B^T B \leq A^T A \leq \frac{6}{5}B^T B$ but with a slightly different constant which will not affect the proof for (3.6.2). Due to the extra identity term, $f_{x_0}(x)$ reduces to an expression of the form $\frac{1}{2}\|Bx - d\|_2^2 + C$ for some vector $d$ and constant $C$. We can now apply Theorem 3.2.1 for the case $A = B$ and get an $x$ such that

$$f_{x_0}(x) - f_{x_0}(z) \leq \frac{1}{200} (f_{x_0}(x_0) - f_{x_0}(z)) \cdot \text{.} \quad \text{(3.6.3)}$$

in time

$$O \left( \left( n + \frac{\sum_{i \in [n]} \|b_i\|_2^2}{\sqrt{\lambda_{\min}(B^T B)}} \right) \cdot s(B) \cdot \log(n \kappa) \cdot \log(\kappa) \right).$$

Note that the extra terms in $B$ does not affect the minimum eigenvalue and it increases $\frac{1}{\sqrt{\lambda}} \sum_{i \in [n]} \|b_i\|_2$ by atmost $n.$
Now, using the formula of $z$, the guarantee (3.6.3) can be written as

$$\|Bx - Bz\|_2^2 \leq \frac{1}{200} \|Ax_0 - Ax^*\|_2^2.$$  

Using that $\frac{6}{5}B^TB \preceq A^TA \preceq \frac{6}{5}B^TB$, we have

$$\|Ax - Az\|_2 \leq \frac{1}{10} \|Ax_0 - Ax^*\|_2.$$  

Combining this with (3.6.2), we have that

$$\|Ax - Ax^*\|_2 \leq 0.9 \|Ax_0 - Ax^*\|_2.$$  

Hence, we get closer to $x^*$ by constant factor. Therefore, to achieve (3.1.2), we only need to repeat this process $\log(1/\varepsilon)$ times. Hence, the total running time is

$$O \left( \left( n + \frac{\sum_{i \in [n]} \|b_i\|_2}{\sqrt{\lambda_{\min}(B^TB)}} \right) \cdot s(B) \log^2(n\kappa) \log(\varepsilon^{-1}) \right).$$

\[ \square \]

### 3.6.2 Accelerated Coordinate Descent for GLM

**Proof of Theorem** 3.4.1 To remind the reader

$$f(x) = \sum_{i=1}^{n} \psi_i(a_i^T x) \text{ where } \psi''_i(x) \in [\mu_i, L_i].$$

Following is a well known theorem. For a proof see [85].

**Theorem 3.6.3** (Strong / Smooth Duality). A closed and convex function $f$ is $\beta$-strongly convex with respect to a norm $\| \cdot \|$ if and only if $f^*$ is $\frac{1}{\beta}$-strongly smooth w.r.t the dual norm of $\| \cdot \|$.  

77
A direct application of the above theorem gives us that \( \psi^*(y) \in [\frac{1}{L_i}, \frac{1}{\mu_i}] \). Consider the function

\[
g_s(y) = \sum_{i=1}^{n} \psi^*_i(y_i) + \frac{1}{2\lambda} \|A^\top y\|_2^2 - s^\top A^\top y
\]

Consider the following modified function \( \tilde{g}_s(y) \equiv g_s(Dy) \) where \( D \) is a diagonal matrix with \( D_{ii} = L_i \). We will equivalently minimize the function \( \tilde{g}_s(y_i) \). We now immediately get that the function \( \tilde{g}_s(y) \) is 1 strongly convex. Moreover we have that

\[
\frac{d^2}{dy_i^2} g_s(y) = \frac{L_i}{\mu_i} + \frac{1}{\lambda} \|a_i\|^2 L_i.
\]

Hence, Theorem 3.6.1 finds \( y \) satisfying (3.6.1) in time

\[
O \left( s(A) \cdot \sum_{i \in [n]} \sqrt{\frac{L_i}{\mu_i} + \frac{1}{\lambda} \|a_i\|^2 L_i \log(\varepsilon^{-1})} \right) =
\]

\[
O \left( \left( \sum_{i=1}^{n} \sqrt{\frac{L_i}{\mu_i} + \frac{1}{\sqrt{\lambda}} \sum_{i=1}^{n} \|a_i\| \sqrt{L_i}} \right) s(A) \log(\varepsilon^{-1}) \right)
\]

A direct application of Theorem 3.6.2 gives that the total running time is

\[
O \left( \left( \sum_{i=1}^{n} \sqrt{\frac{L_i}{\mu_i} + \frac{1}{\sqrt{\lambda}} \sum_{i=1}^{n} \|a_i\| \sqrt{L_i}} \right) s(A) \log(n\kappa) \log(\kappa/\varepsilon) \right)
\]

The above equation assumes that the inner iterations of accelerated coordinate descent can be implemented in \( O(s(A)) \). This is easy to see because diagonal scaling is linear in sparsity. Therefore the only bottleneck is computing the gradient of the dual function \( \psi^* \). We can assume that \( \psi \) is explicit and therefore the gradient of \( \psi^* \) is easily computed. \( \square \)
3.6.3 Reduction from High Probability Solvers to Expected Running Times

Proof of Lemma 3.4.3. To show the lemma we will show the existence of a procedure (described in Algorithm 5) which produces a point $x'$ such that

$$F(x') - F(x^*) \leq \frac{1}{2} (F(x_0) - F(x^*))$$  \hspace{1cm} (3.6.4)

with expected running time bounded by $O((T + T') \log(r))$. Applying this procedure $O(\log(\varepsilon^{-1})$ and using linearity of expectation gives us the Lemma 3.4.3. Consider the following procedure to prove Lemma 3.4.3.

Algorithm 5: Reduction($x_0$, $F(x)$, $\mathcal{P}$, $\text{ALG}$, $c$, $\delta$, $r$)

Set $T = \log_{c^{-1}}(2r^2)$

repeat

for $i = 0 \to T$

for $j = 0 \to \log_{c^{-1}}(2 \log_{c^{-1}}(2r^2)$

Set $x_{ij} = \text{ALG}(x_i, F)$

end

Set $x_{i+1} = \min_j x_{ij}$

end

Compute error estimates $E_1 = \mathcal{P}(x_0)$, $E_2 = \mathcal{P}(x_T)$

Set $E = \frac{E_2}{E_1}$.

until $E \leq 0.5$

Output: $x_T$

Note that since for every $x_{ij}$ we have that

$$F(x_{ij}) - F(x^*) \leq c (F(x_i) - F(x^*))$$

with probability at least $\delta$, therefore we have that

$$F(x_{i+1}) - F(x^*) \leq c (F(x_i) - F(x^*))$$
with probability at least \(1 - \delta^{\log_{\delta^{-1}}(2\log_{\delta^{-1}}(\varepsilon^2)} = 1 - \frac{1}{2\log_{\delta^{-1}}(\varepsilon^2)}.\) Taking a union bound over the outer loop gives us that with probability at least 1/2 we have that

\[
F(x_T) - F(x^*) \leq \frac{1}{2r^2} (F(x_i) - F(x^*))
\]

Moreover by the property of the estimates given by \(\mathcal{P}\) we know that in this case we have that \(E \leq 0.5.\) Therefore we have that with probability at least 1/2 the repeat loop computes an \(x_T\) that reduces error by at least a factor of 1/2 and we can verify it. Therefore in expectation the loop runs a total of 2 times. The total runtime of the above procedure can easily seen to be \(O ((T + T') \log(r) \log(\varepsilon^{-1}).\)

Further suppose we are given a procedure with the guarantee that for any \(\varepsilon\) in expected running time \(T_\varepsilon\) it produces a point \(x'\) such that

\[
F(x') - \min F(x) \leq \varepsilon (F(x_0) - \min F(x))
\]

We now run this procedure for time \(T_{\varepsilon/2}\). By Markov’s inequality with probability at least 1/2 we have a point that satisfies

\[
F(x') - \min F(x) \leq \varepsilon (F(x_0) - \min F(x))
\]

It is now easy to see that if we repeat the above procedure \(\log(\gamma^{-1})\) many times and take the \(x\) with the minimum value we have a point \(x'\) such that

\[
F(x') - \min F(x) \leq \varepsilon (F(x_0) - \min F(x))
\]

with probability at least \(1 - \gamma.\)
3.6.4 Matrix Concentration for Sampling with Replacement

*Proof of Lemma 3.4.4.* The proof of the lemma follows the proof of Lemma 4 in [52]. We only state the differences. We use the inequality given in [79].

**Lemma 3.6.4.** Let $Y_1 \ldots Y_k$ be independent random positive semidefinite matrices of size $d \times d$. Let $Y = \sum Y_i$ and let $Z = \mathbb{E}[Y]$. If $Y_i \leq R.Z$ then

$$\Pr \left[ \sum Y_i \leq (1 - \varepsilon)Z \right] \leq de^{-\frac{\varepsilon^2}{2m}} \quad \text{and} \quad \Pr \left[ \sum Y_i \geq (1 + \varepsilon)Z \right] \leq de^{-\frac{\varepsilon^2}{2m}}.$$

Note that the expectation of $Y_j/m = a_i a_i^T/m$. Moreover note that each

$$\frac{Y_j}{m} \leq \max_i \frac{a_i a_i^T}{m} \frac{\sum k \gamma_k}{\gamma_i} \leq \frac{A^T A}{c \log d \varepsilon^{-2}}$$

The inequality follows from noting that $m \geq \sum \gamma_i$ and Equation 10 in [52]. The calculations now follow exactly in the same way as in the proof in [52].

\[ \Box \]

3.7 Conclusion

In this chapter we provided an algorithm for linear regression as well as its extensions to GLM problems under regularity conditions. The key improvement provided by our algorithms in both the cases is to improve the $\sqrt{\kappa \cdot n}$ term appearing in the analysis of accelerated gradient methods to $\sqrt{\kappa \cdot d}$ leading to a significant improvement in the case when $n >> d$. Our algorithms come about via carefully blending known accelerated gradient based techniques with matrix sub-sampling using leverage scores. In the next chapter we consider the more general case when the function $F$ can potentially be non-convex.
Chapter 4

Finding Approximate Local Minima

Faster Than Gradient Descent

4.1 Introduction

Finding a global minimizer of a non-convex optimization problem is NP-hard. Thus, the standard goal of efficient non-convex optimization algorithms is instead to find a local minimum. This problem has become increasingly important as the state-of-the-art in machine learning is attained by non-convex machines, many of which are variants of deep neural networks. Experiments in [76, 53, 48] suggest that fast convergence to a local minimum is sufficient for training neural nets, while convergence to critical points (points with vanishing gradients) is not. Theoretical works have also affirmed the same phenomenon for other machine learning problems (see [72, 24, 28, 73] and the references therein).

In this chapter, we give a provable linear(in dimension)-time algorithm for finding an approximate local minimum in smooth non-convex optimization. It applies to the general setting of ERM based optimization, and in particular to the optimization problem of training deep neural networks. Furthermore, the running time bound of our algorithm is the fastest
known even for the more lenient task of computing a point with vanishing gradient (called a critical point), for a wide range of parameters.

Formally, the problem of unconstrained mathematical optimization is stated in general terms as that of finding the minimum value that a function attains over Euclidean space, i.e.

$$\min_{x \in \mathbb{R}^d} F(x).$$ (4.1.1)

If $F$ is convex, the above formulation is *convex optimization* and is solvable in (randomized) polynomial time even if only a valuation oracle to $F$ is provided. A crucial property of convex functions is that “local optimality implies global optimality”, allowing for greedy algorithms to reach the global optimum efficiently. Unfortunately, this is no longer the case if $F$ is non-convex; indeed, even a degree four polynomial can be NP-hard to optimize \[81\]. In fact even the task of checking whether a point is not a local minimum \[106\] is also NP-hard. Thus, for non-convex optimization one has to settle for the more modest goal of reaching points which approximate local optima.

As defined in Chapter 1, a particular class of optimization problems of particular interest in machine learning is the ERM problem. These are optimization functions $F : \mathbb{R}^d \mapsto \mathbb{R}$ of the finite-sum form

$$F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$ (4.1.2)

Such functions arise when minimizing loss over a training set, where each example $i$ in the set corresponds to one loss function $f_i$ in the summation.

We say that the function $F$ is second-order smooth if it has Lipschitz continuous gradient and Lipschitz continuous Hessian. Formally, consider the following definition.

**Definition 4.1.1 (ε-approximate local minimum).** We say that a point $x$ is an $\varepsilon$-approximate local minimum if it satisfies:

$$\|\nabla F(x)\| \leq \varepsilon \quad \text{and} \quad \nabla^2 F(x) \succeq -\sqrt{\varepsilon \cdot L_3} \cdot I,$$

83
where $L_3$ is the Lipschitz constant of $\nabla^2 F(x)$ in the $\ell_2$ norm.

We say that a point $x$ is an $\varepsilon$-critical point if it satisfies the gradient condition above, but not necessarily the second-order condition. Critical points include saddle points in addition to local minima. We remark that $\varepsilon$-approximate local minima (even with $\varepsilon = 0$) are not necessarily close to any local minimum, neither in domain nor in function value. However, if we assume in addition the function satisfies the (robust) strict-saddle property \cite{90,70} (see Section 4.2 for the precise definition), then an $\varepsilon$-approximate local minimum is guaranteed to be close to a local minimum for sufficiently small $\varepsilon$.

Our main theorem below states the time required for the proposed algorithm $\text{FastCubic}$ to find an $\varepsilon$-approximate local minimum for second-order smooth functions.

**Theorem** (informal). Ignoring smoothness parameters, the running time of $\text{FastCubic}$ to return an $\varepsilon$-approximate local minimum is

\[
\tilde{O} \left( \left( \frac{n}{\varepsilon^{3/2}} + \frac{n^{3/4}}{\varepsilon^{7/4}} \right) \cdot \mathbb{T}_{h,1} \right) \text{ for (4.1.2)} \quad \text{or} \quad \tilde{O} \left( \frac{1}{\varepsilon^{7/4}} \cdot \mathbb{T}_h \right) \text{ for the general (4.1.1)}.
\]

Above, $\mathbb{T}_h$ is the time to compute Hessian-vector product for $\nabla^2 F(x)$ and $\mathbb{T}_{h,1}$ is that for an arbitrary $\nabla^2 f_i(x)$.

The full statement of Theorem 4.3.1 can be found in Section 4.2.

Hessian-vector products can be computed in linear time —meaning $\mathbb{T}_{h,1} = O(d)$ and $\mathbb{T}_{h,1} = O(nd)$— for many machine learning problems such as generalized linear models and training neural networks \cite{119,8}. We explain this more generally in Section 4.2.1. Therefore,

**Corollary 4.1.2.** Algorithm $\text{FastCubic}$ returns an $\varepsilon$-approximate local minimum for the optimization problem of training a neural network in time

\[
\tilde{O} \left( \frac{nd}{\varepsilon^{3/2}} + \frac{n^{3/4}d}{\varepsilon^{7/4}} \right).
\]

84
Another important aspect of our algorithm is that even in terms of just reaching an $\varepsilon$-critical point, i.e. a point that satisfies $\|\nabla F(x)\| \leq \varepsilon$ without any second-order guarantee, FastCubic is faster than all previous results (see Table 4.1 for a comparison).

The fastest methods to find critical points for a smooth non-convex function are gradient descent and its derivatives, jointly known as first-order methods. These methods are extremely efficient in terms of per-iteration complexity; however, they necessarily suffer from a $1/\varepsilon^2$ convergence rate [112], and only higher-order methods can break this $1/\varepsilon^2$ bottleneck [115]. For certain ranges of parameters, our FastCubic finds local minima even faster than first-order methods, even though they only find critical points. This is depicted in Table 4.1.

### 4.1.1 Related work

In this section we survey various results/works which are related to the results presented in this chapter. We first survey the results that existed at the time the first version of the paper.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Total Time Achieving $|\nabla F(x)| \leq \varepsilon$</th>
<th>Second-Order Guarantee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient Descent (GD)</td>
<td>$O\left(\frac{nd}{\varepsilon^2}\right)$</td>
<td>n/a</td>
</tr>
<tr>
<td>SVRG [15]</td>
<td>$O\left(\frac{nd + n^{2/3}d}{\varepsilon^2}\right)$</td>
<td>n/a</td>
</tr>
<tr>
<td>SGD [74]</td>
<td>$O\left(\frac{d}{\varepsilon^4}\right)$</td>
<td>n/a</td>
</tr>
<tr>
<td>noisy SGD [71]</td>
<td>$O\left(\frac{d^{C_1}}{\varepsilon^4}\right)$</td>
<td>$\nabla^2 F(x) \geq -\frac{\varepsilon^{1/C_2}}{d} I$</td>
</tr>
<tr>
<td>cubic regularization [115]</td>
<td>$\tilde{O}\left(\frac{nd^{C_1-1} + d^2}{\varepsilon^4}\right)$</td>
<td>$\nabla^2 F(x) \geq -\frac{\varepsilon^{1/2}}{d} I$</td>
</tr>
<tr>
<td>this paper</td>
<td>$\tilde{O}\left(\frac{nd}{d^{C_1}}\right)$</td>
<td>$\nabla^2 F(x) \geq -\frac{\varepsilon^{1/2}}{d} I$</td>
</tr>
<tr>
<td>this paper</td>
<td>$\tilde{O}\left(\frac{nd}{d^{C_1}} + \frac{n^{3/4}d}{\varepsilon^4}\right)$</td>
<td>$\nabla^2 F(x) \geq -\frac{\varepsilon^{1/2}}{d} I$</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of known methods in existence before the release of the first manuscript with this work

*aHere $C_1, C_2$ are two constants that are not explicitly written. We believe $C_1 \geq 4$. 

85
manuscript [6] containing the results in this chapter appeared. Further we survey the plethora of exciting developments this particular area has seen since the manuscript.

Methods that Provably Reach Critical Points  Recall that only a gradient oracle is needed to reach a critical point. The most commonly used algorithm in practice for training non-convex learning machines such as deep neural networks is stochastic gradient descent (SGD), also known as stochastic approximation [123] and its derivatives. Some practical enhancements widely used in practice are based on Nesterov’s acceleration [111] and adaptive regularization [57]. The variance reduction technique, introduced in [126], was extremely successful in convex optimization, but only recently there was a non-convex counterpart with theoretical benefits introduced [15].

Methods that Provably Reach Local Minima  The recent work of Ge et al. [70] showed that a noise-injected version of SGD in fact converges to local minima instead of critical points, as long as the underlying non-convex function is strict-saddle. Their theoretical running time is a large polynomial in the dimension and not competitive with our method (see Table 4.1).

The work of Lee et al. [90] shows that gradient descent, starting from a random point, almost surely converges to a local minimum of a strict-saddle function. The rates of convergence and precise step-sizes that are required are, however, yet unknown.

If second-order information (i.e., the Hessian oracle) is provided, the cubic-regularization method of Nesterov and Polyak [115] converges in $O\left(\frac{1}{\epsilon^3}\right)$ iterations. However, each iteration of Nesterov-Polyak requires solving a cubic function which, in general, takes time super-linear in the input representation.

One natural direction is to apply an approximate trust region solver, such as the linear-time solver of [80], to approximately solve the cubic regularization subroutine of Nesterov-Polyak. However, the approximation needed by a naive calculation makes this approach
even slower than vanilla gradient descent. Our main challenge is to obtain approximate second-order local-minima and simultaneously improve upon gradient descent.

**Parallel/Subsequent Work** Since the appearance of the first version of the manuscript [6] containing the results in this chapter, the question of obtaining faster algorithms to achieve approximate local minima has garnered huge interest in the research community with a plethora of results relaxing some of the assumptions made in this work under different models of access to the function. We survey these results in this section.

Concurrently and independently of our work, Carmon et al. [41] showed the same bounds as achieved by us albeit with a different algorithm. The key underlying idea here was to either follow a negative eigen-direction of the Hessian to find sufficient decrease or to ensure that the function is not very non-convex in a local region and extending accelerated gradient descent analysis to this special case. This result was further improved in the work of Allen-Zhu [14]. Carmon et al. [43] further provided a first-order algorithm, based on Accelerated Gradient Descent which converges to first-order optima at the fast rate of $\tilde{O}(\varepsilon^{-7/4})$ (the same rate as shown in this paper) albeit without the requirement of explicitly computing the smallest eigenvector of the Hessian. Allen-Zhu and Li [17] (also see Xu et al. [146]) further extended the above result to finding approximate local minima by providing a generic way to convert a critical point finding algorithm to a local minima finding replacing Hessian vector products to only gradient computations. In a parallel line of work Jin et al. [82, 83] build upon and improve the results of Ge et al. [71] to show the efficient convergence of perturbed gradient descent to local minima.

While the above results capture the case of full gradient based algorithms and hence the fast rate of $\tilde{O}(\varepsilon^{-7/4})$, the recent work of Allen-Zhu [13] (also see [93]) considers the same problem but with access to a stochastic gradient oracle. The best rate in this setting to converge to an approximate local minima as shown by [13] is $\tilde{O}(\varepsilon^{-3.25})$ stochastic gradient evaluations. Further the work of Carmon et al. [42] provides nearly matching lower bounds
for algorithms converging to approximate critical points demonstrating that the results provided in this chapter are nearly tight. However, there still remains a gap between the upper bound of $\tilde{O}(\varepsilon^{-7/4})$ gradient evaluations and the lower bound of $\Omega(\varepsilon^{-12/7})$ to achieve approximate critical points. It is an interesting open question to close this gap.

### 4.1.2 Our Techniques

Our algorithm is based on the cubic regularization method of Nesterov and Polyak \[115, 45, 46\]. At a high level, cubic regularization states that if we can minimize a cubic function $m(h) \doteq g^\top h + \frac{1}{2} h^\top \mathbf{H} h + \frac{L_3}{6} \| h \|^3$ exactly, where $g = \nabla F(x)$, $\mathbf{H} = \nabla^2 F(x)$, and $L_3$ is the second-order smoothness of the function $F$, then we can iteratively perform updates $x' \leftarrow x + h$, and this algorithm converges to an $\varepsilon$-approximate local minimum in $O(1/\varepsilon^{3/2})$ iterations. Unfortunately, solving this cubic minimization problem exactly, to the best of our knowledge, requires a running time of $O(d^\omega)$ where $\omega$ is the matrix multiplication constant. Getting around this requires five observations.

The first observation is that, minimizing $m(h)$ up to a constant multiplicative approximation (plus a few other constraints) is sufficient for showing an iteration complexity of $O(1/\varepsilon^{3/2})$. The proof techniques to show this observation are based on extending Nesterov and Polyak.

The second observation is that the minimizer $h^*$ of $m(h)$ must be of the form $h^* = (\mathbf{H} + \lambda^* \mathbf{I})^+ g + v$, where $\lambda^* \geq 0$ is some constant satisfying $\mathbf{H} + \lambda^* \mathbf{I} \succeq 0$, and $v$ is the smallest eigenvector of $\mathbf{H}$ and $^+$ denotes the pseudo-inverse of a matrix. This can be viewed as moving in a mixture direction between choosing $h \leftarrow v$, and choosing $h$ to follow a shifted Newton’s direction $h \leftarrow (\mathbf{H} + \lambda^* \mathbf{I})^+ g$. Intuitively, we wish to reduce both the computation of $(\mathbf{H} + \lambda^* \mathbf{I})^+ g$ and $v$ to Hessian-vector products.

---

1More specifically, we need $m_t(h) \leq \frac{1}{C} \min_t \{ m_t(h) \}$ for some constant $C$. In addition, we need to have good bounds on $\| h \|$ and $\| \nabla m(h) \|$.  

---
The first task of computing \((H + \lambda*I)^{+}g\) can be slow, and even if \(H + \lambda*I\) is strictly positive-definite, computing it has a complexity depending on the (possibly huge) condition number of \(H + \lambda*I\) \cite{133}. The third observation is that it suffices to pick some \(\lambda' > \lambda^*\) so both (1) the condition number of \(H + \lambda'I\) is small and (2) the vectors \((H + \lambda*I)^{-1}g\) and \((H + \lambda'I)^{-1}g\) are close. This relies on the structure of \(m(h)\).

The second task of computing \(v\) has a complexity depending on \(1/\sqrt{\delta}\) where \(\delta\) is the target additive error \cite{67,69}. The fourth observation is that the choice \(\delta = \sqrt{\varepsilon}\) suffices for the outer loop of cubic regularization to make sufficient progress. This reduces the complexity to compute \(v\).

Finally, finding the correct value \(\lambda^*\) itself is as hard as minimizing \(m_t(h)\). The fifth step is to design an iterative scheme that makes only logarithmic number of guesses on \(\lambda^*\). This procedure either finds the correct one (via binary search), or finds an approximate one, \(\lambda'\), but satisfying \((H + \lambda*I)^{-1}g\) and \((H + \lambda'I)^{-1}g\) being sufficiently close.

Putting all the observations together, and balancing all the parameters, we can obtain a cubic minimization subroutine (see \texttt{FastCubicMin} in Algorithm 4) that runs in time \(O(nd + n^{3/4}d/\varepsilon^{1/4})\).

### 4.2 Preliminaries

We make the following Lipschitz continuity assumptions for the gradient and Hessian of the target function \(F\). Namely, there exist \(L_2, L_3 > 0\) such that

**Assumption 4.2.1.** \(\forall x, y \in \mathbb{R}^d:\ |\nabla^2 F(x)| \leq L_2 \text{ and } |\nabla^2 F(x) - \nabla^2 F(y)| \leq L_3 \|x - y\|\).

**Definition 4.2.2.** We assume the following complexity parameters on the access to \(F(x)\):

- Let \(T_g \in \mathbb{R}^*\) be the time complexity to compute \(\nabla F(x)\) for any \(x \in \mathbb{R}^d\).

- Let \(T_h \in \mathbb{R}^*\) be the time complexity to compute \((\nabla^2 F(x)\) for any \(x, v \in \mathbb{R}^d\).
**Definition 4.2.3.** We say that $F$ is of finite-sum form if $F = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$ and $\|\nabla^2 f_i(x)\| \leq L_2$ for each $i \in [n]$. In this case, we define $T_{h,1}$ to be the time complexity to compute $(\nabla^2 f_i(x))v$ for arbitrary $x, v \in \mathbb{R}^d$ and $i \in [n]$.

Next we define the strict-saddle function for which an $\varepsilon$-approximate local minimum is almost equivalent to a local minimum [70, 90].

**Definition 4.2.4 (strict saddle).** Suppose $F(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ is twice differentiable. For $\alpha, \beta, \gamma \geq 0$, we say $F$ is $(\alpha, \beta, \gamma)$-strict saddle if every $x \in \mathbb{R}^d$ satisfies at least one of the following three conditions:

1. $\|\nabla F(x)\| \geq \alpha$.
2. $\lambda_{\text{min}}(\nabla^2 F) \leq -\beta$.
3. There exists a local minimum $x^*$ that is $\gamma$-close to $x$ in Euclidean distance.

We see that if a function is $(\alpha, \beta, \gamma)$-strict saddle, then for $\varepsilon < \min\{\alpha, \beta^2\}$ an $\varepsilon$-approximate local minimum is $\gamma$-close to some local minimum.

**Two Known Subroutines** Our running time of FastCubic relies on the following recent results for approximate matrix inverse and approximate PCA:

**Theorem 4.2.5 (Approximate Matrix Inverse).** Suppose matrix $M \in \mathbb{R}^{d \times d}$ satisfies $\|M\| \leq L_2$ and $\lambda I + M \succeq \delta I$ for constants $\lambda, \delta, L_2 > 0$. Let $\kappa = \frac{\lambda + L_2}{\delta}$. Then, we can compute vector $x$ satisfying

$$
\|x - (\lambda I + M)^{-1}b\| \leq \varepsilon \|b\|, \quad (4.2.1)
$$

using Accelerated gradient descent (AGD) in $O\left(\kappa^{1/2} \log(\kappa/\varepsilon)\right)$ iterations, each requiring $O(d)$ time plus the time needed to multiply $M$ with a vector.

Moreover, suppose $M = \frac{1}{n} \sum_{i=1}^{n} M_i$ where each $M_i$ is symmetric and satisfies $\|M_i\| \leq L_2$. If $M_ib$ can be computed in time $O(d')$ for each $i$ and vector $b$, then accelerated SVRG [128, 19] computes a vector $x$ that satisfies equation (4.2.1) in time $O\left(\max\{n, n^{3/4} \kappa^{1/2}\} \cdot d' \cdot \log^2 (\kappa/\varepsilon)\right)$.  

90
We refer to the running time for this computation as $T_{\text{inverse}}(\kappa, \varepsilon)$ and the algorithm as $A$.

Above, the SVRG based running time shall be used only towards our finite-sum case in Definition 4.2.3.

**Theorem 4.2.6** (AppxPCA [68, 69, 16]). Let $M \in \mathbb{R}^{d \times d}$ be a symmetric matrix with eigenvalues $1 \geq \lambda_1 \geq \cdots \geq \lambda_d \geq 0$. With probability at least $1 - p$, AppxPCA produces a unit vector $w$ satisfying $w^T M w \geq (1 - \delta_x)(1 - \varepsilon)\lambda_{\text{max}}(M)$. The total running time is $\tilde{O}(T_{\text{inverse}}(1/\delta_x, \varepsilon\delta_x))$.

### 4.2.1 Computing Hessian-Vector Product in Linear Time

In this section we sketch the intuition regarding why Hessian-vector products can be computed in linear time in many interesting (especially machine learning) problems which can be represented as a differentiable circuit. We start by showing that the gradient can be computed in linear time. The algorithm is often referred to as back-propagation, which dates back to Werbos’s PhD thesis [141], and has been popularized by Rumelharte et al. [125] for training neural networks.

**Claim 4.2.7** (back-propagation, informally stated). Suppose a real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ can be evaluated by a differentiable circuit of size $N$. Then, the gradient $\nabla F$ can be computed in time $O(N + d)$ (using a circuit of size $O(N + d)$). $\blacksquare$

The claim follows from simple induction and chain-rule, and is left to the readers. In the training of neural networks, often the size of circuits that computes the objective $F$ is proportional to (or equal to) the number of parameters $d$. Thus the gradient $\nabla F$ can be computed in time $O(d)$ using a circuit of size $d$.

$^2$Technically, we assume that the gradient of each gate can be computed in $O(1)$ time.
Next, we consider computing $\nabla^2 F(p \cdot x \cdot q \cdot v)$ where $v \in \mathbb{R}^d$. Let $g(x) := \langle \nabla F(x), v \rangle$ be a function from $\mathbb{R}^d$ to $\mathbb{R}$. Then, we see that if suffices to compute the gradient of $g$, since

$$\nabla^2 F(x) \cdot v = \nabla g(x).$$

We observe that $g(x)$ can be evaluated in linear time using circuit of size $O(d)$ since we’ve shown $\nabla F(x)$ can. Thus, using Claim 4.2.7 again on function $g$, we conclude that $\nabla g(x)$ can also be computed in linear time.

4.3 Main Results

In this section we present our main results for finding approximate local minima. We describe our main algorithm as Algorithm 3. Following the cubic regularization method of Nesterov and Polyak [115] our algorithm leverages the following upper bound on the change in objective value as we move from a point $x_t$ to $x_t + h$: (which follows via a standard argument based on the Taylor expansion to the third order and the mean value theorem)

$$\forall h \in \mathbb{R}^d: \quad F(x_t + h) - F(x_t) \leq m_t(h) \triangleq \nabla F(x_t)^\top h + \frac{h^\top \nabla^2 F(x_t) h}{2} + \frac{L_3}{6} \|h\|^3. \quad (4.3.1)$$

Denote by $h^*_t$ an arbitrary minimizer of $m_t(h)$. Algorithm 3 uses the sub-routine Algorithm 4 (which forms the core of our presentation) to construct an approximate minimizer of $m_t(h)$ (denoted as $h'$ in the algorithm). The main theorem which we show in this chapter is as follows:

**Theorem 4.3.1.** FastCubic (Algorithm 3) starts from a point $x_0$ and outputs a point $x$ such that

$$\|\nabla F(x)\| \leq \varepsilon \quad \text{and} \quad \lambda_{\min}(\nabla^2 F(x)) \geq -\sqrt{L_3} \varepsilon$$

in total time (denoting by $D \triangleq F(x_0) - F(x^*)$)

---

3We assume here that the original circuits are twice differentiable
Algorithm 3 \texttt{FastCubic}(F, x_0, \varepsilon, L_3, L_2)

\textbf{Input:} \ F(x) \ that \ satisfies \ \textbf{A4.2.1} \ with \ parameters \ L_2 \ and \ L_3; \ a \ starting \ vector \ x_0; \ a \ target \ accuracy \ \varepsilon.

\begin{align*}
\kappa & \leftarrow \left(\frac{900}{L_3}\right)^{1/2} \\
\text{for} \ t = 0 \ \text{to} \ \infty \ \text{do} \\
\quad m_t(h) & \equiv \nabla F(x_t)^\top h + \frac{h^\top \nabla^2 F(x_t) h}{2} + \frac{L_3}{6} \|h\|^3 \\
(\lambda, v, v_{\min}) & \leftarrow \texttt{FastCubicMin}(\nabla F(x_t), \nabla^2 F(x_t), L_3, L_2, \kappa) \\
h' & \leftarrow \text{either} \ v \text{ or} \ \frac{\lambda v_{\min}}{2L_3} \text{ whichever \ gives \ smaller \ value \ for} \ m_t(h); \\
\text{Set} \ \ x_{t+1} & \leftarrow x_t + h' \\
\text{if} \ m_t(h') > -\frac{c_{3/2}}{c\sqrt{L_3}} \ \text{then} \\
\quad \text{return} \ \ x_{t+1}. \\
\end{align*}

\begin{itemize}
\item \(\tilde{O}\left(\frac{D\sqrt{L_3}}{\varepsilon^{3/2}} \cdot \mathbb{T}_g + \frac{DL_3^{1/4} \sqrt{L_2}}{\varepsilon^{1/4}} \cdot \mathbb{T}_h\right), \text{ or} \)
\item \(\tilde{O}\left(\frac{D\sqrt{L_3}}{\varepsilon^{3/2}} \cdot (\mathbb{T}_g + n \mathbb{T}_{h,1}) + \frac{Dn^{3/4}L_3^{1/4} \sqrt{L_2}}{\varepsilon^{7/4}} \cdot \mathbb{T}_{h,1}\right) \text{ in the finite-sum setting (see Definition 4.2.3).} \)
\end{itemize}

\begin{flushright}
\textcircled{c} \text{is a constant; we proved} \ c = 2.4 \times 10^6 \text{ works} \)
\end{flushright}

\textbf{Here} \(\tilde{O}\) \textbf{hides} logarithmic factors in \(L_3, L_2, 1/\varepsilon, \text{d}, \text{and in} \ \max_x \{\|\nabla F(x)\|\}. \)

\textbf{Roadmap} \hspace{1cm} \text{The rest of the chapter is devoted to providing a complete proof of} \textbf{Theorem 4.3.1} \hspace{1cm} \text{We begin by providing in Section 4.4 a characterization of the minimizer (}h^*\text{) of the cubic }m_t(h) \hspace{0.5cm} \text{Lemma 4.4.1} \hspace{0.5cm} \text{and some other useful lemmas and corollaries. Further in Section 4.5 we provide the main sub-routine} \texttt{FastCubicMin} \hspace{0.5cm} \text{(Algorithm 4) and state} \textbf{Theorem 4.5.1} \hspace{0.5cm} \text{and} \textbf{Theorem 4.5.2} \hspace{0.5cm} \text{which state our correctness/runtime guarantees for} \texttt{FastCubicMin} \hspace{0.5cm} \text{Further in Section 4.6 we complete the proof of} \textbf{Theorem 4.3.1} \hspace{0.5cm} \text{based on} \textbf{Theorem 4.5.1} \hspace{0.5cm} \text{and} \textbf{Theorem 4.5.2} \hspace{0.5cm} \text{} \textbf{Theorem 4.5.1} \hspace{0.5cm} \text{and} \textbf{Theorem 4.5.2} \hspace{0.5cm} \text{are proved in 4.7 The proof of} \textbf{Theorem 4.5.1} \hspace{0.5cm} \text{requires our main lemma} \textbf{Lemma 4.7.2} \hspace{0.5cm} \text{which we prove in Section 4.8 We finish the chapter by providing the proof of} \textbf{Lemma 4.4.1}}
4.4 Characterization of the Minimizer of the Cubic Subproblem

For notational simplicity in the remainder of the chapter, we omit the subscript \(t\) when stating \(m_t\) and use \(g\) and \(H\) as generic placeholders for \(\nabla F(x_t)\) and \(\nabla^2 F(x_t)\) respectively. In particular we focus on the following problem:

\[
\begin{aligned}
\text{minimize} & \quad m(h) = g^T h + \frac{h^T H h}{2} + \frac{L_3}{6} \|h\|^3 \\
\text{where} & \quad H \text{ is a symmetric matrix with } \|H\|_2 \leq L_2.
\end{aligned}
\]  

(4.4.1)

Recall from the previous section that we have denoted by \(h^*\) an arbitrary minimizer of \(m(h)\). We have the following lemma which characterizes \(h^*\): (a variant of this lemma has appeared in [45], and we provide the proof in Section 4.9 for the sake of completeness)

**Lemma 4.4.1.** We have that \(h^*\) is a minimizer of \(m(h)\) if and only if there exists \(\lambda^* \geq 0\) such that

\[
H + \lambda^* I \succeq 0, \quad (H + \lambda^* I)h^* = -g, \quad \|h^*\| = \frac{2\lambda^*}{L_3}.
\]

The objective value in this case is given by

\[
m(h^*) = -\frac{1}{2} g^T (H + \lambda^* I) g - \frac{2(\lambda^*)^3}{3L_3^2} \leq 0.
\]

The following corollary follows from **Lemma 4.4.1** and its proof:

**Corollary 4.4.2.** The value \(\lambda^*\) in **Lemma 4.4.1** is unique, and for every \(\lambda\) satisfying \(H + \lambda I > 0\), we have

\[
\|H^{-1} g\| > \frac{2\lambda}{L_3} \iff \lambda^* > \lambda \quad \text{and} \quad \|H^{-1} g\| < \frac{2\lambda}{L_3} \iff \lambda^* < \lambda.
\]
Proof of Corollary 4.4.2. The uniqueness of $\lambda^*$ follows from Lemma 4.9.1. To prove the second part we first make some observations about the function

$$p(y) = \frac{2y}{L_3} - \|(H + yI)^{-1}g\|$$

defined on the domain $y \in (-\lambda_{\min}(H), \infty)$. Note that $p(y)$ is continuous and strictly increasing over the domain and $p(y) \to \infty$ as $y \to \infty$.

The corollary requires us to show that

$$p(\lambda) < 0 \iff \lambda^* > \lambda \quad \text{and} \quad p(\lambda) > 0 \iff \lambda^* < \lambda.$$ 

We begin by showing the first equivalence. To see the backward direction note that if $\lambda^* > \lambda > -\lambda_{\min}(H)$, by the characterization of $\lambda^*$ in Lemma 4.9.1 we have that $\|(H + \lambda^*I)^{-1}g\| = \frac{2\lambda^*}{L_3}$ i.e. $p(\lambda^*) = 0$ which implies that $p(\lambda) < 0$ as $p(y)$ is a strictly increasing function. For the forward direction note that since $p(y)$ is continuous and strictly increasing we see that the range of the function contains $[p(\lambda), \infty)$. Since $p(\lambda) < 0$ there must exist a $\lambda^* > \lambda$ such that $p(\lambda^*) = 0$ which by the characterization in Lemma 4.9.1 finishes the proof.

Now we will prove that $p(\lambda) > 0 \iff \lambda^* < \lambda$. To see the forward direction note that if $\lambda^* \geq \lambda$ then $p(\lambda^*) = 0$ and $p(\lambda) > 0$ which contradicts the fact that $p(y)$ is strictly increasing. For the backward direction we consider two cases. Firstly if $\lambda^* > -\lambda_{\min}(H)$ the conclusion follows similarly by the monotonicity of $p(y)$. If $\lambda^* = -\lambda_{\min}$ then by Lemma 4.9.1 we have that $g$ has no component in the lowest eigenspace of $H$ and therefore if we extend $p(y)$ to $-\lambda_{\min}(H)$ by defining

$$p(-\lambda_{\min}(H)) = \frac{-2\lambda_{\min}(H)}{L_3} - \|(H - \lambda_{\min}(H)I)^+g\|$$
we get that \( p(y) \) is increasing in the domain \( y \in [-\lambda_{\min}(H), \infty) \). Now from the characterization of the solution in Lemma 4.9.1 we can see that \( p(-\lambda_{\min}(H)) \geq 0 \) and therefore by monotonicity \( p(\lambda) > 0 \). This finishes the proof.

The following proposition provides a crude upper bound on \( \lambda^* \):

**Proposition 4.4.3.** We have \( \lambda^* \leq B \equiv \max \{ 2L_2 + \sqrt{L_3 \| g \|}, 1 \} \) with \( \lambda^* \) defined in Lemma 4.4.1.

**Proof.** We have \( L_3 \| (H + BI)^{-1} g \| \leq \frac{L_3 \| g \|}{\lambda_{\min}(H + BI)} \leq \frac{L_3 \| g \|}{B - L_2} < 2B \) and therefore \( \lambda^* \leq B \) due to Corollary 4.4.2.

### 4.5 Solving the Cubic Subproblem - *FastCubicMin*

We now provide our main algorithm *FastCubicMin* (Algorithm 4) to approximately minimize the cubic \( m(h) \) (4.4.1). Note that *FastCubicMin* returns two vectors \( v, v_{\min} \) and a scalar \( \lambda \). To remind the reader while using *FastCubicMin* in Algorithm 3, we construct the approximate minimizer \( h' \) to be either \( v \) or \( \frac{\lambda v_{\min}}{2L_3} \), whichever gives a smaller value for \( m(h) \).

The following theorems respectively state the main guarantee satisfied by the output of *FastCubicMin* and the total time spent in the execution of *FastCubicMin*.

**Theorem 4.5.1 (Guarantees of *FastCubicMin*).** Given a cubic \( m(h) \) (4.4.1), let \( (\lambda, v, v_{\min}) \) be the output of *FastCubicMin*. Define

\[
h' = \arg\min_{h \in \{v, \frac{\lambda v_{\min}}{2L_3}\}} m(h).
\]

Setting parameter \( \kappa = \frac{12}{\sqrt{L_3 \| g \|}} \), we have that one of the following conditions hold

\[
\text{Either } m(h') \leq 0 \quad \text{and} \quad 3000m(h') \leq m(h^*) \tag{4.5.1}
\]

We have not tried to optimize the various constants appearing in Theorem 4.5.1 and therefore are likely to be significantly smaller.
Algorithm 4 \texttt{FastCubicMin}(g, H, L_3, L_2, \kappa) (main algorithm for cubic minimization)

\textbf{Input:} \( g \) a vector, \( H \) a symmetric matrix, parameters \( \kappa, L_3 \) and \( L_2 \) which satisfies \(-L_2 I \preceq L_3 I\).

\textbf{Output:} \((\lambda, v, v_{\text{min}})\)

1: \( B \leftarrow L_2 + \sqrt{L_3 \|g\| + \frac{1}{\kappa}} \).
2: \( \tilde{\varepsilon} \leftarrow 1/(10000 \left( \max \{L_3, \|g\|, \frac{3\kappa}{10}, B, 1 \} \right)^{20}) \).
3: \( \lambda_0 \leftarrow 2B \).
4: \textbf{for} \( i = 0 \) to \( \infty \) \textbf{do}
5: \quad Compute \( v \) such that \( \|v + (H + \lambda_i I)^{-1} g\| \leq \tilde{\varepsilon} \).
6: \quad \textbf{if} \( L_3 \|v\| \in [2\lambda_i - L_3 \tilde{\varepsilon}, 2\lambda_i + L_3 \tilde{\varepsilon}] \) \textbf{then}
7: \quad \quad \textbf{return} \((\lambda_i, v, \emptyset)\).
8: \quad \textbf{else if} \( L_3 \|v\| > 2\lambda_i + L_3 \tilde{\varepsilon} \) \textbf{then}
9: \quad \quad \textbf{return} \texttt{BinarySearch}(\lambda_1 = \lambda_{i-1}, \lambda_2 = \lambda_i, \tilde{\varepsilon}) \).
10: \quad \textbf{else if} \( L_3 \|v\| < 2\lambda_i - L_3 \tilde{\varepsilon} \) \textbf{then}
11: \quad \quad \text{Let Power Method find vector} \( w \) that is \( \frac{9}{10} \)-appx leading eigenvector of \((H + \lambda_i I)^{-1}:
12: \quad \quad \quad \frac{9}{10} \lambda_{\text{max}}((H + \lambda_i I)^{-1}) \leq w^\top (H + \lambda_i I)^{-1} w \leq \lambda_{\text{max}}((H + \lambda_i I)^{-1}).
13: \quad \quad \text{Compute a vector} \( \tilde{w} \) such that \( \|\tilde{w} - (H + \lambda_i I)^{-1} w\| \leq \tilde{\varepsilon} \triangleq \frac{1}{60B} \).
14: \quad \quad \text{if} \( D > \frac{1}{2\kappa} \) \textbf{then}
15: \quad \quad \quad \hat{\lambda}_{i+1} \leftarrow \lambda_i - \frac{D}{2} \).
16: \quad \quad \quad \text{if} \( \hat{\lambda}_{i+1} > 0 \) \textbf{then} \( \lambda_{i+1} \leftarrow \hat{\lambda}_{i+1} \) \textbf{else} \( \lambda_{i+1} \leftarrow 0 \)
17: \quad \quad \text{else}
18: \quad \quad \quad \text{Use AppxPCA to find any unit vector} \( v_{\text{min}} \) such that \( v_{\text{min}}^\top H v_{\text{min}} \leq \lambda_{\text{min}}(H) + \frac{1}{10\kappa}. \)
19: \quad \quad \end{algorithm}
20: \quad \end{algorithm}
21: \quad \end{algorithm}
22: \quad \end{algorithm}
23: \quad \text{Flip the sign of} \( v_{\text{min}} \) \text{so that} \( g^\top v_{\text{min}} \leq 0 \).
24: \quad \textbf{return} \((\lambda_i, v, v_{\text{min}})\).
25: \quad \textbf{end if}
26: \quad \textbf{end if}
27: \textbf{end for}

97
**Algorithm 5** BinarySearch($\lambda_1, \lambda_2, \tilde{\epsilon}$) (binary search subroutine)

**Input:** $\lambda_1 \geq \lambda_2$, $L_3 \|(H + \lambda_1 I)^{-1} g\| \leq 2\lambda_1$, $L_3 \|(H + \lambda_2 I)^{-1} g\| \geq 2\lambda_2$, $\lambda_2 + \lambda_{\min}(H) > 0$

**Output:** $(\lambda, v, \emptyset)$

1. for $t = 1$ to $\infty$ do
2.  $\lambda_{\text{mid}} \leftarrow \frac{\lambda_1 + \lambda_2}{2}$
3.  Compute vector $v$ such that $\|v + (H + \lambda_{\text{mid}} I)^{-1} g\| \leq \tilde{\epsilon}/2$
4.  if $L_3 \|v\| \in [2\lambda_{\text{mid}} - L_3 \tilde{\epsilon}, 2\lambda_{\text{mid}} + L_3 \tilde{\epsilon}]$ then
5. 6.    return $(\lambda_{\text{mid}}, v, \emptyset)$
7.  else if $L_3 \|v\| + L_3 \tilde{\epsilon} \leq 2\lambda_{\text{mid}}$ then
8.    $\lambda_1 \leftarrow \lambda_{\text{mid}}$
9.  else if $L_3 \|v\| - L_3 \tilde{\epsilon} \geq 2\lambda_{\text{mid}}$ then
10.    $\lambda_2 \leftarrow \lambda_{\text{mid}}$
11.  end if
12. end for

$$m(h^*) \geq -\frac{\tilde{\epsilon}^{3/2}}{800\sqrt{L_3}}. \quad (4.5.2)$$

Further we are guaranteed that

If $m(h^*) \geq -\frac{\tilde{\epsilon}^{3/2}}{300\sqrt{L_3}}$, then $\|h'\| \leq \|h^*\| + \frac{\sqrt{\tilde{\epsilon}}}{4\sqrt{L_3}}$ and $\|\nabla m_t(h')\| \leq \frac{\epsilon}{2}. \quad (4.5.3)$

**Theorem 4.5.2** (Runtime of FastCubicMin). FastCubicMin with parameter $\kappa = \Theta\left(\frac{1}{\sqrt{L_3 \tilde{\epsilon}}}\right)$ runs in total time: (using $\tilde{O}$ to hide logarithmic factors in $L_3$, $L_2$, $1/\tilde{\epsilon}$, $d$, $\|\nabla F(x_t)\|$)

- $\tilde{O}\left(\frac{\sqrt{\tilde{\epsilon}}}{(\tilde{\epsilon}L_3)^{1/4}} \cdot T_h\right)$ where $T_h$ is the time to multiply $\nabla^2 F(x_t)$ to a vector;
- $\tilde{O}\left(\max\{n, n^{3/4}\frac{\sqrt{\tilde{\epsilon}}}{(\tilde{\epsilon}L_3)^{1/4}}\} \cdot T_{h,1}\right)$ where $T_{h,1}$ is the time to multiply $\nabla^2 f_i(x_t)$ with a vector.

While we defer the proofs of Theorem 4.5.1 and Theorem 4.5.2 to later sections, in the following section we provide a proof of Theorem 4.3.1 based on the guarantees provided by Theorem 4.5.1 and Theorem 4.5.2.
4.6 Proof of Theorem 4.3.1

We begin by stating the following useful lemma which shows that if (4.5.3) is achieved then we have have achieved the desired stopping criterion.

**Lemma 4.6.1.** If \( m_t(h^*) \geq -\frac{\varepsilon^{3/2}}{800\sqrt{L_3}} \) and \( h' \) is an approximate minimizer of \( m_t(h) \) satisfying

\[
\|h'\| \leq \|h^*\| + \frac{\varepsilon}{4\sqrt{L_3}} \quad \text{and} \quad \|\nabla m_t(h')\| \leq \frac{\varepsilon}{2},
\]

then we have that \( \|\nabla F(x_t + h')\| \leq \varepsilon \) and \( \lambda_{\min}(\nabla^2 F(x_t + h')) \geq -\sqrt{L_3}\varepsilon \).

We now provide a proof of Theorem 4.3.1 based on Theorems 4.5.1 and 4.5.2.

**Proof of Theorem 4.3.1** When \texttt{FastCubic} terminates, we have that

\[
m_t(h') > -\frac{\varepsilon^{3/2}}{c\sqrt{L_3}} \geq -\frac{\varepsilon^{3/2}}{800\sqrt{L_3}}.
\]

Lemma 4.6.1 now implies that in the last iteration of \texttt{FastCubic} our output satisfies \( \|\nabla F(x_t + h')\| \leq \varepsilon \) and \( \lambda_{\min}(\nabla^2 F(x_t + h')) \geq -\sqrt{L_3}\varepsilon \). This finishes the proof with respect to the accuracy conditions.

As for the running time, in every iteration except for the last one, \texttt{FastCubic} satisfies \( m_t(h') \leq -\Omega(-\varepsilon^{3/2}/\sqrt{L_3}) \). Therefore by (4.3.1), we must have decreased the objective by at least \( \Omega(-\varepsilon^{3/2}/\sqrt{L_3}) \) in this round, and this cannot happen for more than \( O\left(\frac{(F(x_0) - f^*)\sqrt{L_3}}{\varepsilon^{3/2}}\right) \) iterations.

The final running time of \texttt{FastCubic} follows from this bound together with Theorem 4.5.2.

\[\square\]

We end this section by providing the proof of Lemma 4.6.1.

**Proof of Lemma 4.6.1.** We first collect some useful facts. Let us denote by \( g = \nabla F(x_t) \) and \( H = \nabla^2 F(x_t) \) in this proof. Note that we have

\[
\nabla m_t(h) = g + Hh + \frac{L_3}{2}\|h\|h.
\]
Let $h^* = \arg\min h m_t(h)$ and recall that the characterization result in Lemma 4.4.1 shows $H + \frac{L_3|h^*|}{2}I \succeq 0$. The first and second order optimality conditions respectively now give us that,

\[ g^\top h^* + (h^*)^\top H h^* + \frac{L_3}{2}\|h^*\|^3 = \nabla m_t(h^*)^\top h^* = 0 \quad (4.6.1) \]
\[ (h^*)^\top H h^* + \frac{L_3}{2}\|h\|^3 = (h^*)^\top \left( H + \frac{L_3\|h^*\|}{2}I \right) h^* \succeq 0. \quad (4.6.2) \]

This in particular implies that,

\[
m_t(h^*) = g^\top h^* + \frac{(h^*)^\top H h^*}{2} + \frac{L_3\|h^*\|^3}{6} \overset{\ominus}{=} -\frac{(h^*)^\top H h^*}{2} - \frac{L_3\|h^*\|^3}{3} \overset{\ominus}{=} \frac{L_3}{4}\|h^*\|^3 - \frac{L_3}{3}\|h^*\|^3 = -\frac{L_3}{12}\|h^*\|^3 \quad (4.6.3)\]

where \(\ominus\) uses (4.6.1) and \(\ominus\) uses (4.6.2).

We now consider the norm of the gradient at a point $x_t + h'$ for any $h' \in \mathbb{R}^d$. We have that

\[
\|\nabla F(x_t + h')\| \leq \|\nabla F(x_t + h') - \nabla m_t(h')\| + \|\nabla m_t(h')\|
\leq \left\| \nabla F(x_t) + \int_0^1 \nabla^2 F(x_t + \tau h') h' d\tau \right\| \left( g + H h' + \frac{L_3}{2}\|h'\|^2 \right) + \|\nabla m_t(h')\|
\leq \left\| \int_0^1 (\nabla^2 F(x_t + \tau h') - H) h' d\tau \right\| \|h'\|^2 + \frac{L_3}{2}\|h'\|^2 + \|\nabla m_t(h')\|
\overset{\ominus}{=} L_3\|h'\|^2 \int_0^1 \tau d\tau + \frac{L_3}{2}\|h'\|^2 + \|\nabla m_t(h')\|
\]

where \(\ominus\) follows from the Lipschitz continuity on the Hessian (A4.2.1). The above gives us the following conclusion

\[
\|\nabla F(x_t + h')\| \leq L_3\|h'\|^2 + \|\nabla m_t(h')\| \quad (4.6.4)
\]
Further note that for all \( h' \in \mathbb{R}^d \), by A4.2.1 we have that

\[
\| \nabla^2 F(x_t + h') - \nabla^2 F(x_t) \| = \| \nabla^2 F(x_t + h') - H \| \leq L_3 \| h' \|.
\]

However, this implies

\[
\lambda_{\min}(\nabla^2 F(x_t + h')) \geq \lambda_{\min}(H) - L_3 \| h' \|.
\]  

(4.6.5)

because if two matrices \( A \) and \( B \) satisfies \( \| A - B \| \leq p \), then they must satisfy \( |\lambda_{\min}(A) - \lambda_{\min}(B)| \leq p \) as well. We consider two cases: if \( \lambda_{\min}(H) \geq 0 \), then we have

\[
\lambda_{\min}(\nabla^2 F(x_t + h')) \geq -L_3 \| h' \|.
\]  

(4.6.6)

Otherwise \( \lambda_{\min}(H) < 0 \). Let \( \nu_d \) be the normalized eigenvector corresponding to \( \lambda_{\min}(H) \), and define

\[
\tilde{h} = -\text{sign}(g^T \nu_d) \cdot \frac{2\lambda_{\min}(H)}{L_3} \nu_d.
\]

We calculate \( m_t(\tilde{h}) \) as follows:

\[
m_t(\tilde{h}) = g^T \tilde{h} + \frac{\tilde{h}^T H \tilde{h}}{2} + \frac{L_3}{6} \| \tilde{h} \|^3 \leq \frac{\tilde{h}^T H \tilde{h}}{2} + \frac{L_3}{6} \| \tilde{h} \|^3
\]

\[
= \frac{2(\lambda_{\min}(H))^2}{L_3^2} \nu_d^T H \nu_d + \frac{4|\lambda_{\min}(H)|^3}{3L_3^2} \quad \Theta
\]

\[
= \frac{2(\lambda_{\min}(H))^3}{L_3^2} + \frac{4|\lambda_{\min}(H)|^3}{3L_3^2} \quad \Theta
\]

(4.6.7)

where \( \Theta \) uses \( \nu_d^T H \nu_d = \lambda_{\min}(H) < 0 \), and \( \Theta \) uses the assumption that \( \lambda_{\min}(H) < 0 \). Since by definition \( m_t(h^*) \leq m_t(\tilde{h}) \), we can deduce from inequality (4.6.7) that

\[
\lambda_{\min}(H) \geq -\left( \frac{3L_3^2 |m_t(h^*)|}{2} \right)^{1/3}.
\]  

(4.6.8)
Now we put together inequalities (4.6.5) and (4.6.8), and obtain

$$\lambda_{\text{min}}(\nabla^2 F(x_t + h')) \geq - \left( \frac{3L_3^2|m_t(h^*)|}{2} \right)^{1/3} - L_3\|h'\|. \quad (4.6.9)$$

We will now use (4.6.6) and (4.6.9) derived above to prove Lemma 4.6.1. Note that our assumption that $m_t(h^*) \geq -\frac{\varepsilon^{3/2}}{800\sqrt{L_3}}$, along with inequality (4.6.3), tells us that

$$\|h^*\| \leq \frac{\sqrt{\varepsilon}}{4\sqrt{L_3}}.$$

This, together with the assumption on $\|h'\|$ in Lemma 4.6.1 implies that

$$\|h'\| \leq \frac{\sqrt{\varepsilon}}{2\sqrt{L_3}}. \quad (4.6.10)$$

Since we also assume $\|\nabla m_t(h')\| \leq \frac{\varepsilon}{2}$, we have from (4.6.4) that

$$\|\nabla F(x_t + h')\| \leq L_3\|h'\|^2 + \|\nabla m_t(h')\| \leq \frac{\varepsilon}{4} + \frac{\varepsilon}{2} \leq \varepsilon.$$

For the second-order condition, we combine (4.6.9), (4.6.10) and the condition $m_t(h^*) \geq -\frac{\varepsilon^{3/2}}{800\sqrt{L_3}}$ to obtain

$$\lambda_{\text{min}}(\nabla^2 F(x_t + h')) \geq - \left( \frac{3L_3^2\max\{0, -m_t(h^*)\}}{2} \right)^{1/3} - L_3\|h'\|$$

$$\geq - \left( \frac{3L_3^2\varepsilon^{3/2}}{1600} \right)^{1/3} - \frac{\sqrt{L_3\varepsilon}}{2} \geq -\sqrt{L_3\varepsilon}.$$

This finishes the proof. \qed
4.7 Proof of Theorems 4.5.1 and 4.5.2

In the rest of the chapter we now focus on proving Theorems 4.5.1 and 4.5.2. As argued above this suffices to finish our proof of the main result (Theorem 4.3.1). The section is structured as follows. We begin by stating and proving Lemma 4.7.1 which establishes some invariants regarding the values $\lambda_i$ maintained by $\text{FastCubicMin}$. Further in Subsection 4.7.1, we prove Theorem 4.5.1 and in Subsection 4.7.2, we prove Theorem 4.5.2. The proof of Theorem 4.5.1 makes use of Lemma 4.7.2 which is our main lemma. We prove this lemma in the next section.

Lemma 4.7.1. The following statements hold for all $i$ until $\text{FastCubicMin}$ terminates

1. $\lambda_i \in [0, 2B], \lambda_i + \lambda_{\max}(H) \leq 3B$

2. $\lambda_i + \lambda_{\min}(H) \geq \frac{3}{10\epsilon}$

3. $\lambda_{i+1} + \lambda_{\min}(H) \leq \frac{3}{4}(\lambda_i + \lambda_{\min}(H))$ unless $\lambda_{i+1} = 0$

Moreover when $\text{FastCubicMin}$ terminates at Line 20 we have $\lambda_i + \lambda_{\min}(H) \leq \frac{1}{10\epsilon}$.

Proof of Lemma 4.7.1. The lemma follows via induction.

To see (a) and (b) at the base case $i = 0$, recall that the definitions of $B$ and $L_2$ together ensure $\lambda_0 + \lambda_{\max}(H) \leq 3B$ and $\lambda_0 + \lambda_{\min}(H) \geq \frac{3}{10\epsilon}$. Also $\lambda_0 \in [0, 2B]$.

Suppose now for some $i \geq 0$ properties (a) and (b) hold. It is easy to check that $\lambda_i \leq \lambda_{i-1}$ and thus we have $\lambda_i + \lambda_{\max}(H) \leq 2B$ and $\lambda_i \leq 2B$. This implies property (a) at iteration $i + 1$ also hold. We now proceed to show property (c) at iteration $i$ and property (b) at iteration $i + 1$. Recall that the algorithm ensures

$$\frac{9}{10} \lambda_{\max}((H + \lambda_i I)^{-1}) \leq w^\top(H + \lambda_i I)^{-1}w \leq \lambda_{\max}((H + \lambda_i I)^{-1}),$$

and by the definition of $\bar{w}$ we have

$$\frac{9}{10} \lambda_{\max}((H + \lambda_i I)^{-1}) - 2\epsilon \leq \bar{w}^\top w - \epsilon \leq \lambda_{\max}((H + \lambda_i I)^{-1}).$$ (4.7.1)
Now, since \( \frac{3}{10k} \leq \lambda_i + \lambda_{\min}(H) \leq 3B \) from the inductive assumption, it follows from the choice of \( \hat{\varepsilon} \) that

\[
2\hat{\varepsilon} \leq \frac{1}{30B} \leq \frac{1}{10(\lambda_i + \lambda_{\min}(H))} = \frac{\lambda_{\max}((H + \lambda_i I)^{-1})}{10}. \tag{4.7.2}
\]

Plugging Equation (4.7.2) into Equation (4.7.1) we get

\[
\frac{8}{10} \frac{1}{\lambda_i + \lambda_{\min}(H)} = \frac{8}{10} \lambda_{\max}((H + \lambda_i I)^{-1}) \leq \hat{w}^T w - \hat{\varepsilon} \leq \lambda_{\max}((H + \lambda_i I)^{-1}) = \frac{1}{\lambda_i + \lambda_{\min}(H)}. \tag{4.7.3}
\]

Inverting this chain of inequalities, we have

\[
\frac{\lambda_i + \lambda_{\min}(H)}{2} \leq D \leq \frac{5(\lambda_i + \lambda_{\min}(H))}{8}. \tag{4.7.4}
\]

From this we derive the following implications:

\[
D \leq \frac{1}{2\kappa} \implies (\lambda_i + \lambda_{\min}(H)) \leq \frac{1}{\kappa} \tag{4.7.4}
\]

\[
D > \frac{1}{2\kappa} \implies (\lambda_i + \lambda_{\min}(H)) > \frac{4}{5\kappa} \tag{4.7.5}
\]

If Condition (4.7.4) happens, our algorithm \texttt{FastCubicMin} outputs on Line 20; in such a case (4.7.4) implies our desired inequality \( \lambda_i + \lambda_{\min}(H) \leq \frac{1}{\kappa} \). If Condition (4.7.5) happens, our choice \( \bar{\lambda}_{i+1} \leftarrow \lambda_i - \frac{D}{2} \) and Equation (4.7.3) together imply that

\[
\frac{3}{4} (\lambda_i + \lambda_{\min}(H)) \geq \bar{\lambda}_{i+1} + \lambda_{\min}(H) \geq \frac{11}{16} (\lambda_i + \lambda_{\min}(H))
\]

Combining this with (4.7.5) we get that

\[
\frac{3}{4} (\lambda_i + \lambda_{\min}(H)) \geq \bar{\lambda}_{i+1} + \lambda_{\min}(H) \geq \frac{11}{16} \left( \frac{4}{5\kappa} \right) \geq \frac{3}{10\kappa}. \]
Therefore, we conclude that property (c) at iteration \( i \) holds and property (b) at iteration \( i + 1 \) hold because \( \lambda_{i+1} \geq \hat{\lambda}_{i+1} \). This finishes the proof of Lemma 4.7.1.

### 4.7.1 Proof of Theorem 4.5.1

The following lemma collects a set of sufficient conditions under which the assumptions in Theorem 4.5.1 can be satisfied.

**Lemma 4.7.2.** Given a cubic \( m(h) \) \([4.4.1]\), consider an algorithm that outputs a real \( \lambda \in [0, 2B] \), a vector \( v \in \mathbb{R}^d \), and a unit vector \( v_{\min} \in \mathbb{R}^d \). Additionally, suppose numbers \( \kappa, \tilde{\varepsilon} \geq 0 \) satisfying the following conditions:

\[
\tilde{\varepsilon} \leq \frac{1}{10000 \left( \max \{ \kappa, L_3, L_2, \|g\|, \|(H + \lambda I)^{-1}\|, B \} \right)^2} \quad (4.7.6)
\]

\[
(H + (\lambda - L_3 \tilde{\varepsilon})I)^{-1} > 0 \quad (4.7.7)
\]

Moreover, suppose that the outputs \( (\lambda, v, v_{\min}) \) satisfy one of the following two cases:

**Case 1:** \( L_3 \| (H + \lambda I)^{-1} g \| \in [2\lambda - 2L_3 \tilde{\varepsilon}, 2\lambda + 2L_3 \tilde{\varepsilon}] \) and \( \| v + (H + \lambda I)^{-1} g \| \leq \tilde{\varepsilon} \)

**Case 2:** The following conditions are satisfied:

(a) \( \lambda \geq \lambda^* \) and \( \lambda + \lambda_{\min}(H) \leq \frac{1}{\kappa} \)

(b) \( L_3 \| (H + \lambda I)^{-1} g \| \leq 2\lambda \) and \( \| v + (H + \lambda I)^{-1} g \| \leq \tilde{\varepsilon} \)

(c) \( v_{\min}^T H v_{\min} \leq \lambda_{\min}(H) + \frac{1}{10\kappa} \)

Further define

\[ h' = \arg \min_{h \in \{v, \frac{\lambda_{\min}}{2L} \}} m(h) \]

Then we have that

\[
\begin{align*}
\text{either } & m(h^*) \geq 3000m(h') \quad \text{or} \quad m(h^*) \geq -\frac{32}{\kappa^3 L_3^2}. \quad (4.7.8)
\end{align*}
\]
where $h^* = \arg\min_h m(h)$. Further suppose $m(h^*) \geq -\frac{3/2}{300\sqrt{L_3}}$. Then $h'$ satisfies the following conditions:

$$
\|v\| \leq \|h^*\| + \frac{3}{\kappa L_3} \quad \text{and} \quad \|\nabla m(v)\| \leq \frac{\varepsilon}{4} + \frac{15}{\kappa^2 L_3}.
$$

(4.7.9)

To interpret the cases in 4.7.2 better, let us compare the conditions to the characterization provided in Lemma 4.4.1.

- In Case 1, up to a very small error $\bar{\varepsilon}$, we have essentially found a vector $h' = v$ that satisfies $h' \approx -(H + \lambda I)^{-1}g$ and $\|h'\| \approx \frac{2\lambda}{L_3}$. Therefore, owing to Lemma 4.4.2, $h'$ should be close to $h^*$. (This is the simple case.)

- In Case 2, we have only found a vector $v$ that satisfies $v \approx -(H + \lambda I)^{-1}g$ and $\|v\| \leq \frac{2\lambda}{L_3}$. In this case, we also compute an approximate lowest eigenvector $v_{\min}$ of $\lambda_{\min}(H)$ up to an additive $1/10\kappa$ accuracy (see case 2-c). We will make sure that, as long as the conditions in 2-a hold, then either $v$ or $\frac{\lambda v_{\min}}{2L_3}$ will be an approximate minimizer for $m(h)$. (This is the hard case.)

The following Lemma provides the observation that the BinarySearch subroutine (Algorithm 5) returns $(\lambda, v, \emptyset)$ that satisfies Case 1 of Lemma 4.7.2.

**Fact 4.7.3.** BinarySearch outputs a pair $\lambda$ and $v$ such that

$$
L_3\|(H + \lambda I)^{-1}g\| \in [2\lambda - 2L_3\bar{\varepsilon}, 2\lambda + 2L_3\bar{\varepsilon}] \quad \text{and} \quad \|v + (H + \lambda I)^{-1}g\| \leq \bar{\varepsilon}.
$$

**Proof.** The latter is guaranteed by line 3 in BinarySearch, and the former is implied by the latter because

$$
L_3\|(H + \lambda I)^{-1}g\| \in \left[L_3\|v\| - L_3\bar{\varepsilon}/2, L_3\|v\| + L_3\bar{\varepsilon}/2\right] \subseteq \left[2\lambda - 2L_3\bar{\varepsilon}, 2\lambda + 2L_3\bar{\varepsilon}\right].
$$

\[\square\]
We now provide the proof of Theorem 4.5.1.

**Proof of Theorem 4.5.1.** We begin by verifying that the output \((\lambda, v, v_{\min})\) of \texttt{FastCubicMin} satisfies the sufficient conditions of Lemma 4.7.2. To this end we have that

- \textbf{Lemma 4.7.1} implies \(\lambda \in [0, 2B]\).

- \(\lambda_i + \lambda_{\min}(H) \geq \frac{3}{10\kappa}\) from \textbf{Lemma 4.7.1} implies \(\|H + \lambda_i I\| \leq 4\kappa\). It is now immediate that the choice of \(\varepsilon\) on Line 2 satisfies the Condition (4.7.6) in the assumption of \textbf{Lemma 4.7.2}.

- Since \(\varepsilon < \frac{1}{10\kappa L_3}\) and \(\lambda_i + \lambda_{\min}(H) \geq \frac{3}{10\kappa}\) it follows that \(\|H + (\lambda_i - L_3 \varepsilon) I\| \geq 0\) which proves Condition (4.7.7) in \textbf{Lemma 4.7.2}.

- We now verify Case 1 and 2 in the assumption of \textbf{Lemma 4.7.2}. At the beginning of the algorithm, our choice \(\lambda_0 = 2B\) ensures (using Proposition 4.4.3) that \(L_3\|H + \lambda_0 I\| \leq 2\lambda_0\). Let us now consider the various places where the algorithm outputs:

  - If \texttt{FastCubicMin} terminates at Line 7 then we have \(\|v + (H + \lambda_i I)^{-1} g\| < \varepsilon\) and additionally

    \[L_3\|H + \lambda_i I\|^{-1} g \in [L_3\|v\| - L_3\varepsilon, L_3\|v\| + L_3\varepsilon] \subseteq [2\lambda_i - 2L_3\varepsilon, 2\lambda_i + 2L_3\varepsilon].\]

    Therefore, the output meets Case 1 requirement of \textbf{Lemma 4.7.2} with \(\lambda = \lambda_i\).

  - If \texttt{FastCubicMin} terminates at Line 9 then \(L_3\|H + \lambda_i I\|^{-1} g > L_3\|v\| - L_3\varepsilon \geq 2\lambda_i\). Obviously, we must have \(i \geq 1\) in this case because \(L_3\|H + \lambda_0 I\|^{-1} g < 2\lambda_0\). Therefore, Line 10 must have been reached at the previous iteration, so it implies \(L_3\|H + \lambda_{i-1} I\|^{-1} g < 2\lambda_{i-1}\). Together, these two imply that we can call \texttt{BinarySearch} with \((\lambda_{i-1}, \lambda_i)\). Owing to \textbf{Fact 4.7.3} the subroutine outputs a pair \((\lambda, v)\) satisfying the Case 1 requirement of \textbf{Lemma 4.7.2}.
– If \texttt{FastCubicMin} terminates on Line 20 we verify that Case 2 of Lemma 4.7.2 with $\lambda = \lambda_i$ holds. We first have

$$L_3 \|(H + \lambda_i I)^{-1}g\| \leq L_3\|v\| + L_3\tilde{\varepsilon} \leq 2\lambda_i.$$  

By Corollary 4.4.2 we also have that $\lambda_i \leq \lambda^*$. Lemma 4.7.1 tells us $\lambda_i$ satisfies $\lambda_i + \lambda_{\text{min}}(H) \leq \frac{1}{\kappa}$. Vector $v$ satisfies $\|v + (H + \lambda_i I)^{-1}g\| \leq \tilde{\varepsilon}$. Vector $v_{\text{min}}$ satisfies $v_{\text{min}}^T Hv_{\text{min}} \leq \lambda_{\text{min}}(H) + \frac{1}{10\kappa}$.

In sum, we have verified that all the assumptions of Lemma 4.7.2 hold. Now, using the special choice of $\kappa$ in \texttt{FastCubic}, Theorem 4.5.1 immediately follows from Lemma 4.7.2. 

\[
4.7.2 \quad \textbf{Proof of Theorem 4.5.2}
\]

Having proven the correctness of the algorithm (i.e. Theorem 4.5.1), we now provide a proof of Theorem 4.5.2 to bound the overall running time of \texttt{FastCubicMin}. We first state the following lemmas which bound the maximum number of iterations the \texttt{FastCubicMin} procedure and the \texttt{BinarySearch} procedure invoked from within \texttt{FastCubicMin} makes.

\textbf{Lemma 4.7.4.} \texttt{FastCubicMin} ends in at most $O(\log(B\kappa))$ outer loops.

\textit{Proof.} According to Lemma 4.7.1 we have $\frac{3}{2}(\lambda_{i-1} + \lambda_{\text{min}}(H)) \geq \lambda_i + \lambda_{\text{min}}(H)$ so the quantity $\lambda_i + \lambda_{\text{min}}(H)$ decreases by a constant factor per iteration (except possibly $\lambda_i = 0$ the last outer loop in which case we shall terminate in one more iteration). On one hand, we have began with $\lambda_0 + \lambda_{\text{min}}(H) \leq 3B$. On the other hand, we always have $\lambda_i + \lambda_{\text{min}}(H) \geq \frac{3}{10\kappa}$ according to Lemma 4.7.1. Therefore, the total number of outer loops is at most $O(\log(B\kappa))$. 

\textbf{Lemma 4.7.5.} Each invocation of \texttt{BinarySearch} (line 9 in \texttt{FastCubicMin}) ends in $O(\log(1/\tilde{\varepsilon}))$ iterations.
Proof of Lemma 4.7.5. We first note that in all iterations of \texttt{BinarySearch} it always satisfies

\[ L_3\|(H + \lambda_1 I)^{-1} g\| \leq 2\lambda_1 \quad \text{and} \quad L_3\|(H + \lambda_2 I)^{-1} g\| \geq 2\lambda_2. \quad (4.7.10) \]

This is true at the beginning. In each of the follow-up iterations, if we have set \( \lambda_1 \leftarrow \lambda_{\text{mid}} \) then it must satisfy \( L_3\|v\| + L_3\varepsilon \leq 2\lambda_{\text{mid}} \) but this implies \( L_3\|(H + \lambda_{\text{mid}} I)^{-1} g\| \leq 2\lambda_{\text{mid}} \) according to triangle inequality and \( \|v + (H + \lambda_{\text{mid}} I)^{-1} g\| \leq L_3\varepsilon \); similarly, if we have set \( \lambda_2 \leftarrow \lambda_{\text{mid}} \) then it must satisfy \( L_3\|(H + \lambda_{\text{mid}} I)^{-1} g\| \geq 2\lambda_{\text{mid}} \).

Suppose now the loop has run for at least \( \log_2(\frac{\lambda_1 - \lambda_2}{\varepsilon}) \) iterations where \( \varepsilon \triangleq \frac{L_3\varepsilon_{\text{c}}}{40B} \). Then, it must satisfy \( \lambda_1 - \lambda_2 \leq \hat{\varepsilon} \). At this point, we compute

\[
(H + \lambda_1 I)^{-1} = (H + \lambda_2 I)^{-1} - (\lambda_1 - \lambda_2)(H + \lambda_2 I)^{-1}(H + \lambda_1 I)^{-1}
\]

and therefore

\[
L_3\|(H + \lambda_1 I)^{-1} g\| \geq L_3\|(H + \lambda_2 I)^{-1} g\| - L_3\|(\lambda_1 - \lambda_2)(H + \lambda_2 I)^{-1}(H + \lambda_1 I)^{-1} g\|
\geq \frac{1}{2}\lambda_1 - \hat{\varepsilon}\|(H + \lambda_2 I)^{-1}\| \cdot 2\lambda_1 \geq 2\lambda_1 - 2\hat{\varepsilon} - \hat{\varepsilon}\|(H + \lambda_2 I)^{-1}\| \cdot 2\lambda_1
\]

Above, inequality \( \circ \) uses \((4.7.10)\) and \( \lambda_1 - \lambda_2 \leq \hat{\varepsilon} \); inequality \( \triangledown \) uses again \( \lambda_1 - \lambda_2 \leq \hat{\varepsilon} \).

Now, we notice that \( \|(H + \lambda_2 I)^{-1}\| \leq \frac{1}{\varepsilon_{\text{c}}} \) and \( \lambda_1 \leq 2B \) because \( \lambda_2 \) only increases and \( \lambda_1 \) only decreases through the execution of the algorithm. Therefore by the choice of \( \hat{\varepsilon} = \frac{\varepsilon_{\text{c}}}{40B} \), we get

\[
L_3\|(H + \lambda_1 I)^{-1} g\| \geq 2\lambda_1 - L_3\hat{\varepsilon}/5.
\]

A completely analogous argument also shows that

\[
L_3\|(H + \lambda_2 I)^{-1} g\| \leq 2\lambda_2 + L_3\hat{\varepsilon}/5.
\]
Therefore, in the immediate next iteration when picking \( \lambda_{\text{mid}} \leftarrow (\lambda_1 + \lambda_2)/2 \), it must satisfy

\[
2\lambda_{\text{mid}} - L_3\varepsilon/2 \leq 2\lambda - L_3\varepsilon/5 \leq L_3 \| (H + \lambda_{\text{mid}}I)^{-1} g \| \leq 2\lambda_2 + L_3\varepsilon/5 \leq 2\lambda_{\text{mid}} + L_3\varepsilon/2.
\]

Then, at this iteration when \( v \) is computed to satisfy \( \| v + (H + \lambda_{\text{mid}}I)^{-1} g \| \leq \varepsilon/2 \), we also have

\[
2\lambda_{\text{mid}} - L_3\varepsilon \leq L_3 \| v \| \leq 2\lambda_{\text{mid}} + L_3\varepsilon
\]

which means \( \text{BinarySearch} \) will stop in this iteration. In sum, we have concluded that there will be no more than \( O(\log((\lambda_1 - \lambda_2)/c_1 L_3\varepsilon)) \) iterations.

Further since \( \text{Lemma 4.7.1} \) implies that, in \( \text{FastCubicMin} \), \( \lambda_i \leq 2B \) and \( \lambda_i + \lambda_{\text{min}}(H) \geq \frac{3}{10\kappa} \). This taken together with the upper bound on the number of iterations, along with the choice of \( \varepsilon \) finishes the proof.

Further we collect some claims which control the running time of specific steps of \( \text{FastCubicMin} \).

**Matrix Inverse**

Since the key component of the running time is the computation of \( (H + \lambda_iI)^{-1} b \) for different vectors \( b \) we will first bound the condition number of the matrix \( (H + \lambda_iI)^{-1} \) via the following lemma

**Claim 4.7.6.** Through out the execution of \( \text{FastCubicMin} \) and \( \text{BinarySearch} \) whenever we compute \( (H + \lambda_iI)^{-1} b \) for some vector \( b \) it satisfies \( \frac{\lambda_i + L_2}{\lambda_i + \lambda_{\text{min}}(H)} \leq 10\kappa L_2 \).

**Proof of Claim 4.7.6.** We first focus on \( \text{Line 5} \) and \( \text{Line 11} \) of \( \text{FastCubicMin} \). There are two cases. If \( \lambda_i \geq 2L_2 \), then according to \( -L_2I \leq H \leq L_2I \) we can bound \( \frac{\lambda_i + L_2}{\lambda_i + \lambda_{\text{min}}(H)} \leq 3 \) because the left hand side is the largest when \( \lambda_i = 2L_2 \). If \( \lambda_i < 2L_2 \), then by \( \text{Lemma 4.7.1} \) we know \( \lambda_i + \lambda_{\text{min}}(H) \geq \frac{3}{10\kappa} \). This implies \( \frac{\lambda_i + L_2}{\lambda_i + \lambda_{\text{min}}(H)} \leq 10\kappa L_2 \).
We now focus on [Line 3 of BinarySearch]. We claim that all values $\lambda_{mid}$ iterated over BinarySearch also satisfy $\lambda_{mid} + \lambda_{\min}(H) \geq \frac{3}{10\kappa}$ (because the values $\lambda_{mid} \geq \lambda_i$ and $\lambda_i$ satisfies $\lambda_i + \lambda_{\min}(H) \geq \frac{3}{10\kappa}$ according to [Lemma 4.7.1]). Therefore, the same case analysis (with respect to $\lambda_{mid} \geq 2L_2$ and $\lambda_{mid} < 2L_2$) also gives $\frac{\lambda_i + L_2}{\lambda_i + \lambda_{\min}(H)} \leq 10\kappa L_2$. 

**Claim 4.7.7.** [Line 5 of FastCubicMin] and [Line 3 of BinarySearch] runs in time

$$O(T_{\text{inverse}}(\kappa L_2; \tilde{\varepsilon})).$$

**Proof.** Whenever we compute $(H + \lambda_i I)^{-1}b$ for some vector $v$ it satisfies $\|b\| \leq 1/\varepsilon$; therefore to find $v$ satisfying $\|v + (H + \lambda_i I)^{-1}b\| \leq \varepsilon$ it suffices to find $\|v + (H + \lambda_i I)^{-1}b\| \leq \varepsilon^2\|b\|$. This costs a total running time $O(T_{\text{inverse}}(\kappa L_2; \tilde{\varepsilon}))$ according to Theorem 4.2.5.

Therefore by Theorem 4.2.5 every time we need to multiply a vector $v$ to $(H + \lambda I)^{-1}$ to error $\delta$, the time required to approximately solve such a system is $T_{\text{inverse}}(O(\kappa L_2), \delta)$. We will state our running time with respect to $T_{\text{inverse}}$ as it is the dominant operation in the algorithm.

**Power Method**

We now bound the running time of Power Method in [Line 11 of FastCubicMin]. It is a folklore (cf. [16, Appendix A]) that getting any constant multiplicative approximation to the leading eigenvector of any PSD matrix $M \in \mathbb{R}^{d \times d}$ requires only $O(\log d)$ iterations, each computing $Mb$ for some vector $b$. In our case, we have $M = (H + \lambda_i I)^{-1}$ so we cannot compute $Mb$ exactly. Fortunately, folklore results on inaccurate power method suggests that, as long as each $Mb$ is computed to a very good accuracy such as $\varepsilon^{-\Omega(\log d)}$, then we can still get a constant multiplicative approximate leading eigenvector that satisfies [Line 11 of FastCubicMin]. Ignoring all the details (which are quite standard and can be found for instance in [16, Appendix A]), we claim that
**Claim 4.7.8.** Line 11 of **FastCubicMin** runs in time $\tilde{O} \left( \mathbb{T}_{\text{inverse}} \left( \kappa L_2, \varepsilon^{-\Theta(\log(d))} \right) \right) = \tilde{O} \left( \mathbb{T}_{\text{inverse}} \left( \kappa L_2, \varepsilon \right) \right)$.

**Lowest Eigenvector**

We will now focus on the running time for the computation of the lowest eigenvector of the Hessian which is required in Line 18. We recall Theorem 4.2.6 from Section 4.2 which uses Shift and Invert to compute the largest eigenvalue of a matrix.

Since we are concerned with the lowest eigenvector of $H$ and by assumption $-L_2 I \preceq H \preceq L_2 I$, we can equivalently compute the largest eigenvector of $M \triangleq I - \frac{H + L_2 I}{2L_2}$ which satisfies $0 \preceq M \preceq I$. Note that computing $Mv$ is of the same time complexity as computing $Hv$. By setting $\varepsilon = \delta_x = \frac{0.01}{\kappa L_2}$ in Theorem 4.2.6 and running AppxPCA, we obtain a unit vector $w$ such that

$$1 - \frac{w^T H w + L_2}{2L_2} = w^T M w \geq (1 - 2\delta_x) \lambda_{\text{max}}(M) \lesssim \lambda_{\text{max}}(M) - 2\delta_x \geq 1 - \frac{\lambda_{\text{min}}(H) + L_2}{2L_2} - 2\delta_x$$

Above, $\lesssim$ uses $\lambda_{\text{max}}(M) \leq 1$. Rearranging the terms we obtain $w^T H w \leq \lambda_{\text{min}}(H) + 0.05\kappa$ as desired. In sum,

**Claim 4.7.9.** The approximate lowest eigenvector computation on Line 18 runs in time $\tilde{O} \left( \mathbb{T}_{\text{inverse}} \left( \kappa L_2, \varepsilon \right) \right)$.

**Putting It All Together**

*Proof of Theorem 4.5.2* Putting together our bounds in Lemma 4.7.4 and Lemma 4.7.5 which bound the number of iterations, as well as our bounds in Claim 4.7.8, Claim 4.7.7, and Claim 4.7.9 for power method, matrix inverse, and lowest eigenvectors, we conclude that our total running time of **FastCubicMin** is at most $\tilde{O} \left( \mathbb{T}_{\text{inverse}}(\kappa L_2, \varepsilon) \right)$, where $\tilde{O}$ contains factors polylogarithmic in $\kappa, L_3, L_2, B, d$. 

112
Further by putting together the choice of $\kappa$ and $\bar{\varepsilon}$ as well as the running time of either accelerated gradient descent or accelerated SVRG from Theorem 4.2.5 into formula $O(\kappa L_2, \bar{\varepsilon})$, we finish the proof.

4.8 Proof of Main Lemma 4.7.2

The proof will be divided into two parts. The first part focuses on ensuring (4.7.8) and the second part focuses on ensuring (4.7.9). Each part further subdivides the proof based on Case 1 vs Case 2.

4.8.1 Ensuring Condition (4.7.8)

To structure the proof for readability we subdivide the proof into multiple claims. We begin by considering Case 1.

Suppose Case 1 happens. According to Corollary 4.4.2, if $\bar{\varepsilon} = 0$ then $v$ is a minimizer of $m(h)$. The following claim extends this argument to the setting when $\bar{\varepsilon} > 0$:

Claim 4.8.1. If $\lambda$ and $v$ satisfy Case 1 and $\bar{\varepsilon}$ satisfies (4.7.6), then $m(v) \leq m(h^*) + \frac{1}{250 \kappa^3 L_3}$.

From the above it follows that either $m(h^*) \geq -\frac{8}{\kappa^3 L_3}$ otherwise $m(h^*) \geq 1.1 m(v)$ which satisfies (4.7.8). We now consider Case 2.

Suppose Case 2 happens We make the following two claims:

Claim 4.8.2. If $\lambda_{\min}(H) \leq -\frac{1}{\kappa}$ then $m(h^*) \geq 1500 \min \left\{ m(v), m \left( \frac{\lambda_{\min}}{2L_3} \right) \right\} - \frac{1}{500 \kappa L_3^3}$.

Claim 4.8.3. If $\lambda_{\min}(H) \geq -\frac{1}{\kappa}$ then $m(h^*) \geq 2m(v) - \frac{16}{\kappa L_3}$.

(4.7.8) now follows immediately from the two claims and the choice of $h'$.

Proof of Claim 4.8.1
Proof of Claim 4.8.1. Note that by the conditions of the theorem we have that $(H + (\lambda - L_3 \varepsilon)I)^{-1} \succ e_q$ and

$$L_3\|(H + (\lambda - L_3 \varepsilon))^{-1}g\| \geq 2\lambda - 2L_3 \varepsilon \quad \text{and} \quad L_3\|(H + (\lambda + L_3 \varepsilon))^{-1}g\| \leq 2\lambda - 2L_3 \varepsilon,$$

according to Corollary 4.4.2 we must have

$$\lambda^* \in [\lambda - L_3 \varepsilon, \lambda + L_3 \varepsilon] \quad (4.8.1)$$

This also implies (using our assumption on $\varepsilon$)

$$L_3\|v\| \leq [2\lambda^* - 5L_3 \varepsilon, 2\lambda^* + 5L_3 \varepsilon].$$

Next, consider the value $m(v)$

$$m(v) = g^T v + \frac{v^T Hv}{2} + \frac{L_3}{6}\|v\|^3 = g^T v + \frac{v^T (H + \lambda I)v}{2} - \|v\|^2 \left(\frac{\lambda}{2} - \frac{L_3\|v\|}{6}\right). \quad (4.8.2)$$

We bound the two parts on the right hand side of (4.8.2) separately. The first part

$$g^T v + \frac{v^T (H + \lambda I)v}{2} \leq -\frac{g^T (H + \lambda I)^{-1}g}{2} + \|g\|\varepsilon + \|(H + \lambda I)^{-1}\varepsilon\| \leq -\frac{g^T (H + \lambda I)^{-1}g}{2} + \frac{1}{1000\kappa^3 L_3^2} \quad (4.8.3)$$

$$\leq -\frac{g^T (H + \lambda I)^{-1}g}{2} + L_3\|g\|^2 \|(H + \lambda I)^{-1}g\| + \frac{1}{1000\kappa^3 L_3^2}$$

$$\leq -\frac{g^T (H + \lambda I)^{-1}g}{2} + \frac{1}{500\kappa^3 L_3^2}.$$
Above, inequalities $\heartsuit$ and $\clubsuit$ use the assumption on $\tilde{\varepsilon}$ in (4.7.6), and inequality $\spadesuit$ uses

$$-(H + \lambda I)^{-1} \leq -(H + (\lambda^* + L_3 \tilde{\varepsilon})I)^{-1}$$

$$= -(H + \lambda^* I)^{-1} - L_3 \tilde{\varepsilon}(H + \lambda^* I)^{-1}(H + (\lambda^* + L_3 \tilde{\varepsilon})I)^{-1}$$

Note that $(H + \lambda^* I)^{-1} > 0$ by Equations (4.8.1) and (4.7.7). The second part of (4.8.2) can be bounded as follows

$$\|v\|^2 \left(\frac{\lambda}{2} - \frac{L_3 \|v\|}{6}\right) \geq \frac{(2\lambda^* - 5L_3 \tilde{\varepsilon})^2}{L_3^2} \left(\frac{\lambda^* - L_3 \tilde{\varepsilon}}{2} - \frac{2\lambda^* + 5L_3 \tilde{\varepsilon}}{6}\right)$$

$$\geq \frac{2(\lambda^*)^3}{3L_3^2} - 1000\tilde{\varepsilon}L_3(\lambda^*)^2 \geq \frac{2(\lambda^*)^3}{3L_3^2} - \frac{1}{500\kappa^3L_3^2}$$

Above, inequality $\heartsuit$ uses $\lambda^* \leq B$ (owing to Proposition 4.4.3) and our assumption on $\tilde{\varepsilon}$ from (4.7.6). Putting these together we get that

$$m(v) \leq m(h^*) + \frac{1}{250\kappa^3L_3^2}. \quad \square$$

**Proofs for Claims 4.8.2 and 4.8.3** For notational simplicity, let us rotate the space into the basis in the eigenspace of $H$; let the $i$-th dimension correspond to the $i$-th largest eigenvalue $\lambda_i$ of $H$. We have $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_d = \lambda_{\min}$. Let $g_i$ denote the $i$-th coordinate of $g$ in this basis. Lemma 4.4.1 implies

$$m(h^*) = -\frac{1}{2} \sum_i \frac{g_i^2}{\lambda_i + \lambda^*} - \frac{2(\lambda^*)^3}{3L_3^2} = S_1 + S_2 - \frac{2(\lambda^*)^3}{3L_3^2}. \quad (4.8.4)$$

where we denote by

$$S_1 = -\sum_{i: \lambda_i + \lambda^* \geq \frac{1}{\kappa}} \frac{g_i^2}{\lambda_i + \lambda^*} \quad S_2 = -\sum_{i: 0 < \lambda_i + \lambda^* \leq \frac{1}{\kappa}} \frac{g_i^2}{\lambda_i + \lambda^*}$$
From Corollary 4.4.2 we can also obtain

\[ \sum_{i: \lambda_i + \lambda^* > 0} \frac{g_i^2}{(\lambda_i + \lambda^*)^2} \leq \frac{4(\lambda^*)^2}{L_3^2}. \] (4.8.5)

Now the assumption \( \| (H + \lambda I)^{-1} g \| \leq \frac{2\lambda}{L_3} \) is equivalent to

\[ \sum_{i} \frac{g_i^2}{(\lambda_i + \lambda)^2} \leq \frac{4\lambda^2}{L_3^2} \] (4.8.6)

We first collect some useful lemmas controlling the value of \( S_1, S_2 \).

**Lemma 4.8.4.** If \( \lambda_{\text{min}}(H) \leq -\frac{1}{\kappa} \) then \( S_2 \geq 1000 \cdot m \left( \frac{\lambda_{\text{min}}}{2L_3} \right) \)

**Proof of Lemma 4.8.4.** We compute that

\[ S_2 = - \sum_{i: 0 < \lambda_i + \lambda^* \leq \frac{1}{\kappa}} \frac{g_i^2}{\lambda_i + \lambda^*} = - \sum_{i: 0 < \lambda_i + \lambda^* \leq \frac{1}{\kappa}} \frac{g_i^2(\lambda_i + \lambda^*)}{(\lambda_i + \lambda^*)^2} \geq -\frac{1}{\kappa} \sum_{i: 0 < \lambda_i + \lambda^* \leq \frac{1}{\kappa}} \frac{g_i^2}{(\lambda_i + \lambda^*)^2} \]

\[ \overset{\oplus}{\geq} -\frac{4}{\kappa L_3^2} (\lambda^*)^2 \overset{\ominus}{\geq} -16\frac{|\lambda_{\text{min}}|^3}{L_3^2}. \] (4.8.7)

Above, \( \oplus \) uses (4.8.5), and \( \ominus \) follows because we have \( \lambda_{\text{min}}(H) \leq -\frac{1}{\kappa} \) in the assumption and have \( \lambda^* \leq -\lambda_{\text{min}}(H) + \frac{1}{\kappa} \) in the assumption of Case 2 of Lemma 4.7.2.

Let us now consider the value of the vector \( \frac{\lambda v_{\text{min}}}{2L_3} \). We have that

\[ m \left( \frac{\lambda v_{\text{min}}}{2L_3} \right) = \lambda g^T v_{\text{min}} + \frac{\lambda^2 v_{\text{min}}}{2L_3} + \frac{\lambda^3}{8L_3^2} \overset{\ominus}{\leq} \frac{\lambda g^T v_{\text{min}}}{2L_3} + \frac{\lambda^2 v_{\text{min}}}{16L_3^2} + \frac{\lambda^3}{48L_3^2} \]

\[ \overset{\oplus}{\leq} \frac{\lambda g^T v_{\text{min}}}{2L_3} + \frac{\lambda^2 v_{\text{min}}}{16L_3^2} - \frac{\lambda^2 v_{\text{min}}}{24L_3^2} \leq \frac{\lambda g^T v_{\text{min}}}{2L_3} + \frac{\lambda^2 v_{\text{min}}}{48L_3^2} \]

Above, \( \oplus \) is because our assumption \( \lambda_{\text{min}}(H) \leq -\frac{1}{\kappa} \) and assumption \( v_{\text{min}} H v_{\text{min}} \leq \lambda_{\text{min}}(H) + \frac{1}{10\kappa} \) together imply \( v_{\text{min}} H v_{\text{min}} \leq \frac{\lambda_{\text{min}}}{2} \). \( \ominus \) follows from \( \lambda_{\text{min}}(H) \leq -\frac{1}{\kappa} \) and \( \lambda \leq -\lambda_{\text{min}}(H) + \frac{1}{\kappa} \).
Now, recall that the sign of \(v_{\text{min}}\) is chosen so \(g^T v_{\text{min}}\) is non-positive, and therefore by our assumptions \(\lambda_{\text{min}}(H) \leq -\frac{1}{\kappa}\) and \(\lambda \leq -\lambda_{\text{min}}(H) + \frac{1}{\kappa}\), we get the following inequality:

\[
m \left( \frac{\lambda v_{\text{min}}}{2L_3} \right) \leq -\frac{|\lambda_{\text{min}}|^3}{48L_3^2}
\] (4.8.8)

Putting inequalities (4.8.8) and (4.8.7) together finishes the proof of Lemma 4.8.4.

**Lemma 4.8.5.** \(S_1 \geq 4m(v) - \frac{1}{250\kappa^3L_3^2}\)

**Proof of Lemma 4.8.5.** We have that

\[
m(v) = g^T v + \frac{v^T (H + \lambda I)v}{2} - \frac{\lambda}{2} \|v\|^2 + \frac{L_3^2}{6} \|v\|^3
\]

\[
\geq -\frac{g^T (H + \lambda I)^{-1}g}{2} - \|v\|^2 \left( \frac{\lambda}{2} - \frac{L_3^2}{6} \|v\|^2 \right) + \frac{1}{1000\kappa^3L_3^2}
\]

\[
\geq -\frac{g^T (H + \lambda I)^{-1}g}{2} - \frac{2 \lambda - 3L_3 \varepsilon}{L_3} \left( \frac{\lambda}{2} + \frac{L_3 \varepsilon}{3} \right) + \frac{1}{1000\kappa^3L_3^2}
\]

\[
\leq \frac{\lambda}{2} \frac{L_3^2}{3} + \frac{1}{500\kappa^3L_3^2}
\] (4.8.9)

Above, \(\textcircled{1}\) follows from inequality (4.8.3), as part of the proof of Claim 4.8.1; \(\textcircled{2}\) uses our condition on \(v\) which gives \(L_3 \|v\| \in [2 \lambda - 3L_3 \varepsilon, 2 \lambda + 3L_3 \varepsilon]\); \(\textcircled{3}\) uses our condition (4.7.6) on \(\varepsilon\).

We now bound \(S_1\). For this purpose first we note that if \(\lambda_i + \lambda^* \geq \frac{1}{\kappa}\) and \(\lambda - \lambda^* \leq \frac{1}{\kappa}\) then

\[2(\lambda_i + \lambda^*) \geq 1/\kappa + \lambda_i + \lambda^* \geq \lambda_i + \lambda.
\]

Therefore, the sum \(S_1\) satisfies

\[
S_1 = -\sum_{i: \lambda_i + \lambda^* \geq \frac{1}{\kappa}} \frac{g_i^2}{\lambda_i + \lambda^*} \geq -2 \sum_{i: 0 < \lambda_i + \lambda^* \leq \frac{1}{\kappa}} \frac{g_i^2}{(\lambda_i + \lambda)}
\]

\[
\geq -2(g^T (H + \lambda I)^{-1}g) \geq 4m(v) - \frac{1}{250\kappa^3L_3^2}
\]

117
(Note that we have $H + \lambda I > 0$.) This finishes the proof of Lemma 4.8.5.

We are now ready to provide the proofs of Claims 4.8.2 and 4.8.3.

**Proof of Claim 4.8.2** We derive that

$$m(h^*) \geq \frac{1}{2}(S_1 + S_2) - \frac{2(\lambda^*)^3}{3L_3^2} \geq \frac{1}{2}(S_1 + S_2) - \frac{16|\lambda_{\text{min}}|^3}{3L_3^2}$$

Above, $\circled{1}$ uses equation (4.8.4), inequality $\circled{2}$ follows because we have $\lambda_{\text{min}}(H) \leq -\frac{1}{\kappa}$ in the assumption and have $\lambda^* \leq -\lambda_{\text{min}}(H) + \frac{1}{\kappa}$ in the assumption of Case 2 of Lemma 4.7.2, inequality $\circled{3}$ uses Lemma 4.8.5 and Lemma 4.8.4 and inequality $\circled{4}$ uses (4.8.8). This finishes the proof of Claim 4.8.2.

**Proof of Claim 4.8.3** This time we lower bound $S_2$ slightly differently:

$$S_2 \geq -\frac{4}{\kappa L_3^2}(\lambda^*)^2 \geq -\frac{16}{\kappa^3 L_3^2} \tag{4.8.10}$$

where $\circled{1}$ comes from the second to last inequality from (4.8.7) and $\circled{2}$ comes from $\lambda^* \leq \lambda \leq -\lambda_{\text{min}}(H) + \frac{1}{\kappa} \leq \frac{2}{\kappa}$ using our assumption in Case 2 of Lemma 4.7.2.

Putting these together we get that

$$m(h^*) = \frac{1}{2}(S_1 + S_2) - \frac{2(\lambda^*)^3}{3L_3^2} \geq 2m(v) - \frac{1}{500\kappa^3 L_3^2} - \frac{15}{\kappa^3 L_3^2} \geq 2m(v) - \frac{16}{\kappa^3 L_3^2}.$$ 

Above, $\circled{1}$ comes from (4.8.4), $\circled{2}$ uses Lemma 4.8.5, lower bound (4.8.10) and $\frac{2(\lambda^*)^3}{3L_3^2} \leq \frac{16}{3\kappa^3 L_3^2}$.
4.8.2 Ensuring Condition (4.7.9)

Let’s first note that Lemma 4.4.1 implies that,

\[(\lambda^*)^3 \leq \frac{3L_3^2|m(h^*)|}{2} \leq \frac{L_3^{3/2}\varepsilon^{3/2}}{200}.\]  \hspace{1cm} (4.8.11)

Suppose Case 1 occurs. We have that

\[
\|v\| \leq \|(H + \lambda I)^{-1}g\| + \varepsilon \leq \frac{2\lambda + 2L_3\varepsilon}{L_3} + \frac{2\lambda^*}{L_3} + 5\varepsilon \leq \|h^*\| + \frac{1}{20\kappa L_3}.
\]

Above, inequalities 1 and 2 both use the assumptions of Case 1; inequality 3 uses the fact that \(\lambda^* \in [\lambda - L_3\varepsilon, \lambda + L_3\varepsilon]\) which again follows from the assumptions of Case 1 (see (4.8.1)); inequality 4 uses \(\|h^*\| = \frac{2\lambda^*}{L_3}\) from Lemma 4.4.1 as well as our assumption (4.7.6) on \(\varepsilon\).

As for the quantity \(\|\nabla m(v)\|\), we bound it as follows

\[
\|\nabla m(v)\| = \left\| g + Hv + \frac{L_3\|v\|}{2}v \right\| \leq \|g + (H + \lambda I)v\| + \lambda\|v\| + L_3\|v\|^2 \leq \|H + \lambda I\|\varepsilon + \lambda\|v\| + L_3\|v\|^2 \leq (L_2 + 2B)\varepsilon + \frac{\lambda(2\lambda + 3L_3\varepsilon) + (2\lambda + 3L_3\varepsilon)^2}{L_3}
\]

\[
= (L_2 + 2B)\varepsilon + \frac{6\lambda^2}{L_3} + 15\varepsilon\lambda + 9L_3\varepsilon^2 \leq \frac{6(\lambda^* + L_3\varepsilon)^2}{L_3} + (L_2 + 32B)\varepsilon + 9L_3\varepsilon^2 \leq \frac{6(\lambda^*)^2}{L_3} + (L_2 + 56B)\varepsilon + 15L_3\varepsilon^2 \leq \frac{\varepsilon}{4} + \frac{15}{\kappa^2L_3}.
\]

Above, inequality 1 uses triangle inequality; inequality 2 uses \(\|v + (H + \lambda I)^{-1}g\| \leq \varepsilon\); inequality 3 uses \(\|H + \lambda I\| \leq L_2 + 2B\) and \(L_3\|v\| \leq 2\lambda + 3L_3\varepsilon\) which comes from our upper bound on \(\|v\|\) above; 4 uses the fact that \(\lambda^* \in [\lambda - L_3\varepsilon, \lambda + L_3\varepsilon]\) which again follows from the assumptions of Case 1 (see (4.8.1)); inequality 5 uses \(\lambda^* \leq 2B\); and inequality 6 uses (4.8.11) together with our assumption (4.7.6) on \(\varepsilon\). We now consider the second case.
Suppose Case 2 occurs. We have that
\[ \|v\| \leq \|(H + \lambda I)^{-1}g\| + \bar{\varepsilon} \leq \frac{2\lambda}{L_3} + \bar{\varepsilon} \leq \frac{2(\lambda^* + 1/\kappa)}{L_3} + \bar{\varepsilon} \leq \|h^*\| + \frac{3}{\kappa L_3}. \] (4.8.12)

Above, inequalities 1 and 2 both use the assumptions of Case 2; inequality 3 uses \(\lambda \leq -\lambda_{\text{min}}(H) + 1/\kappa\) from our assumption of Case 2 as well as \(-\lambda_{\text{min}}(H) \leq \lambda^*\) which comes from Lemma 4.4.1, inequality 4 uses \(\|h^*\| = \frac{2\lambda^*}{L_3}\) from Lemma 4.4.1 as well as our assumption (4.7.6) on \(\bar{\varepsilon}\).

The quantity \(\|\nabla m(v)\|\) can be bounded in an analogous manner as Case 1:
\[ \|\nabla m(v)\| \leq \|H + \lambda I\|\bar{\varepsilon} + \lambda \|v\| + L_3 \|v\|^2 \leq (L_2 + 2B)\bar{\varepsilon} + \frac{\lambda(2\lambda + L_3\bar{\varepsilon}) + (2\lambda + L_3\bar{\varepsilon})^2}{L_3} \leq \frac{6\lambda^2}{L_3} + \frac{1}{10\kappa^2L_3} \lesssim \frac{6(\lambda^* + \frac{1}{\kappa})^2}{L_3} + \frac{1}{10\kappa^2L_3} \leq \frac{12(\lambda^*)^2}{L_3} + \frac{15}{\kappa^2L_3} \lesssim \frac{\varepsilon}{4} + \frac{15}{\kappa^2L_3}. \]

Above, inequality 1 uses our assumption (4.7.6) on \(\bar{\varepsilon}\); inequality 2 uses \(\lambda \leq \lambda^* + \frac{1}{\kappa}\) which appeared in (4.8.12); inequality 3 uses (4.8.11).

### 4.9 Proof of Lemma 4.4.1

We begin by proving a few lemmas that characterize the system of equations.

**Lemma 4.9.1.** Consider the following system of equations/inequalities in variables \(\lambda, h\):
\[ H + \lambda I \geq 0, \quad (H + \lambda I)h = -g, \quad \|h\| = \frac{2\lambda}{L_3}. \] (4.9.1)

The following statements hold for any solution \((\lambda', h')\) of the above system:

- There is a unique value \(\lambda'\) that satisfies the above equations. \(\lambda'\) is such that \(\lambda' \geq -\lambda_{\text{min}}(H).\)
• If $\lambda' > -\lambda_{\text{min}}(H)$, then the corresponding $h'$ is also unique and is given by $h' = -(H + \lambda I)^{-1}g$.

• If $\lambda' = -\lambda_{\text{min}}(H)$ then $g^\top v = 0$ for any vector $v$ belonging to the eigenspace corresponding to $\lambda_{\text{min}}(H)$. Subsequently we also have that the corresponding $h'$ is of the form

$$h' = -(H + \lambda I)^{+}g + \gamma v$$

for some $\gamma$ and $v$ in the lowest eigenspace of $H$.

**Proof of Lemma 4.9.1.** Note that $H + \lambda I \succeq 0$ ensures that for any solution $\lambda'$, we have $\lambda' \geq -\lambda_{\text{min}}(H)$. Furthermore, for any $\lambda' > -\lambda_{\text{min}}(H)$, the corresponding $h$ is uniquely defined by $h = (H + \lambda I)^{-1}g$ since $H + \lambda I$ is invertible. If indeed $\lambda' = -\lambda_{\text{min}}(H)$, then we have that the equation $(H - \lambda_{\text{min}}(H)I)h = -g$ has a solution. This implies that $g$ has no component in the null space of $H - \lambda_{\text{min}}(H)I$, or equivalently that it has no component in the eigenspace corresponding to $\lambda_{\text{min}}(H)$. We also have that every solution of $(H - \lambda_{\text{min}}(H)I)h = -g$ is necessarily of the form

$$h = -(H - \lambda_{\text{min}}I)^{+}g + \gamma v$$

for some $\gamma$ and $v$ in the lowest eigenspace of $H$.

We will now prove the uniqueness of $\lambda'$ by contradiction. Consider two distinct values of $\lambda_1, \lambda_2$ that satisfy the system (4.9.1). If both $\lambda_1, \lambda_2 > -\lambda_{\text{min}}(H)$ we get that

$$\| (H + \lambda_1 I)^{-1}g \| = \frac{2\lambda_1}{L_3} \quad \text{and} \quad \| (H + \lambda_2 I)^{-1}g \| = \frac{2\lambda_2}{L_3}.$$

Now note that $\| (H + \lambda I)^{-1}g \|$ is a strictly decreasing function over the domain $\lambda \in (-\lambda_{\text{min}}(H), \infty)$ and $\frac{2\lambda}{L_3}$ is strictly increasing over the same domain. Therefore the above two equations cannot be satisfied for two distinct $\lambda_1, \lambda_2 > -\lambda_{\text{min}}(H)$ which is a contradiction. Suppose now without loss of generality that $\lambda_1 = -\lambda_{\text{min}}(H)$. Then we have that the
corresponding solution is of the form

\[ h = -(H + \lambda I)^+ g + \gamma v \]

for some \( \gamma \) and \( v \) in the lowest eigenspace of \( H \) and \( g \) has no component in the lowest eigenspace of \( H \). It follows that \( \| (H - \lambda_{\text{min}}(H))^{-1} g \| \geq \| (H + \lambda I)^{-1} g \| \) for any \( \lambda > -\lambda_{\text{min}}(H) \). By a similar argument as in the first case, we can now see that the following conditions,

\[ \| (H + \lambda_1 I)^+ g + \gamma v_{\text{min}}(H) \| = \frac{2\lambda_1}{L_3} \quad \text{and} \quad \| (H + \lambda_2 I)^{-1} g \| = \frac{2\lambda_2}{L_3}, \]

cannot both be satisfied for \( \lambda_2 > \lambda_1 = -\lambda_{\text{min}}(H) \), giving us a contradiction. This finishes the proof of Lemma 4.9.1.

**Lemma 4.9.2.** Let \((\lambda, h)\) be a solution of the system (4.9.1). Then we have that

\[ m(h) = \frac{1}{2} g^\top (H + \lambda I)^+ g - \frac{2\lambda^3}{3L_3^2}. \]

**Proof of Lemma 4.9.2** By the definition of the system (4.9.1), any solution \( \lambda, h \) to the system should be such that there exists some \( \gamma \) such that

\[ h = (H + \lambda I)^+ g + \gamma v_0 \]

where \( v_0 \) is in the null space of \( H + \lambda I \) if it exists; otherwise \( \gamma = 0 \). This gives us the following:

\[
\begin{align*}
m(h) &= g^\top h + \frac{1}{2} h^\top H h + \frac{L_3}{6} \| h \|^3 \\
&= -\frac{1}{2} h^\top (H + \lambda I) h - \frac{\lambda}{2} \| h \|^2 + \frac{L_3}{6} \| h \|^3 \\
&= -\frac{1}{2} g^\top (H + \lambda I)^+ g - \frac{2\lambda^3}{3L_3^2}.
\end{align*}
\]
Equality 1 follows because \((H + \lambda I)h = -g\). Equality 2 follows because \(h = (H + \lambda I)g + \gamma v_0\) and \(\|h\| = \frac{2\lambda}{L_3}.

Lemma [4.4.1] \(h^*\) is a minimizer of \(m(h)\) if and only if there exists \(\lambda^* \geq 0\) such that

\[
H + \lambda^* I \geq 0, \quad (H + \lambda^* I)h^* = -g, \quad \|h^*\| = \frac{2\lambda^*}{L_3}.
\]

The objective value in this case is given by

\[
m(h^*) = -\frac{1}{2}g^\top(H + \lambda^* I)^+g - \frac{2(\lambda^*)^3}{3L_3^2} \leq 0.
\]

Proof of Lemma 4.4.1 We first compute that

\[
\nabla m(h) = g + Hh + \frac{L_3}{2}\|h\|\quad \text{and} \quad \nabla^2 m(h) = H + \frac{L_3}{2}\|h\|I + \frac{L_3}{2}\|h\|\left(\frac{h}{\|h\|}\right)^\top\left(\frac{h}{\|h\|}\right).
\]

For the forward direction, suppose \(h^*\) is a minimizer of \(m(h)\). Let \(\lambda^* = \frac{L_3}{2}\|h^*\|.\) Then, the necessary conditions \(\nabla m(h^*) = 0\) and \(\nabla^2 m(h^*) \succeq 0\) can be written as

\[
g + (H + \lambda^* I)h^* = 0 \quad \text{and} \quad w^\top\left(\begin{array}{c} H + \lambda^* I + \lambda^* \left(\frac{h^*}{\|h^*\|}\right) \\ \|h^*\| \end{array}\right) \left(\begin{array}{c} h^* \\ \|h^*\| \end{array}\right)^\top w \succeq 0, \forall w \in \mathbb{R}^n.
\]

(4.9.2)

From this we see \((H + \lambda^* I)h^* = -g\) and \(\|h^*\| = \frac{2\lambda^*}{L_3}\), and the only thing left to verify is \(H + \lambda^* I \succeq 0\).

Note that if \(h^* = 0\), then the second inequality in (4.9.2) directly implies \(H + \lambda^* I \succeq 0\). Thus, we only need to focus on \(h^* \neq 0\). We want to show that \(w^\top (H + \lambda^* I)w \geq 0\) for every \(w \in \mathbb{R}^d\). Now, if \(w^\top h^* = 0\) then this trivially follows from (4.9.2), so it suffices to focus on those \(w\) that satisfies \(w^\top h^* \neq 0\).

Since \(w\) and \(h^*\) are not orthogonal, there exists \(\gamma \in \mathbb{R}\setminus\{0\}\) such that \(\|h^* + \gamma w\| = \|h^*\|.\) (This can be done by squaring both sides and solving the linear system in \(\lambda\).) Squaring both
sides we have
\[(\gamma w)^\top h^* + \frac{\gamma^2\|w\|^2}{2} = 0. \tag{4.9.3}\]

Now we bound the difference

\[
m(h^* + \gamma w) - m(h^*) = g^\top ((h^* + \gamma w) - h^*) + \frac{(h^* + \gamma w)^\top H(h^* + \gamma w)}{2} - \frac{h^*Hh^*}{2}
\]

\[
= \frac{\lambda^*\gamma^2}{2}\|w\|^2 + (h^* - (h^* + \gamma w))^\top Hh^* + \frac{(h^* + \gamma w)^\top H(h^* + \gamma w)}{2} - \frac{h^*Hh^*}{2}
\]

\[
= \frac{\lambda^*\gamma^2}{2}\|w\|^2 + \frac{h^*Hh^*}{2} - (h^* + \gamma w)^\top Hh^* + \frac{(h^* + \gamma w)^\top H(h^* + \gamma w)}{2}
\]

\[
= \frac{\lambda^*\gamma^2}{2}\|w\|^2 + \frac{\gamma^2}{2}w^\top Hw = \frac{\gamma^2}{2}w^\top (H + \lambda^*I)w, \tag{4.9.4}
\]

where $\odot$ and $\circledast$ follow from (4.9.2) and (4.9.3), respectively. Since $h^*$ is a minimizer of $m(h)$, we immediately have

\[
m(h^* + \gamma w) - m(h^*) = \frac{\gamma^2}{2}w^\top (H + \lambda^*I)w \geq 0,
\]

and we conclude that $(H + \lambda^*I) \succeq 0$.

For the backward direction, we will make use of Lemma 4.9.1 and Lemma 4.9.2. First we note that the function $m(h)$ is continuous and bounded from below, and there exists at least one minimizer $h^*$. Suppose now there exists a $\lambda^*$ and a corresponding $h^*$ such that $(\lambda^*, h^*)$ is a solution to the system 4.9.1. The backward direction requires us to prove that $h^*$ must be a minimizer of $m(h)$. By Lemma 4.9.1 we get the following two cases.

We prove the backward direction by showing that the conditions in Equation 4.9.2 determine the minimizer up to its norm. To this end we will use Lemma 4.9.1 and Lemma 4.9.2.

First we note that the function $m(h)$ is continuous, bounded from below, and tends to $+\infty$ when $\|h\| \to \infty$, so there exists at least one minimizer $h^*$. 

124
Suppose now there exists a $\lambda^*$ and a corresponding $h^*$ such that $(\lambda^*, h^*)$ is a solution to the system (4.9.1). The backward direction requires us to prove that $h^*$ must be a minimizer of $m(h)$. By Lemma 4.9.1 we get the following two cases.

- If $\lambda^* > -\lambda_{\text{min}}(H)$ then $(\lambda^*, h^*)$ is the only solution to the system (4.9.1). By the proof of the forward direction we see that any minimizer of $m(h)$ must satisfy system (4.9.1) and therefore $h^*$ must be the minimizer.

- If above is not the case, then $\lambda^* = -\lambda_{\text{min}}(H)$. Let $h'$ be any minimizer of $m(h)$. Lemma 4.9.1 and the proof of the forward direction ensures that $(\lambda^*, h')$ also satisfies the system (4.9.1). By Lemma 4.9.2 we get $m(h^*) = m(h')$ and therefore $h^*$ is a minimizer too.

\qed
Chapter 5

Adaptive Cubic Regularization Methods
for Riemannian Manifolds

5.1 Introduction

Adaptive regularization with cubics (ARC) is an iterative algorithm used to solve unconstrained optimization problems of the form

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f: \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Given any initial iterate $x_0 \in \mathbb{R}^n$, under some regularity conditions on $f$, ARC produces an iterate $x_k$ with small gradient, namely, $\|\nabla f(x_k)\| \leq \varepsilon$, in at most $O(1/\varepsilon^{1.5})$ iterations [46] [29]. This is optimal compared to the complexity of steepest descent and classical trust-region methods, which has lead to a lot of interest in this method recently.
In this chapter, we generalize ARC to optimization problems on manifolds, that is, problems of the form
\[
\min_{x \in \mathcal{M}} f(x),
\]
where \( \mathcal{M} \) is a given Riemannian manifold and \( f : \mathcal{M} \to \mathbb{R} \) is a (sufficiently smooth) cost function \([66, 135, 59, 3]\). Building upon the existing literature about the Euclidean case, we generalize the worst-case iteration complexity analysis of ARC to manifolds, obtaining essentially the same guarantees. In particular, with the appropriate assumptions discussed in Section 5.3, we find that \( \varepsilon \)-critical points of \( f \) on \( \mathcal{M} \) can be computed in \( O(1/\varepsilon^{1.5}) \) iterations. Subsequent sections investigate the new assumptions, with the aim to provide insight and user-friendly sufficient conditions for them to hold. Furthermore, we implement the algorithm within the Manopt framework \([35]\), and we close with numerical comparisons to existing solvers, in particular the closely related Riemannian trust-region method (RTR) \([2]\).

\subsection*{5.1.1 Main Results}

We give here an informal statement of the main result, which is stated precisely in Corollary 5.3.8. Assumptions A5.3.1 to A5.3.4 are detailed in Section 5.3, while the algorithm itself is described in Section 5.2. Retractions are also defined in that section.

**Theorem 5.1.1** (Informal). Let \( \mathcal{M} \) be a Riemannian manifold equipped with a retraction \( R \), and let the cost function \( f : \mathcal{M} \to \mathbb{R} \) be twice continuously differentiable. Assume the following:

\begin{itemize}
  \item \textbf{A5.3.1} \( f \) is lower bounded;
  \item \textbf{A5.3.2} The pullbacks \( f \circ R_x \) uniformly satisfy a type of second-order Lipschitz regularity;
  \item \textbf{A5.3.3} The subproblem solver meets certain weak requirements at each iteration; and
\end{itemize}
A5.3.4 The differential of the retraction is well behaved.

Then, for an arbitrary $x_0 \in \mathcal{M}$, provided $\varepsilon$ is small enough, Algorithm 6 produces an iterate $x_k \in \mathcal{M}$ such that $f(x_k) \leq f(x_0)$ and $\left\| \nabla f(x_k) \right\| \leq \varepsilon$ with $k = O(1/\varepsilon^{1.5})$, where $\nabla f$ is the Riemannian gradient of $f$ and $\left\| \cdot \right\|$ is the Riemannian norm.

In the Euclidean case with the canonical retraction, the assumptions reduce to the classical ones and we recover the known result exactly [29]. In further sections, we study these assumptions in detail and show they can hold for large classes of problems. In particular, if $\mathcal{M}$ is compact, $f$ is smooth and $R$ is second-order nice (Definition 5.4.3), all our assumptions hold.

5.1.2 Related Work

Numerous algorithms for unconstrained optimization have been generalized to manifolds [3], including gradient descent, nonlinear conjugate gradients, stochastic gradients [33, 148], BFGS [122], Newton’s method [4] and trust-regions [2]. ARC in particular was extended to manifolds in the PhD thesis of Qi [121]. There, asymptotic convergence analyses are proposed, in the same spirit as the analyses presented in [3] for other methods. In contrast, we here favor a global convergence analysis with explicit bounds on iteration complexity, in the same vein as those produced for Riemannian gradient descent and RTR in [34]. In Definition 5.4.3 below, we formulate second-order assumptions on the retraction to disentangle the requirements on $f$ from those on the retraction. These are related to the assumptions and discussions in [122] Lemma 6, Remark 3.

Our complexity analysis is rooted in prior work for the Euclidean case by Cartis et al. [46] and Birgin et al. [29]. Complexity lower bounds given by [44, 47, 40] show that the bounds in [115, 46, 29] are optimal for the appropriate class of functions. A variant of ARC that is closely related to trust-region methods was presented in [58].

Zhang and Zhang [149] recently proposed a convergence analysis of a cubically regularized method on manifolds. Their analysis focuses on compact submanifolds of a Euclidean space.
Algorithm 6 Riemannian adaptive regularization with cubics (ARC)

1: **Parameters:** \( \theta > 0, \varsigma_{\text{min}} > 0, 0 < \eta_1 \leq \eta_2 < 1, 0 < \gamma_1 < 1 < \gamma_2 < \gamma_3 \) and \( \varepsilon > 0 \)
2: **Input:** \( x_0 \in \mathcal{M}, \varsigma_0 \geq \varsigma_{\text{min}} \)
3: **Init:** \( k \leftarrow 0 \)
4: while \( \| \nabla f(x_k) \| > \varepsilon \) do
5: Compute a step \( s_k \in T_{x_k} \mathcal{M} \) as an approximate minimizer of the model \( m_k (5.2.4) \). See A5.5.3 for sufficient requirements to meet.
6: Compute the regularized ratio between actual improvement and model improvement:
   \[
   \rho_k = \frac{f(x_k) - f(R_{x_k}(s_k))}{m_k(0) - m_k(s_k) + \frac{s_k}{3} \| s_k \|^3}. \quad (5.2.1)
   \]
7: If \( \rho_k \geq \eta_1 \), accept the step: \( x_{k+1} = R_{x_k}(s_k) \). Otherwise, reject it: \( x_{k+1} = x_k \).
8: Update the regularization parameter:
   \[
   \varsigma_{k+1} \in \begin{cases} 
   [\max(\varsigma_{\text{min}}, \gamma_1 s_k), s_k] & \text{if } \rho_k \geq \eta_2 \\
   [s_k, \gamma_2 s_k] & \text{if } \rho_k \in [\eta_1, \eta_2] \\
   [\gamma_2 s_k, \gamma_3 s_k] & \text{if } \rho_k < \eta_1
   \end{cases} \quad (5.2.2)
   \]
9: \( k \leftarrow k + 1 \)
10: **end while**
11: **Output:** \( x_k \in \mathcal{M} \) such that \( f(x_k) \leq f(x_0) \) and \( \| \nabla f(x_k) \| \leq \varepsilon \) (under assumptions.)

and uses a fixed regularization parameter \( \varsigma \) (which must be set properly by the user). They further study second-order stationarity conditions and local convergence in that context.

### 5.2 ARC on Manifolds

ARC on manifolds is listed as Algorithm 6. It is a direct adaptation from [46, 29]. Like many other optimization algorithms, its generalization to manifolds relies on a chosen retraction [134, 3]. For some \( x \in \mathcal{M} \), let \( T_x \mathcal{M} \) denote the tangent space at \( x \). Intuitively, a retraction \( R \) on a manifold provides a means to move away from \( x \) along a tangent direction \( s \in T_x \mathcal{M} \) while remaining on the manifold, producing \( R_x(s) \in \mathcal{M} \).

**Definition 5.2.1** (Retraction [3 Def. 4.1.1]). A retraction on a manifold \( \mathcal{M} \) is a smooth mapping \( R \) from the tangent bundle \( T \mathcal{M} \) (that is, the set of pairs \( (x, s) \) where \( x \in \mathcal{M} \) and
s ∈ T_x M) to M with the following properties. Let R_x : T_x M → M denote the restriction of R to T_x M. Then,

(i) \( R_x(0) = x \), where 0 is the zero vector in T_x M; and

(ii) The differential of R_x at 0, \( DR_x(0) \), is the identity map.

Around \( t = 0 \), retraction curves \( t \mapsto R_x(ts) \) agree up to first order with geodesics passing through \( x \) with velocity \( s \). For the special case where \( \mathcal{M} \) is a linear space, the canonical retraction is \( R_x(s) = x + s \). For the unit sphere, a typical retraction is \( R_x(s) = \frac{x + s}{\|x + s\|_2} \).

Importantly, the retraction \( R \) chosen to optimize over a particular manifold \( \mathcal{M} \) is part of the algorithm specification. For a given cost function \( f \) and a specified retraction \( R \), at iterate \( x_k \), we define the pullback of the cost function on the tangent space \( T_{x_k} \mathcal{M} \):

\[
\hat{f}_k = f \circ R_{x_k} : T_{x_k} \mathcal{M} \to \mathbb{R}.
\] (5.2.3)

At each iteration, we define a model \( m_k : T_{x_k} \mathcal{M} \to \mathbb{R} \) for the pullback, obtained as a truncated second-order Taylor expansion with cubic regularization:

\[
m_k(s) = \hat{f}_k(0) + \langle s, \nabla \hat{f}_k(0) \rangle + \frac{1}{2} \langle s, \nabla^2 \hat{f}_k(0)[s] \rangle + \frac{s_k}{3} \|s\|^3.
\] (5.2.4)

Here, \( \langle \cdot, \cdot \rangle \) denotes the Riemannian metric at a point \( x \in \mathcal{M} \), that is, the inner product associated to the tangent space \( T_x \mathcal{M} \), and we often omit the subscript \( x \) when the root point is clear from context. Likewise, \( \|s\| = \sqrt{\langle s, s \rangle} \) is the norm of \( s \) associated to the Riemannian metric, in this case at \( x_k \).

At iteration \( k \), a subproblem solver is used to approximately minimize the model \( m_k \), producing a tentative step \( s_k \). The quality of this step is evaluated by computing \( \rho_k \) (5.2.1):

the ratio of actual to anticipated cost improvement. If \( \rho_k \geq \eta_1 \), we accept the tentative step and set \( x_{k+1} = R_{x_k}(s_k) \): such steps are called successful. Otherwise, we reject the step and
set $x_{k+1} = x_k$: these steps are unsuccessful. Depending on the value of $\rho_k$, we may also choose to change the regularization parameter $\varsigma_k$.

For a given $K$, we let

$$S = \{ k \in 0, \ldots, K - 1 : \rho_k \geq \eta_1 \}$$

(5.2.5)

denote the set of successful iterations among the first $K$. In contrast, $U$ indexes unsuccessful iterations: it is the complement of $S$ in $0, \ldots, K - 1$. Within $S$, we further identify very successful iterations, for which $\rho_k \geq \eta_2$; for such iterations, not only is the step accepted, but the regularization parameter $\varsigma_k$ is (usually) decreased.

### 5.3 First-order Analysis

We here propose a set of four assumptions under which ARC on manifolds produces an iterate with gradient less than $\varepsilon$ in $O(1/\varepsilon^{1.5})$ iterations. In subsequent sections, we discuss sufficient conditions for the assumptions to be met. The first assumption is that the cost function $f$ is lower bounded. This is necessary to ensure existence of points on $\mathcal{M}$ with arbitrarily small gradient (though it does not guarantee existence of a point where the gradient is exactly zero.)

**Assumption 5.3.1.** There exists $f_{\text{low}}$ such that $f(x) \geq f_{\text{low}}$ for all $x \in \mathcal{M}$.

Our second assumption regards regularity of the pullbacks $\hat{f}_k$. This differs from the standard assumptions because of the role of the retraction. It is similar in spirit to the regularity assumptions in [34]. When optimizing over $\mathbb{R}^n$ with the usual retraction $R_x(s) = x + s$, the conditions below are satisfied if the Hessian of $f$ is $L$-Lipschitz continuous [46]. In Section 5.4, we derive sufficient conditions on $f$ and the retraction for this assumption to hold on manifolds.
Assumption 5.3.2. The cost function $f$ is twice continuously differentiable. Furthermore, there exists a constant $L$ such that, at each iteration $k$, for the tentative step $s_k$ selected by the subproblem solver,

$$
\left| \hat{f}_k(s_k) - \left[ \hat{f}_k(0) + \langle s_k, \nabla \hat{f}_k(0) \rangle + \frac{1}{2} \langle s_k, \nabla^2 \hat{f}_k(0)[s_k] \rangle \right] \right| \leq \frac{L}{6} \| s_k \|^3, \quad (5.3.1)
$$

and

$$
\left\| \nabla \hat{f}_k(s_k) - \left[ \nabla \hat{f}_k(0) + \nabla^2 \hat{f}_k(0)[s_k] \right] \right\| \leq \frac{L}{2} \| s_k \|^2. \quad (5.3.2)
$$

The third assumption requires the subproblem solver to make sufficient progress, though notice that it does not require much—the particular requirements follow the lead of Birgin et al. [29]. In particular, there is no need for $s_k$ to be approximately optimal for the (possibly non-convex) subproblem. The subproblem is posed in a linear space (the tangent space at $x_k$). Thus, any standard technique from the Euclidean case carries over in principle. We give a brief discussion in Section 5.5.

Assumption 5.3.3. At each iteration $k$, the subproblem solver produces a tentative step $s_k \in T_{x_k} M$ such that

$$
m_k(s_k) < m_k(0), \quad \text{and} \quad \| \nabla m_k(s_k) \| \leq \theta \| s_k \|^2. \quad (5.3.3)
$$

The fourth assumption differs significantly from the Euclidean treatment of ARC. It occurs for the following reason. On the one hand, as long as ARC does not terminate, $\| \nabla f(x_{k+1}) \| > \varepsilon$. On the other hand, we show in the proof below that $\| \nabla \hat{f}_k(s_k) \| \leq (L/2 + \theta + \varsigma_k) \| s_k \|^2 \quad (5.3.11)$. In the Euclidean case ($M = \mathbb{R}^n$ and $R_x(s) = x + s$), for a successful step, $x_{k+1} = x_k + s_k$ and it is easily verified that $\nabla f(x_{k+1}) = \nabla \hat{f}_k(s_k)$. From there, we may conclude that, so long as ARC does not stop, successful steps are somewhat large, and this leads to the desired conclusion. In contrast, on manifolds, the gradient of $f$ at
\(x_{k+1}\) and the gradient of the pullback \(\hat{f}_k\) at \(s_k\) do not coincide (they are not even in the same tangent space). They are, however, related by a linear operator. Indeed, for all \(\hat{s} \in T_{x_k}M\), by definition of gradient, directional derivative and using the chain rule:

\[
\langle \nabla \hat{f}_k(s), \hat{s} \rangle_{x_k} = D\hat{f}_k(s)[\hat{s}]
\]

\[
= Df(R_{x_k}(s)) [DR_{x_k}(s)[\hat{s}]]
\]

\[
= \langle \nabla f(R_{x_k}(s)), DR_{x_k}(s)[\hat{s}] \rangle_{R_{x_k}(s)}
\]

\[
= \langle (DR_{x_k}(s))^* [\nabla f(R_{x_k}(s))], \hat{s} \rangle_{x_k}. \tag{5.3.4}
\]

Above, \(DR_{x_k}(s)\) is a linear operator mapping vectors from the tangent space at \(x_k\) to the tangent space at \(R_{x_k}(s)\), and the star indicates we take the adjoint of that operator. By identification, we find (this also appears in [122, p599]):

\[
\nabla \hat{f}_k(s) = (DR_{x_k}(s))^* [\nabla f(R_{x_k}(s))]. \tag{5.3.5}
\]

Since \(DR_{x_k}(s)\) is a “square” operator (it is a map between two subspaces of the same dimension), it shares the same singular values with its adjoint. With \(\sigma_{\text{min}}\) extracting the smallest singular value of an operator, by definition,

\[
\|\nabla \hat{f}_k(s)\| \geq \sigma_{\text{min}} (DR_{x_k}(s)) \|\nabla f(R_{x_k}(s))\|. \tag{5.3.6}
\]

For our purpose, it is important that this least singular value remain bounded away from zero. This is only a concern for small steps (as large successful steps provide sufficient improvement for other reasons.) We materialize this discussion in the following assumption. For the case of \(\mathbb{R}^n\) with the canonical retraction, \(DR_{x}(s)\) is an isometry and one can set \(a = +\infty\) and \(b = 1\). We secure this property in Section 5.6 for a large family of manifolds and retractions.
Assumption 5.3.4. There exist constants $a > 0$ and $b > 0$ such that, at each successful iteration $k$, if $\|s_k\| \leq a$, then

$$\sigma_{\min}(DR_{x_k}(s_k)) \geq b.$$  \hfill (5.3.7)

(The constant $a$ is allowed to be $+\infty$, while $b$ is necessarily at most 1.)

We start the analysis with two supporting lemmas which follow the standard Euclidean analysis almost exactly. The first lemma establishes the regularization parameter $\varsigma_k$ does not grow unbounded.

Lemma 5.3.5 ([29, Lem. 2.2]). Under A5.3.2 and A5.3.3, the regularization parameter remains bounded:

$$\forall k, \quad \varsigma_k \leq \varsigma_{\text{max}} \triangleq \max \left( \varsigma_0, \frac{L\gamma_3}{2(1-\eta_2)} \right).$$ \hfill (5.3.8)

Proof of Lemma 5.3.5 Using the definition of $\rho_k$ (5.2.1), $m_k(0) = f(x_k)$ (5.2.4) and $m_k(0) - m_k(s_k) \geq 0$ by A5.3.3

$$|\rho_k - 1| = \left| \frac{f(x_k) - f(R_{x_k}(s_k))}{m_k(0) - m_k(s_k) + \frac{\varsigma_k}{3} \|s_k\|^3} - 1 \right| \leq \left| \frac{m_k(s_k) - \frac{\varsigma_k}{3} \|s_k\|^3 - f(R_{x_k}(s_k))}{\frac{\varsigma_k}{3} \|s_k\|^3} \right|.

Owing to A5.3.2, the numerator is upper bounded by $(L/6)\|s_k\|^3$. Hence, $|\rho_k - 1| \leq \frac{L}{2\varsigma_k}$. If $\varsigma_k \geq \frac{L}{2(1-\eta_2)}$, then $1 - \rho_k \leq |\rho_k - 1| \leq 1 - \eta_2$ so that $\rho_k \geq \eta_2$, meaning step $k$ is very successful. The regularization mechanism (5.2.2) then ensures $\varsigma_{k+1} \leq \varsigma_k$. Thus, $\varsigma_{k+1}$ may exceed $\varsigma_k$ only if $\varsigma_k < \frac{L}{2(1-\eta_2)}$, in which case it can grow at most to $\frac{L\gamma_3}{2(1-\eta_2)}$, but cannot grow beyond that level in ulterior iterations.

Conditioned on the conclusions of the lemma above, the lemma below shows the total number of iterations of ARC is bounded above in terms of the number of successful iterations.
Lemma 5.3.6 ([46, Thm. 2.1]). If $s_{\max} > 0$ is such that $s_k \leq s_{\max}$ for all $k$ (as provided by Lemma 5.3.5), then the number $|S|$ of successful iterations among $0, \ldots, K - 1$ satisfies

$$K \leq \frac{1}{\log(\gamma_2)} \left[ |S| \log \left( \frac{\gamma_2}{\gamma_1} \right) + \log \left( \frac{s_{\max}}{s_0} \right) \right].$$

Proof of Lemma 5.3.6. This result follows from the regularization parameter update mechanism of ARC: it is not affected by the fact we here work on a manifold. For sake of completeness, we provide the standard proof. Following (5.2.2), for $k \in S$, $s_{k+1} \geq \gamma_1 s_k$, while for $k \in U$, $s_{k+1} \geq \gamma_2 s_k$. Thus, by induction, $s_K \geq s_0 \gamma_1^{\frac{|S|}{|U|}} \gamma_2^{\frac{|U|}{|S|}}$. By assumption, $s_K \leq s_{\max}$ so that

$$\log \left( \frac{s_{\max}}{s_0} \right) \geq |S| \log(\gamma_1) + |U| \log(\gamma_2) = |S| \log(\gamma_1/\gamma_2) + K \log(\gamma_2),$$

where we also used $|S| + |U| = K$. Isolating $K$ using $\gamma_2 > 1 > \gamma_1$ allows to conclude.

The following theorem is the pivotal argument to establish a worst-case iteration complexity of ARC to reach approximate first-order critical points. The proof follows [29, Thm. 2.5], up to two distinctions. First, since the proof requires relating the gradient of the model $m_k$ at $s_k$ and the gradient of the cost function at $R_{x_k}(s_k)$, we use (5.3.6) which indicates there is a need to control the least singular values of the differential of the retraction at $(x_k, s_k)$. This is secured via A5.3.4 (where the constants $a$ and $b$ come from). Second, in order to apply A5.3.4, we introduce a distinction between long and short successful steps. Long successful steps produce sufficient decrease in the cost function on their own, while for short successful steps we invoke A5.3.4.

Theorem 5.3.7. Under A5.3.1, A5.3.2, A5.3.3 and A5.3.4, for an arbitrary $x_0 \in \mathcal{M}$, let $x_0, \ldots, x_K$ be iterates produced by ARC. If $0 < \varepsilon \leq \frac{\alpha^2 (L + \theta + s_{\max})}{b}$ and the number of
successful iterations among $0, \ldots, K - 1$ (5.2.5) obeys

$$|S| > \frac{3(f(x_0) - f_{low})}{\eta_1 \varsigma_{\min}} \left(\frac{L}{2} + \theta + \varsigma_{\max}\right)^{1.5} \frac{1}{\varepsilon^{1.5}},$$

then $f(x_k) \leq f(x_0)$ and $\|\nabla f(x_k)\| \leq \varepsilon$ for some $k$ in $0, \ldots, K$. On the other hand, if

$\varepsilon > \frac{a^2(\frac{L}{2} + \theta + \varsigma_{\max})}{b}$, then such a point $x_k$ is produced in particular if $|S| > \frac{3(f(x_0) - f_{low})}{\eta_1 \varsigma_{\min}} \frac{1}{a^3}$.

Combining Lemmas 5.3.5 and 5.3.6 and Theorem 5.3.7 yields the main result: a bound on the total number of iterations it may take ARC to produce an approximate critical point on a manifold.

**Corollary 5.3.8.** Under A5.3.1, A5.3.2, A5.3.3 and A5.3.4, for an arbitrary $x_0 \in \mathcal{M}$, let $x_0, \ldots, x_K$ be iterates produced by ARC. If $0 < \varepsilon \leq \frac{a^2(\frac{L}{2} + \theta + \varsigma_{\max})}{b}$ and

$$K > \left(1 + \frac{|\log(\gamma_1)|}{\log(\gamma_2)}\right) \frac{3(f(x_0) - f_{low})}{\eta_1 \varsigma_{\min}} \left(\frac{L}{2} + \theta + \varsigma_{\max}\right)^{1.5} \frac{1}{\varepsilon^{1.5}} + \frac{1}{\log(\gamma_2)} \log \left(\frac{\varsigma_{\max}}{\varsigma_0}\right),$$

then $f(x_k) \leq f(x_0)$ and $\|\nabla f(x_k)\| \leq \varepsilon$ for some $k$ in $0, \ldots, K$.

Comparing with [29], the known result in the Euclidean case is exactly the same with $b = 1$ (as is allowed in that case by our analysis), and up to the fact that we now have a condition that $\varepsilon$ must be sufficiently small (which also disappears here in the Euclidean case as $a$ can be taken arbitrarily large). We finish this section by providing the proof of Theorem 5.3.7.

**Proof of Theorem 5.3.7** For some $\varepsilon > 0$, let $K$ be such that none of $x_0, \ldots, x_K$ have gradient norm at or below $\varepsilon$. By definition, step $k$ is successful if $\rho_k \geq \eta_1$ (5.2.1), that is, if

$$f(x_k) - f(\operatorname{R}_{x_k}(s_k)) \geq \eta_1 \left(m_k(0) - m_k(s_k) + \frac{s_k}{3}\|s_k\|^3\right).$$

136
In this case, \( x_{k+1} = R_{x_k}(s_k) \). Thus, using A5.3.3 for successful steps,

\[
f(x_k) - f(x_{k+1}) = \frac{\eta_1 s_k}{3} \| s_k \|^3 \geq \frac{\eta_1 \min s}{3} \| s_k \|^3.
\]

On the other hand, for unsuccessful steps, \( x_{k+1} = x_k \). Using A5.3.1, a classical telescoping sum argument yields

\[
f(x_0) - f_{\text{low}} \geq f(x_0) - f(x_K) = \sum_{k \in S} f(x_k) - f(x_{k+1}) \geq \frac{\eta_1 \min s}{3} \sum_{k \in S} \| s_k \|^3.
\]

We further partition successful steps in two subsets, namely long and short steps. The constant \( c > 0 \) will be chosen later.

\[
S_{\text{long}} = \{ k \in S : \| s_k \| > c \sqrt{\varepsilon} \}, \quad \text{and} \quad S_{\text{short}} = S \setminus S_{\text{long}}.
\]

Successful long steps each produce an improvement in the cost function proportional to \( \varepsilon^{1.5} \):

\[
\frac{3(f(x_0) - f_{\text{low}})}{\eta_1 \min s} \geq \sum_{k \in S_{\text{long}}} \| s_k \|^3 + \sum_{k \in S_{\text{short}}} \| s_k \|^3 \geq |S_{\text{long}}| c^3 \varepsilon^{1.5} + \sum_{k \in S_{\text{short}}} \| s_k \|^3. \tag{5.3.9}
\]

Our goal now is to show the same for successful short steps (\( \| s_k \| \leq c \sqrt{\varepsilon} \)). For all steps, a triangle inequality yields

\[
\| \nabla \hat{f}_k(s_k) \| \leq \| \nabla \hat{f}_k(s_k) - \left[ \nabla \hat{f}_k(0) + \nabla^2 \hat{f}_k(0)[s_k] \right] \| + \| \nabla \hat{f}_k(0) + \nabla^2 \hat{f}_k(0)[s_k] \|. \tag{5.3.10}
\]

Owing to A5.3.2, the first term is upper bounded by \( \frac{L}{2} \| s_k \|^2 \). For the second term, notice that

\[
\nabla m_k(s_k) = \nabla \hat{f}_k(0) + \nabla^2 \hat{f}_k(0)[s_k] + \varsigma_k \| s_k \| s_k.
\]

137
so that

$$\left\| \nabla \hat{f}_k(0) + \nabla^2 \hat{f}_k(0)[s_k] \right\| = \left\| \nabla m_k(s_k) - s_k \right\| s_k \leq \left\| \nabla m_k(s_k) \right\| + s_k \leq s_k^2.$$

Following A5.3.3 and Lemma 5.3.5, this is upper bounded by $(\theta + \varsigma_{\text{max}}) s_k^2$. Combining these findings with (5.3.10) yields, for all steps,

$$\left\| \nabla \hat{f}_k(s_k) \right\| \leq \left( \frac{L}{2} + \theta + \varsigma_{\text{max}} \right) s_k^2. \tag{5.3.11}$$

On the other hand, relation (5.3.6) states

$$\left\| \nabla \hat{f}_k(s_k) \right\| \geq \sigma_{\text{min}}(\text{DR}_{x_k}(s_k)) \left\| \nabla f(x_k) \right\|. \tag{5.3.12}$$

If step $k$ is successful, then $R_{x_k}(s_k) = x_{k+1}$. Combine this with the first assumption made in this proof: $\|\nabla f(x_{k+1})\| > \varepsilon$. Furthermore, owing to A5.3.4, there exist constants $a, b > 0$ such that $\sigma_{\text{min}}(\text{DR}_{x_k}(s_k)) \geq b$, provided $\|s_k\| < a$. To ensure this for short steps, we set $c$ such that $a = c\sqrt{\varepsilon}$. Combining this discussion with (5.3.11) and (5.3.12), for successful short steps,

$$\left( \frac{L}{2} + \theta + \varsigma_{\text{max}} \right) s_k^2 > b\varepsilon.$$

Plugging this inequality in (5.3.9) and using $c = a/\sqrt{\varepsilon}$ shows

$$\frac{3(f(x_0) - f_{\text{low}})}{\eta_{1\text{min}}} \geq |S_{\text{long}}| \|c^3\varepsilon^{1.5} + |S_{\text{short}}| \left( \frac{b}{\frac{L}{2} + \theta + \varsigma_{\text{max}}} \right)^{1.5} \varepsilon^{1.5}
\geq |S| \min \left( \frac{a^3}{\varepsilon^{1.5}}, \left( \frac{b}{\frac{L}{2} + \theta + \varsigma_{\text{max}}} \right)^{1.5} \varepsilon^{1.5} \right).$$

138
If \( \varepsilon \leq \frac{a^2(\frac{L}{2} + \theta + \varsigma_{\text{max}})}{b} \), then the minimum evaluates to the second input, and we find

\[
|S| \leq \frac{3(f(x_0) - f_{\text{low}})}{\eta_1 \varsigma_{\text{min}}} \left( \frac{\frac{L}{2} + \theta + \varsigma_{\text{max}}}{b} \right)^{1.5} \frac{1}{\varepsilon^{1.5}}.
\]

If \( |S| \) exceeds the right hand side, then it must be that \( \nabla f(x_k) \leq \varepsilon \) for some \( k \leq K \). Likewise, if \( \varepsilon > \frac{a^2(\frac{L}{2} + \theta + \varsigma_{\text{max}})}{b} \), then the minimum evaluates to the first input and we find \( |S| \leq \frac{3(f(x_0) - f_{\text{low}})}{\eta_1 \varsigma_{\text{min}}} \frac{1}{a^3} \) is necessary for all gradients to be larger than \( \varepsilon \).

5.4 Regularity Assumptions

The regularity assumption A5.3.2 pertains to the pullbacks \( f \circ R_{x_k} \). Each is a function from a Euclidean space \( T_{x_k}M \) to \( \mathbb{R} \), so that standard calculus applies. Since the retraction is smooth by definition, pullbacks are as smooth as \( f \). This leads to the following simple fact.

Lemma 5.4.1. Assume \( f : M \to \mathbb{R} \) is twice continuously differentiable. If there exists \( L \) such that, for all \( (x, s) \) among the sequence of iterates and steps \( \{(x_0, s_0), (x_1, s_1), \ldots \} \) produced by Algorithm 6 with \( \hat{f} = f \circ R_x \), it holds that

\[
\left\| \nabla^2 \hat{f}(ts) - \nabla^2 \hat{f}(0) \right\|_{\text{op}} \leq tL\|s\| \quad (5.4.1)
\]

for all \( t \in [0, 1] \), then A5.3.2 holds.

Proof of Lemma 5.4.1. By the fundamental theorem of calculus applied to \( \hat{f} \) and \( \nabla \hat{f} \),

\[
\hat{f}(s) - \left[ \hat{f}(0) + \langle s, \hat{f}(0) \rangle + \frac{1}{2} \langle s, \nabla^2 \hat{f}(0)[s] \rangle \right] = \int_0^1 \int_0^t \left[ \nabla^2 \hat{f}(t_1 t_2 s) - \nabla^2 \hat{f}(0) \right][s], s \, dt_1 dt_2,
\]

\[
\nabla \hat{f}(s) - \left[ \nabla \hat{f}(0) + \nabla^2 \hat{f}(0)[s] \right] = \int_0^1 \left[ \nabla^2 \hat{f}(ts) - \nabla^2 \hat{f}(0) \right][s] \, dt.
\]
Taking norms on both sides, by a triangular inequality to pass the norm through the integral and integrating respectively \( t_1^2 t_2 \) and \( t \), we find using (5.4.1) that

\[
\left| \dot{f}(s) - \left[ \dot{f}(0) + \langle s, \dot{f}(0) \rangle + \frac{1}{2} \langle s, \nabla^2 \dot{f}(0)[s] \rangle \right] \right| \leq \frac{1}{6} L \|s\|^3, \quad \text{and}
\]

\[
\left\| \nabla \dot{f}(s) - \left[ \nabla \dot{f}(0) + \nabla^2 \dot{f}(0)[s] \right] \right\| \leq \frac{1}{2} L \|s\|^2. \tag{5.4.2}
\]

We call (5.4.1) a *Lipschitz-type* assumption on \( \nabla^2 \dot{f} \) because it compares the Hessians at \( ts \) and 0, rather than comparing them at two arbitrary points on the tangent space. On the other hand, we require this to hold on several tangent spaces with the same constant \( L \).

Thus, to understand A5.3.2 one way is to understand the Hessian of the pullback at points which are not the origin. Lemma 5.4.2 provides the necessary identities. The Hessian formula might be new.

The statement of the lemma as well as its proof require some tools from Riemannian geometry—see [118, pp59–67]. Specifically, we let \( \nabla \) denote the Levi–Civita connection on \( M \) (not to be confused with \( \nabla \) and \( \nabla^2 \) which denote gradient and Hessian of functions on linear spaces, such as \( \dot{f} \)). With this notation, the Riemannian Hessian [3, Def. 5.5.1] is defined by \( \text{Hess} f = \nabla \nabla f \). Furthermore, \( \frac{D}{dt} \) denotes the covariant derivative of vector fields along curves on \( M \). With this notation, given a smooth curve \( \gamma: \mathbb{R} \to M \), the intrinsic acceleration is defined as \( \gamma''(t) = \frac{D^2}{dt^2} \gamma(t) \). For example, for a Riemannian submanifold of a Euclidean space, \( \gamma''(t) \) is obtained by orthogonal projection of the classical acceleration of \( \gamma \) in the embedding space to the tangent space at \( \gamma(t) \). Geodesics have zero intrinsic acceleration.

**Lemma 5.4.2.** Given \( f: M \to \mathbb{R} \) smooth and \( x \in M \), the gradient and Hessian of the pullback \( \hat{f} = f \circ R_x \) at \( s \in T_x M \) are given by

\[
\nabla \hat{f}(s) = T_s^* \nabla f(R_x(s)), \tag{5.4.2}
\]

\[
\nabla^2 \hat{f}(s) = T_s^* \circ \text{Hess} f(R_x(s)) \circ T_s + W_s, \tag{5.4.3}
\]

140
where

\[ T_s = DR_x(s) : T_xM \rightarrow T_{R_x(s)}M \]  \hspace{1cm} (5.4.4)

is linear and \( W_s \) is a symmetric linear operator on \( T_xM \) defined through polarization by

\[ \langle W_s[s], \dot{s} \rangle = \langle \nabla f(R_x(s)), \gamma''(0) \rangle, \]  \hspace{1cm} (5.4.5)

with \( \gamma''(0) \in T_{R_x(s)}M \) the intrinsic acceleration on \( M \) of \( \gamma(t) = R_x(s + t\dot{s}) \) at \( t = 0 \).

**Proof of Lemma 5.4.2** For an arbitrary \( \dot{s} \in T_xM \), consider the curve \( \gamma \) as above. Let \( g = f \circ \gamma : \mathbb{R} \rightarrow \mathbb{R} \). We compute the derivatives of \( g \) in two different ways. On the one hand, \( g(t) = \hat{f}(s + t\dot{s}) \) so that

\[
\begin{align*}
g'(t) &= D\hat{f}(s + t\dot{s})[\dot{s}] = \langle \nabla \hat{f}(s + t\dot{s}), \dot{s} \rangle, \\
g''(t) &= \left\langle \frac{d}{dt} \nabla \hat{f}(s + t\dot{s}), \dot{s} \right\rangle = \langle \nabla^2 \hat{f}(s + t\dot{s})[\dot{s}], \dot{s} \rangle.
\end{align*}
\]

On the other hand, \( g(t) = f(\gamma(t)) \) so that, using properties of \( \frac{d}{dt} \) [118, pp59–67]:

\[
\begin{align*}
g'(t) &= Df(\gamma(t))[\gamma'(t)] = \langle \nabla f(\gamma(t)), \gamma'(t) \rangle, \\
g''(t) &= \frac{d}{dt} \langle (\nabla f \circ \gamma)(t), \gamma'(t) \rangle \\
&= \langle \nabla_{\gamma'(t)} \nabla f, \gamma'(t) \rangle + \left\langle (\nabla f \circ \gamma)(t), \frac{d}{dt} \gamma'(t) \right\rangle \\
&= \langle \text{Hess} f(\gamma(t))[\gamma'(t)], \gamma'(t) \rangle + \langle \nabla f(\gamma(t)), \gamma''(t) \rangle.
\end{align*}
\]
Equating the different identities for $g'(t)$ and $g''(t)$ at $t = 0$, using $\gamma'(0) = T_s \dot{s}$, we find that for all $\dot{s} \in T_x M$,

$$\langle \nabla \hat{f}(s), \dot{s} \rangle = \langle \nabla f(R_x(s)), T_s \dot{s} \rangle,$$

$$\langle \nabla^2 \hat{f}(s)[\dot{s}], \dot{s} \rangle = \langle \text{Hess} f(R_x(s))[T_s \dot{s}], T_s \dot{s} \rangle + \langle \nabla f(R_x(s)), \gamma''(0) \rangle.$$

The last term, $\langle \nabla f(R_x(s)), \gamma''(0) \rangle$, is the difference of two quadratic forms in $\dot{s}$, so that it is itself a quadratic form in $\dot{s}$, justifying the definition of $W_s$ through polarization. The announced identities follow by identification.

Combining Lemmas 5.4.1 and 5.4.2 allows to guarantee $A5.3.2$ holds in particular if $M$ is compact, $f$ is smooth and the retraction is nice enough. We formalize this in the definition below and in Theorem 5.4.4. We stress that these are sufficient but not necessary conditions.

**Definition 5.4.3 (Second-order nice retraction).** A retraction $R$ on $M$ is second-order nice if there exist constants $c_1, c_2, c_3$ such that, for all $x \in M$ and $s, \dot{s} \in T_x M$:

1. $\|\gamma''(0)\| \leq c_1 \|s\| \|\dot{s}\|^2$ where $\gamma(t) = R_x(s + t\dot{s})$;

2. $\|T_s\|_{op} \leq c_2$ where $T_s$ is as in (5.4.4); and

3. $\forall t \in [0, 1], \|D_t U(t)\| \leq c_3\|s\|\|\dot{s}\|$, where $U(t) = T_{ts} \dot{s}$.

**Theorem 5.4.4.** Let $f : M \to \mathbb{R}$ be smooth. Assume the retraction is second-order nice as per Definition 5.4.3. Let $\{(x_0, s_0), (x_1, s_1), \ldots\}$ be the sequence of points and steps generated by Algorithm 6. Consider

$$\mathcal{N} = \bigcup_k \{R_{x_k}(ts_k) : t \in [0, 1]\},$$

(5.4.6)

the subset of $M$ obtained by collecting all curves generated by retracted steps (both accepted and rejected). If the closure of $\mathcal{N}$ is a compact subset of $M$, then $A5.3.2$ is satisfied.
Note that the closure of $\mathcal{N}$ is compact in particular if $\mathcal{M}$ is compact, since closed subsets of compact sets are compact. See the proof for a bound on $L$.

Before providing a proof for Theorem 5.4.4 we make some comments on Definition 5.4.3. Notice that the choice $s = 0$ in the first condition indicates such retractions are, in particular, second-order retractions in the sense of [3, S5]. For $\mathcal{M}$ a Euclidean space, the canonical retraction $R_x(s) = x + s$ is second-order nice with $c_1 = c_3 = 0$ and $c_2 = 1$. The classical retraction on the sphere (as shown by the proposition below) is also second-order nice, with small constants $c_1, c_2, c_3$. We expect this to be the case for many usual retractions.

**Proposition 5.4.5.** For the unit sphere $\mathcal{M} = \{x \in \mathbb{R}^n : \|x\| = 1\}$, the retraction $R_x(s) = \frac{x + s}{\|x + s\|}$ is second-order nice with $c_1 = 2$, $c_2 = \frac{3}{2}$, $c_3 = \frac{4\sqrt{3}}{9}$.

We end this section by first providing a proof of Theorem 5.4.4 and then the proof of Proposition 5.4.5.

**Proof of Theorem 5.4.4.** Let $\{{(x_0, s_0), (x_1, s_1), \ldots}\}$ be the sequence of points and steps generated by Algorithm 6. For some $k$ and $\bar{t} \in [0, 1]$, let $(x, s) = (x_k, \bar{t}s_k)$ and define the pullback $\hat{f} = f \circ R_x$. Combine the expression for the Hessian of the pullback (5.4.3) with (5.4.1) to get:

$$\|\nabla^2 \hat{f}(s) - \nabla^2 \hat{f}(0)\|_{op} \leq T_s^* \circ \text{Hess}_x(s) \circ T_s - \text{Hess}_x(0)\|_{op} + \|W_s - W_0\|_{op}.$$  

By definition of $W_s$ (5.4.5), using the first condition on the retraction, we find that $W_0 = 0$ and

$$\|W_s\|_{op} = \max_{s \in T_x \mathcal{M}} \langle W_s[s], \hat{s} \rangle \leq \|\nabla f(R_x(s))\| \cdot \max_{s \in T_x \mathcal{M}} \|\gamma''(0)\| \leq c_1 G\|s\|.$$

143
where $G = \max_{y \in \mathcal{N}} \| \nabla f(y) \|$ is finite by compactness of $\mathcal{N}$ (the closure of $\mathcal{N}$) and continuity of the gradient norm. Thus, it remains to show that

$$\| T^*_s \circ \text{Hess}(R_x(s)) \circ T_s - \text{Hess}(x) \|_{\text{op}} \leq c' \| s \|$$

for some constant $c'$. For an arbitrary $\dot{s} \in T_x \mathcal{M}$, owing to $f$ being smooth, it holds that

$$\left\langle \left[ T^*_s \circ \text{Hess}(R_x(s)) \circ T_s - \text{Hess}(x) \right] \dot{s}, \dot{s} \right\rangle = \int_0^1 \frac{d}{dt} \left\langle T^*_s \circ \text{Hess}(R_x(ts)) \circ T_{ts}[\dot{s}], \dot{s} \right\rangle dt.$$

We aim to upper bound the above by $c' \| s \| \| \dot{s} \|^2$. Consider the curve $\gamma(t) = R_x(ts)$ and a tangent vector field $U(t) = T_{ts} \dot{s}$ along $\gamma$. Then, define

$$h(t) = \left\langle T^*_s \circ \text{Hess}(\gamma(t)) \circ T_{ts}[\dot{s}], \dot{s} \right\rangle$$

$$= \left\langle \text{Hess}(\gamma(t))[T_{ts} \dot{s}], T_{ts} \dot{s} \right\rangle$$

$$= \left\langle \text{Hess}(\gamma(t))[U(t)], U(t) \right\rangle.$$

The integrand above is the derivative of the real function $h$:

$$h'(t) = \frac{d}{dt} \left\langle \text{Hess}(\gamma(t))[U(t)], U(t) \right\rangle$$

$$= \left\langle \frac{D}{dt} \left[ \text{Hess}(\gamma(t))[U(t)] \right], U(t) \right\rangle + \left\langle \text{Hess}(\gamma(t))[U(t)], \frac{D}{dt} U(t) \right\rangle$$

$$= \left\langle (\nabla_{\gamma(t)} \text{Hess}(f)) [U(t)], U(t) \right\rangle + 2 \left\langle \text{Hess}(\gamma(t))[U(t)], U'(t) \right\rangle,$$

where $U'(t) = \frac{D}{dt} U(t)$ and we used that the Hessian is symmetric. Here, $\nabla_{\gamma(t)} \text{Hess}(f)$ is the Levi–Civita derivative of the Hessian tensor field at $\gamma(t)$ along $\gamma'(t)$—see [55 Def. 5.7] for the notion of derivative of a tensor field. For every $t$, the latter is a symmetric linear operator on the tangent space at $\gamma(t)$. By Cauchy–Schwarz,

$$|h'(t)| \leq \| \nabla_{\gamma(t)} \text{Hess}(f) \|_{\text{op}} \| U(t) \|^2 + 2 \| \text{Hess}(\gamma(t)) \|_{\text{op}} \| U(t) \| \| U'(t) \|.$$
By compactness of $\mathcal{N}$ and continuity of the Hessian, we can define

$$ H = \max_{y \in \mathcal{N}} \|\text{Hess} f(y)\|_{\text{op}}. $$

By linearity of the connection $\nabla$, if $\gamma'(t) \neq 0$,

$$ \nabla_{\gamma'(t)} \text{Hess} f = \|\gamma'(t)\| \cdot \nabla \frac{\gamma'(t)}{\|\gamma'(t)\|} \text{Hess} f. $$

Furthermore, $\gamma'(t) = T_{ts}s$, with norm bounded by the second assumption on the retraction: $\|\gamma'(t)\| \leq c_2\|s\|$. Thus, in all cases, by compactness of $\mathcal{N}$ and continuity of the function $v \rightarrow \nabla_v \text{Hess} f$ on the tangent bundle $T\mathcal{M}$,

$$ \|\nabla_{\gamma(t)} \text{Hess} f\|_{\text{op}} \leq c_2\|s\| \cdot \max_{y \in \mathcal{N}, v \in T_y \mathcal{M}} \|\nabla_v \text{Hess} f\|_{\text{op}}. $$

Of course, $\|U(t)\| \leq c_2\|\dot{s}\|$. Finally, we bound $\|U'(t)\|$ using the third property of the retraction: $\|U'(t)\| \leq c_3\|s\|\|\dot{s}\|$. Overall,

$$ \left| \left\langle [T_s^* \circ \text{Hess} f(R_x(s)) \circ T_s - \text{Hess} f(x)][\dot{s}], \dot{s} \right\rangle \right| \leq \int_0^1 |h'(t)| dt \leq [c_2^2 H' + 2c_2 c_3 H] \|s\|\|\dot{s}\|^2. $$

It follows from Lemma 5.4.1 that A5.3.2 holds with $L = c_1 G + 2c_2 c_3 H + c_2^2 H'$. 

Proof of Proposition 5.4.5: We start with property 2. With $R_x(s) = \frac{x + s}{\sqrt{1 + |s|^2}}$, it is easy to derive

$$ T_s \dot{s} = \text{DR}_x(s)[\dot{s}] = \left[ \frac{1}{\sqrt{1 + \|s\|^2}} I_n - \frac{1}{\sqrt{1 + \|s\|^2}}(x + s) s^T \right] \dot{s}. \quad (5.4.7) $$
Since $T_s$ is an operator on $T_x \mathcal{M} \subset \mathbb{R}^n$, its operator norm is bounded by the operator norm of the matrix appearing between brackets. Hence,

$$\|T_s\|_{op} \leq \frac{1}{\sqrt{1 + \|s\|^2}} \sqrt{1 + \|s\|^2 \|x + s\|} \leq 1 + \frac{\|s\|}{1 + \|s\|^2} \leq \frac{3}{2}.$$

For property 3, consider $U(t) = T_s \dot{s}$ and

$$U'(t) = \frac{d}{dt} U(t) = \text{Proj}_{\gamma(t)} \frac{d}{dt} U(t),$$

where $\text{Proj}_y(v) = v - y(y^tv)$ is the orthogonal projector to $T_y \mathcal{M}$ and $\gamma(t) = R_x(ts)$. Define $g(t) = \frac{1}{\sqrt{1 + t^2\|s\|^2}}$. Then, from (5.4.7), we have

$$U(t) = [g(t)I_n - tg(t)^3(x + ts)s^\top] \dot{s}. \quad (5.4.8)$$

This is easily differentiated in the classical sense:

$$\frac{d}{dt} U(t) = [g'(t)I_n - (tg(t)^3)'(x + ts)s^\top - tg(t)^3ss^\top] \dot{s}.$$  

The projection at $\gamma(t)$ zeros out the middle term, as it is parallel to $x + ts$. This offers a simple expression for $U'(t)$, where in the last equality we use $g'(t) = -tg(t)^3\|s\|^2$:

$$U'(t) = \text{Proj}_{\gamma(t)} \left([g'(t)I_n - tg(t)^3ss^\top] \dot{s}\right) = -tg(t)^3 \cdot \text{Proj}_{\gamma(t)} \left([\|s\|^2I_n + ss^\top] \dot{s}\right).$$

The norm can only decrease after projection, so that, for $t \in [0, 1]$,

$$\|U'(t)\| \leq 2tg(t)^3\|s\|^2\|\dot{s}\|.$$
Let \( h(t) = 2t g(t)^3 \|s\|^2 = \frac{2t^2 \|s\|^2}{(1 + t^2 \|s\|^2)^{3/2}} \). For \( s = 0 \), \( h \) is identically zero. Otherwise, \( h \) attains its maximum \( h \left( t = \frac{1}{\sqrt{2} \|s\|} \right) = \frac{4\sqrt{3}}{9} \|s\| \). It follows that \( \|U''(t)\| \leq c_3 \|s\| \|\dot{s}\| \) for all \( t \in [0, 1] \) with \( c_3 = \frac{4\sqrt{3}}{9} \).

Finally, we establish property 1. Given \( s, \dot{s} \in X, M \), consider the curve \( \gamma(t) = R_x(s + t\dot{s}) \). Simple calculations yield:

\[
\gamma'(t) = \frac{d}{dt} \gamma(t) = \frac{1}{\sqrt{1 + \|s + t\dot{s}\|^2}} \left[ \dot{s} - \langle \dot{s}, \gamma(t) \rangle \gamma(t) \right] = \frac{1}{\sqrt{1 + \|s + t\dot{s}\|^2}} \text{Proj}_{\gamma(t)} \dot{s}.
\]

This is indeed in the tangent space at \( \gamma(t) \). The classical derivative of \( \gamma'(t) \) is given by

\[
\frac{d}{dt} \gamma'(t) = -\frac{1}{\sqrt{1 + \|s + t\dot{s}\|^2}} \left[ \langle \dot{s}, \gamma'(t) \rangle \gamma(t) + \langle \dot{s}, \gamma(t) \rangle \gamma'(t) + \frac{\langle s + t\dot{s}, \dot{s} \rangle}{1 + \|s + t\dot{s}\|^2} \text{Proj}_{\gamma(t)} \dot{s} \right]
\]

\[
= -\frac{1}{\sqrt{1 + \|s + t\dot{s}\|^2}} \left[ \langle \dot{s}, \gamma'(t) \rangle \gamma(t) + 2 \frac{\langle s + t\dot{s}, \dot{s} \rangle}{1 + \|s + t\dot{s}\|^2} \text{Proj}_{\gamma(t)} \dot{s} \right],
\]

where we used (5.4.9) and orthogonality of \( x \) and \( \dot{s} \) in \( \langle \gamma(t), \dot{s} \rangle = \frac{1}{\sqrt{1 + \|s + t\dot{s}\|^2}} \langle x + s + t\dot{s}, \dot{s} \rangle \).

The acceleration of \( \gamma \) is \( \gamma''(t) = \frac{D}{dt} \gamma'(t) = \text{Proj}_{\gamma(t)} \left( \frac{d}{dt} \gamma'(t) \right) \). The first term vanishes after projection, while the second term is unchanged. Overall,

\[
\gamma''(t) = -\frac{2 \langle s + t\dot{s}, \dot{s} \rangle}{\sqrt{1 + \|s + t\dot{s}\|^2}} \text{Proj}_{\gamma(t)} \dot{s} = -\frac{2 \langle \gamma(t), \dot{s} \rangle}{1 + \|s + t\dot{s}\|^2} \text{Proj}_{\gamma(t)} \dot{s}.
\]

In particular, \( \gamma''(0) = -2 \frac{\langle s, \dot{s} \rangle}{\sqrt{1 + \|s\|^2}} \text{Proj}_{\gamma(0)} \dot{s} \), so that \( \|\gamma''(0)\| \leq 2 \min(\|s\|, 0.4) \|\dot{s}\|^2 \) and the property holds with \( c_1 = 2 \).

\[\square\]

### 5.5 Solving the Subproblem

In this section we discuss an algorithm for the subproblem solver that guarantees satisfaction of (5.3.3). Given a vector space \( X \) of dimension \( n \) with an inner product \( \langle \cdot, \cdot \rangle \) (and associated
norm $\| \cdot \|$), an element $g \in X$, a self-adjoint linear operator $H : X \to X$ and a real $\sigma > 0$, define the function $m : X \to \mathbb{R}$ as

$$ m(s) = \langle g, s \rangle + \frac{1}{2} \langle s, H(s) \rangle + \frac{\sigma}{3} \| s \|^3. \tag{5.5.1} $$

We wish to obtain an element $s \in X$ such that $m(s) < m(0)$ and

$$ \| \nabla m(s) \| \leq \theta \| s \|^2. \tag{5.5.2} $$

Note that A5.3.3 requires solving exactly the above problem where, at iteration $k$, we set $X = T_{x_k}M$ with the Riemannian metric at $x_k$, $g = \nabla \hat{f}_k(0)$, $H = \nabla^2 \hat{f}_k(0)$ and $\sigma = \sigma_k$.

We note that while it is possible to extend the algorithm FastCubic described in Chapter 4 to this general scenario, in this section we describe an alternate algorithm based on the Lanczos method which might be useful in settings when the dimension is not very large. Since the focus of this chapter is not to bound the running time of the sub-problem solver we do not provide such bounds here.

**Lanczos Method.** Certainly, a global minimizer of (5.5.1) meets our requirements. Such a minimizer can be computed, but known procedures for this task involve an eigendecomposition of $H$, which may be expensive [46]. Instead, motivated by the method proposed in [46], we use the Lanczos method [137, Lect. 36]. Formally, the Lanczos method iteratively produces a sequence of orthonormal vectors $\{q_1, \ldots, q_n\}$ and a symmetric tridiagonal matrix $T$ of size $n$ such that

$$ q_1 = \frac{g}{\| g \|}, \quad \text{and} \quad \forall i, j \in 1 \ldots n, \ T_{ij} = \langle q_i, H(q_j) \rangle. \tag{5.5.3} $$

Producing the $k$ first vectors requires $k$ calls to $H$. The Lanczos method guarantees that, for all $k$, vectors $\{q_1, \ldots, q_k\}$ form an orthonormal basis for a subspace which contains
the $k$-dimensional Krylov subspace spanned by the vectors $\{g, H(g), \ldots, H^{k-1}(g)\}$, where $H^k$ represents the $k$-times composition of $H$. Consider $m(s)$ (5.5.1) restricted to such a subspace:

$$\forall y \in \mathbb{R}^k, \text{ with } s = \sum_{i=1}^{k} y_i q_i, \quad m(s) = \langle g, y_1 q_1 \rangle + \frac{1}{2} \sum_{i,j=1}^{k} y_i y_j \langle q_i, H(q_j) \rangle + \frac{\sigma}{3} \|y\|_2^3$$

$$= y_1 \|g\| + \frac{1}{2} y^T T_k y + \frac{\sigma}{3} \|y\|_2^3,$$

(5.5.4)

where $\| \cdot \|_2$ is the 2-norm over $\mathbb{R}^k$ and $T_k$ is the $k \times k$ principal submatrix of $T$. This cubic in $y$ can be minimized efficiently using the explicit procedure alluded to above, specifically because $T_k$ is tridiagonal: it is inexpensive to eigendecompose it. Furthermore, since the Lanczos basis is constructed incrementally, we can minimize the restricted cubic at $k = 1$, check the stopping criterion (5.5.2), and proceed to $k = 2$ only if necessary, etc. The hope (borne out in experiments) is that the algorithm stops well before $k$ reaches $n$ (at which point it necessarily succeeds.) In this way, we limit the number of calls to operator $H$, which is typically the most expensive part of the process.

**Checking the stopping criterion.** Importantly, the process described involves only linear combinations and inner products of vectors in $\mathcal{X}$, and calls to $H$ as an operator (as opposed to requiring a matrix representation of operator $H$, for example). This is particularly well suited for optimization on manifolds, where tangent vectors are usually not represented in particular coordinates, and the Hessian is only available through function calls.

We draw attention to a technical point. Upon minimizing (5.5.4), we obtain a vector $y \in \mathbb{R}^k$. To check the stopping criterion (5.5.2), we must compute $\| \nabla m(s) \|$, where $s = \sum_{i=1}^{k} y_i q_i$. Since

$$\nabla m(s) = g + H(s) + \sigma s \|s\|,$$  

(5.5.5)
one approach is to compute $s$ (that is, form the linear combination of $q'_i$’s) and apply $H$ to $s$: both these operations may be expensive in high dimension. An alternative (which features in Algorithm 7) is to recognize that $\nabla m(s)$ lies in the subspace spanned by $\{q_1, \ldots, q_{k+1}\}$ (if $k < n$). Indeed, using all properties in (5.5.3),

$$\nabla m(s) = \|g\|q_1 + \sum_{i=1}^{k+1} (T_{1:k+1,1:k} y)q_i + \sigma \|y\|^2 \sum_{i=1}^k y_i q_i,$$

(5.5.6)

where $T_{a:b,c:d}$ is the submatrix of $T$ containing rows $\{a, \ldots, b\}$ and columns $\{c, \ldots, d\}$. This expression gives a direct way to compute $\|\nabla m(s)\|$ simply by running the Lanczos iteration one step ahead. This modification reduces the number of calls to $H$ by a factor of two, and postpones forming the vector $s$ until the algorithm terminates.

**Algorithm 7** Lanczos-based cubic model subsolver

1: **Parameters:** $\theta, \sigma > 0$, vector space $\mathcal{X}$ with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$
2: **Input:** $g \in \mathcal{X}$ nonzero, a self-adjoint linear operator $H : \mathcal{X} \rightarrow \mathcal{X}$
3: Set $k = 1$
4: Obtain $q_1, T_1$ via a Lanczos iteration (5.5.3)
5: Solve $y^{(1)} = \text{argmin } y \in \mathbb{R} \|g\|y + \frac{1}{2} T_1 y^2 + \frac{1}{3} \sigma |y|^3$
6: Obtain $q_2, T_2$ via a Lanczos iteration (5.5.3)
7: Compute $\|\nabla m(s_1)\|$ via (5.5.6), where $s_1 = y^{(1)} q_1$
8: while $\|\nabla m(s_k)\| > \theta \|s_k\|^2$ do
9: Set $k = k + 1$
10: Solve $y^{(k)} = \text{argmin } y \in \mathbb{R}^k \|g\|y + \frac{1}{2} y^\top T_k y + \frac{1}{3} \sigma \|y\|^3$
11: Obtain $q_{k+1}, T_{k+1}$ via a Lanczos iteration (5.5.3)
12: Compute $\|\nabla m(s_k)\|$ via (5.5.6), where $s_k = \sum_{i=1}^k y_i^{(k)} q_i$
13: end while
14: **Output:** $s_k \in \mathcal{X}$

### 5.6 Controlling the Differentiated Retraction

As described in Section 5.3, the worst-case running time for ARC (Theorem 5.3.7) depends upon the differential of the retraction $R$ through $A_{5.3.4}$. Specifically, this assumption
involves constants $a, b$ which control $\sigma_{\min}(\text{DR}_{x_k}(s_k))$. In this section we establish (existence of) bounds on the values of $a$ and $b$ for both a popular manifold-and-retraction pair, and for a large class of manifolds and retractions. Specifically, we consider the following cases.

1. **Stiefel manifolds.** For the Stiefel manifold $\text{St}(n, p) = \{X \in \mathbb{R}^{n \times p} : X^\top X = I_p\}$ we explicitly control $(a, b)$ for the popular Q-factor retraction (Gram–Schmidt orthonormalization). Special cases include the sphere ($p = 1$) and the orthogonal group ($p = n$).

2. **Manifolds with bounded sectional curvature.** For such manifolds we control $(a, b)$ for the case of the exponential retraction (geodesics). Important special cases include the Euclidean case (flat manifold), manifolds with negative curvature, and compact manifolds. For compact manifolds, we further show existence of appropriate values $(a, b)$ for arbitrary retractions.

### 5.6.1 Stiefel Manifold

In this part we consider the Stiefel manifold $\text{St}(n, p)$ as a Riemannian submanifold of $\mathbb{R}^{n \times p}$ equipped with the usual inner product $\langle A, B \rangle = \text{Tr}(A^\top B)$ and the Q-factor retraction $R$ defined by:

$$\forall X \in \text{St}(n, p), \forall S \in T_X \mathcal{M}, \quad R_X(S) = Q,$$

where $X + S = QR$ is a thin QR-decomposition, i.e., $Q \in \mathbb{R}^{n \times p}$ is obtained by Gram–Schmidt orthonormalization of the columns of $X + S$. We prove the following lemma.

**Lemma 5.6.1.** For the Stiefel manifold $\text{St}(n, p)$ and the Q-factor retraction $R$ defined by (5.6.1),

$$\forall X \in \text{St}(n, p), \forall S \in T_X \mathcal{M}, \quad \sigma_{\min}(\text{DR}_X(S)) \geq 1 - 3\|S\|_F - \frac{1}{2}\|S\|_F^2$$
where $\| \cdot \|_F$ represents the Frobenius norm. Moreover, in the special case of $p = 1$ (i.e., the unit sphere in $\mathbb{R}^n$), the retraction reduces to $R_x(s) = \frac{x+s}{\|x+s\|}$ and we have that

$$\forall x \in St(n,1), \forall s \in T_xM, \quad \sigma_{\min}(DR_x(s)) = \frac{1}{1 + \|s\|_2^2}.$$ 

**Corollary 5.6.2.** If $M = St(n,p)$ and the retraction $R$ is given by (5.6.1), then for any $a > 0$, define $b = 1 - 3a - \frac{1}{2}a^2$. If $b$ is positive, then A5.3.4 holds with these $a$ and $b$. Moreover, for the sphere ($p = 1$), we have that A5.3.4 is satisfied for any $a > 0$ and $b = \frac{1}{1+a^2}$.

Upon inspecting Corollary 5.3.8, the takeaway is as follows: if the targeted tolerance $\varepsilon$ is close to zero, then $a$ can be chosen close to zero, and as a result $b$ can be chosen close to one. Thus, in this scenario, $(a, b)$ has only limited impact on the worst-case convergence rate. We finish this subsection with the proof of Lemma 5.6.1.

**Proof of Lemma 5.6.1** Let $X \in St(n,p)$ and $S \in T_XSt(n,p) = \{ \dot{X} \in \mathbb{R}^{n \times p} : \dot{X}^TX + X^T\dot{X} = 0 \}$ be fixed. Define $Q, R$ as the thin $QR$-decomposition of $X + S$, that is, $Q$ is an $n \times p$ matrix with orthonormal columns and $R$ is a $p \times p$ upper triangular matrix with nonnegative diagonal entries. By definition (5.6.1) we have that $R_X(S) = Q$.

For a matrix $M$, define $\text{tril}(M)$ as the lower triangular portion of the matrix $M$, that is, $\text{tril}(M)_{ij} = M_{ij}$ if $i \geq j$ and 0 otherwise. Further define $\rho_{\text{skew}}(M)$ as

$$\rho_{\text{skew}}(M) \triangleq \text{tril}(M) - \text{tril}(M)^\top.$$ 

As derived in [3] Ex. 8.1.5] (see also the erratum for the reference) we have a formula for the directional derivative of the retraction along any $Z \in T_XSt(n,p)$:

$$DR_X(S)[Z] = Q\rho_{\text{skew}}(Q^TZR^{-1}) + (I - QQ^\top)ZR^{-1}. \quad (5.6.2)$$
We first confirm that \( R \) is always invertible. To see this, note that \( S \) being tangent at \( X \) means \( S^\top X + X^\top S = 0 \) and therefore

\[
R^\top R = (X + S)^\top (X + S) = X^\top X + X^\top S + S^\top X + S^\top S = I_p + S^\top S, \tag{5.6.3}
\]

which shows \( R \) is invertible. Moreover the above expression also implies that:

\[
\sigma_k(R) = \sigma_k(X + S) = \sqrt{\lambda_k((X + S)^\top (X + S))} = \sqrt{1 + \lambda_k(S^\top S)} = \sqrt{1 + \sigma_k(S)^2},
\]

where \( \sigma_k(M) \) represents the \( k \)th singular value of \( M \) and \( \lambda_k \) likewise extracts the \( k \)th eigenvalue (in decreasing order for symmetric matrices). In particular we have that

\[
\sigma_{\min}(R^{-1}) = \frac{1}{\sqrt{1 + \sigma_{\max}(S)^2}} \geq \frac{1}{\sqrt{1 + \|S\|_F^2}} \geq 1 - \frac{1}{2}\|S\|_F^2,
\]

\[
\sigma_{\max}(R^{-1}) = \frac{1}{\sqrt{1 + \sigma_{\min}(S)^2}} \leq 1. \tag{5.6.4}
\]

Further note that since \( QR = X + S \), we have that \( Q = (X + S)R^{-1} \) and therefore

\[
Q^\top Z R^{-1} = (R^{-1})^\top (X + S)^\top Z R^{-1}
= (R^{-1})^\top X^\top Z R^{-1} + (R^{-1})^\top S^\top Z R^{-1}.
\]

The first term above is always skew-symmetric since \( Z \) is tangent at \( X \), so that \( X^\top Z + Z^\top X = 0 \). Furthermore, for any skew-symmetric matrix \( M \), \( \rho_{\text{skew}}(M) = M \). Therefore, we have that

\[
\text{DR}_X(S)[Z] = Q \rho_{\text{skew}}(Q^\top Z R^{-1}) + (I - QQ^\top)Z R^{-1}
= Q \left( \rho_{\text{skew}}(Q^\top Z R^{-1}) - Q^\top Z R^{-1} \right) + Z R^{-1}
= Q \left( \rho_{\text{skew}}((R^{-1})^\top S^\top Z R^{-1}) - (R^{-1})^\top S^\top Z R^{-1} \right) + Z R^{-1}, \tag{5.6.5}
\]
where in the last step we used \( XR^{-1} - Q = -SR^{-1} \). Further note that for any matrix \( M \) of size \( p \times p \),

\[
\|Q(\rho_{\text{skew}}(M) - M)\|_F = \|\text{tril}(M) - \text{tril}(M)^\top - M\|_F \leq 3\|M\|_F. \tag{5.6.6}
\]

Therefore we have that,

\[
\|\text{DR}_X(S)[Z]\|_F \geq \|ZR^{-1}\|_F - 3\|(R^{-1})^TS^TZR^{-1}\|_F \\
\geq \|Z\|_F \left(\sigma_{\text{min}}(R^{-1}) - 3\sigma_{\text{max}}(R^{-1})^2\sigma_{\text{max}}(S)\right), \tag{5.6.7}
\]

where we have used \( \|A\|_F\sigma_{\text{min}}(B) \leq \|AB\|_F \leq \|A\|_F\sigma_{\text{max}}(B) \) multiple times. Using the bounds on the singular values of \( R^{-1} \) (derived in (5.6.4)) we get that

\[
\|\text{DR}_X(S)[Z]\|_F \geq \|Z\|_F \left(1 - \frac{1}{2}\|S\|_F^2 - 3\|S\|_F\right).
\]

Since this holds for all tangent vectors \( Z \), we get that

\[
\sigma_{\text{min}}(\text{DR}_X(S)) \geq 1 - 3\|S\|_F - \frac{1}{2}\|S\|_F^2.
\]

To prove a better bound for the case of \( p = 1 \) (the sphere), we improve the analysis of the expression derived in (5.6.5). Note that for \( p = 1 \), the matrix inside the \( \rho_{\text{skew}} \) operator is a scalar, whose skew-symmetric part is necessarily zero. Also note that \( Q \) is a single column matrix with value \( \frac{x + s}{\|x + s\|_2} \) and \( R = \|x + s\|_2 \). Also, \( X^TSX^TZ = 0 \) since \( S, Z \) are tangent.
Therefore,

\[
\text{DR}_X(S)[Z] = ZR^{-1} - Q(R^{-1})^T S^T R^{-1} \\
= \frac{1}{\|x + s\|^2} \left( z - \frac{s^T z}{1 + \|s\|^2} (x + s) \right) \\
= \frac{1}{\|x + s\|^2} \left( z - \frac{s^T z}{1 + \|s\|^2} s - \frac{s^T z}{1 + \|s\|^2} x \right).
\]

Since \( x \) is orthogonal to \( s \) and \( z \),

\[
\|\text{DR}_x(s)[z]\|_2^2 = \frac{1}{1 + \|s\|^2} \left( \|z\|^2 - \frac{(s^T z)^2}{1 + \|s\|^2} \right) \\
= \frac{1}{1 + \|s\|^2} \left( \frac{\|z\|^2}{1 + \|s\|^2} - \frac{(s^T z)^2}{1 + \|s\|^2} \right) \\
\geq \frac{\|z\|^2}{1 + \|s\|^2} \left( 1 - \frac{\|s\|^2}{1 + \|s\|^2} \right) \\
= \frac{\|z\|^2}{(1 + \|s\|^2)^2}.
\]

The worst-case scenario is achieved when \( z \) and \( s \) are aligned. Overall, we get

\[
\|\text{DR}_x(s)[z]\|_2 \geq \frac{\|z\|^2}{1 + \|s\|^2},
\]

which establishes the bound for the sphere.

\[\square\]

5.6.2 Manifolds with Bounded Sectional Curvature - Exponential Retraction

In this part we consider manifolds with (upper) bounded sectional curvature. The first case we consider is that of the exponential retraction. Let \( \mathcal{M} \) be a Riemannian manifold with the associated exponential map \( \text{Exp}: T_x \mathcal{M} \to \mathcal{M} \). Given a point \( x \in \mathcal{M} \) and a tangent
vector \( s \in T_xM \), consider the linear operator \( \text{DExp}_x(s) : T_xM \to T_{\text{Exp}_x(s)}M \). The following lemma follows directly from the Jacobi field comparison theorem in Riemannian geometry [91, Thm. 11.2].

**Lemma 5.6.3.** Assume all sectional curvatures of \( M \) are bounded above by \( C \):

- If \( C \leq 0 \), then \( \sigma_{\min}(\text{DExp}_x(s)) = 1 \);
- If \( C = \frac{1}{R^2} > 0 \) and \( \|s\| \leq \pi R \), then \( 1 \geq \sigma_{\min}(\text{DExp}_x(s)) \geq \frac{\sin(|s|/R)}{|s|/R} \).

As usual, we use the convention \( \sin(x)/x = 1 \) at \( x = 0 \).

**Corollary 5.6.4.** Let \( M \) have sectional curvature upper bounded by \( C \), and let the retraction \( R \) be the exponential retraction \( \text{Exp} \). We have that:

- If \( C \leq 0 \), then [5.3.4] is satisfied for any \( a > 0 \) and \( b = 1 \);
- If \( C > 0 \), then [5.3.4] is satisfied for any \( 0 < a < \frac{\pi}{\sqrt{C}} \) and \( b = \frac{\sin(\sqrt{C}a)}{a\sqrt{C}} \).

The take away is similar to the comment following Corollary 5.6.2. Particular cases here include the Euclidean case with the exponential map \( \text{Exp}_x(s) = x + s \) (\( C = 0 \)), as well as Hadamard manifolds (\( C \leq 0 \)) (including positive definite matrices [102, 27]) and compact manifolds (see also below).

**Proof of Lemma 5.6.3** The proof relies essentially on Jacobi fields and the Jacobi field comparison theorem, as detailed below. Inequalities hold as equalities if the sectional curvature is constant and equal to \( C \) (use [91] Lemma 10.8] instead of the comparison theorem in that case.) Of course, for \( s = 0 \), \( \text{DExp}_x(0) \) is the identity operator, with all singular values equal to 1. Thus, we focus on \( s \neq 0 \).

Consider the smooth map \( \Gamma : I_1 \times I_2 \to M \) defined by

\[
\Gamma(q, t) = \text{Exp}_x \left( \frac{t}{|s|} [s + q\dot{s}] \right).
\]
(Intervals $I_1, I_2 \subseteq \mathbb{R}$ are neighborhoods of 0, as large as $\text{Exp}$ permits.) Notice that, for any $q$, $\Gamma(q, \cdot)$ defines a geodesic: we say that $\Gamma$ is a variation through geodesics of the geodesic $\gamma(t) = \Gamma(0, t)$. Furthermore, $\gamma$ has unit speed since its velocity at $\gamma(0) = x$ is $\dot{\gamma}(0) = \frac{s}{\|s\|}$.

The following (smooth) vector field along $\gamma$ describes how $\gamma$ varies at every point:

$$J(t) = \frac{\partial}{\partial q} \Gamma(q, t) \bigg|_{q=0} = \frac{t}{\|s\|} \text{DExp}_x \left( \frac{t}{\|s\|} s \right) [s].$$

Clearly, $J(t) \in T_{\gamma(t)}\mathcal{M}$ for all $t$. In particular,

$$J(\|s\|) = \text{DExp}_x(s)[s],$$

so that our goal is to understand $J(\|s\|)$.

The variation field $J$ of $\gamma$ is called a Jacobi field along $\gamma$. A Jacobi field is called normal along $\gamma$ if $J(t)$ is orthogonal to $\dot{\gamma}(t)$ for all $t$. By [91] Lemma 10.6, this is the case if and only if $J(0)$ and $\dot{J}(0)$ are orthogonal to $\dot{\gamma}(0)$. In our case, $\dot{\gamma}(0) = \frac{s}{\|s\|}$ and

$$J(0) = 0,$$

$$\dot{J}(0) = \frac{1}{\|s\|} \dot{s},$$

so that $J$ is normal if and only if $\dot{s}$ is orthogonal to $s$. We now have all the pieces to apply the following theorem (particularized to our situation).

**Theorem 5.6.5** (Jacobi field comparison theorem, [91] Thm. 11.2). *Assume all sectional curvatures of $\mathcal{M}$ are bounded above by $C$. If $J$ is normal (that is, if $\dot{s}$ is orthogonal to $s$), then*

$$\|J(t)\| \geq \begin{cases} 
\frac{t \|\dot{s}\|}{\|s\|} & \text{for } 0 \leq t, \text{ if } C = 0; \\
R \sin \left( \frac{t}{R} \right) \frac{\|\dot{s}\|}{\|s\|} & \text{for } 0 \leq t \leq \pi R, \text{ if } C = \frac{1}{R^2} > 0; \\
R \sinh \left( \frac{t}{R} \right) \frac{\|\dot{s}\|}{\|s\|} & \text{for } 0 \leq t, \text{ if } C = -\frac{1}{R^2} < 0.
\end{cases}$$

157
In our setting, with \( J(\|s\|) = \text{DExp}_x(s)[\hat{s}] \), this means that if \( \hat{s} \) is orthogonal to \( s \), then

\[
\|\text{DExp}_x(s)[\hat{s}]\| \geq \left\{
\begin{array}{ll}
\|\hat{s}\| & \text{if } C = 0; \\
\frac{\sin(\|s\|/R)}{\|s\|/R} \|\hat{s}\| & \text{if } \|s\| \leq \pi R \text{ and } C = \frac{1}{R} > 0; \\
\frac{\sinh(\|s\|/R)}{\|s\|/R} \|\hat{s}\| & \text{if } C = -\frac{1}{R^2} < 0,
\end{array}
\right.
\tag{5.6.8}
\]

and \( \text{DExp}_x(s)[\hat{s}] \) is orthogonal to \( \dot{\gamma}(\|s\|) \) owing to normality of \( J \).

On the other hand, \( J \) is called tangential if \( J(t) \) is a multiple of \( \dot{\gamma}(t) \) for all \( t \). This is the case if \( \dot{s} = \alpha s \) for some \( \alpha \in \mathbb{R} \). Indeed, in this case,

\[
\Gamma(q,t) = \text{Exp}_x \left( \frac{t}{\|s\|} \left[ 1 + \alpha q \right] s \right) = \gamma((1 + \alpha q)t).
\]

That is, \( \Gamma(q,\cdot) \) is a geodesic with speed \( 1 + \alpha q \). By the chain rule,

\[
J(t) = \left. \frac{\partial}{\partial q} \Gamma(q,t) \right|_{q=0} = \alpha t \dot{\gamma}(t).
\]

As announced, \( J \) is tangential. Furthermore, \( \|\dot{\gamma}(t)\| = 1 \) for all \( t \), so that

\[
\|\text{DExp}_x(s)[\hat{s}]\| = \|J(\|s\|)\| = \alpha \|s\| = \|\hat{s}\|.
\]

To summarize, we found \( \text{DExp}_x(s) \) is a linear operator such that \( \text{DExp}_x(s)[s] = \|s\| \dot{\gamma}(\|s\|) \) and such that if \( \hat{s} \perp s \), then \( \text{DExp}_x(s)[\hat{s}] \perp \text{DExp}_x(s)[s] \) (where \( u \perp v \) means \( u \) is orthogonal to \( v \)). The following elementary lemma applies.

**Lemma 5.6.6.** Consider a linear operator \( A: \mathcal{E} \to \mathcal{F} \) between two Euclidean spaces such that there exists \( v \in \mathcal{E} \), \( u \in \mathcal{F} \) of unit norm and \( \sigma \geq 0 \) with \( Av = \sigma u \), and such that \( Az \perp Av \) for all \( z \in \mathcal{E} \) with \( z \perp v \). Then, \( (u,v,\sigma) \) is a singular triplet of \( A \).

**Proof.** We work in coordinates: \( A \) is a matrix and \( u, v \) are column vectors. Let \( U_\perp \) and \( V_\perp \) be such that \([u \ U_\perp]\) and \([v \ V_\perp]\) are orthogonal matrices. Then,

\[
\begin{bmatrix} u & U_\perp \end{bmatrix}^T A \begin{bmatrix} v & V_\perp \end{bmatrix} = \]
for some matrix $K$, owing to our assumptions. With the SVD decomposition $K = U \Sigma V^T$, we find

$$A = \begin{bmatrix} u & U_{\perp} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & U \end{bmatrix} \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V \end{bmatrix}^T \begin{bmatrix} v & V_{\perp} \end{bmatrix}^T$$

$$= \begin{bmatrix} u & U_{\perp} \end{bmatrix} \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma \end{bmatrix} \begin{bmatrix} v & V_{\perp}V \end{bmatrix}^T.$$

Up to permutations, this is an SVD of $A$, which concludes the proof.

Thus, $\text{DExp}_x(s)$ admits 1 as a singular value, and all other singular values are bounded as per (5.6.8). To conclude, observe that $\sinh(x)/x \geq 1$ for all $x$.

5.6.3 Compact Manifolds with Bounded Sectional Curvature and Arbitrary Retraction

We extend the results above to general retractions $R$ for the case of compact manifolds. In doing so, we use the fact that compact manifolds have bounded sectional curvature [30, S9.3, p166]. We show the following theorem. Ultimately, the proof relies on the fact that retractions cannot depart too much from the exponential map for small steps, and that this departure can be bounded uniformly on compact manifolds.

**Theorem 5.6.7.** If $M$ is a compact manifold, then for every $\delta \in (0, 1)$, defining $b = 1 - \delta$, there exists $a > 0$ such that $A5.3.4$ is satisfied with this choice of $a$ and $b$.

**Proof of Theorem 5.6.7** We wish to lower bound the function

$$t(r) \triangleq \inf_{x \in M} \inf_{s \in T_x M, \|s\|_s \leq r} \sigma_{\min}(\text{DR}_x(s))$$
on $\mathbb{R}^+ = \{ r \in \mathbb{R} : r \geq 0 \}$. Since $\sigma_{\min}(A + B) \geq \sigma_{\min}(A) - \|B\|_{op}$ for any two linear operators $A$ and $B$ between the same vector spaces, we have that

$$t(r) \geq \inf_{x \in M} \inf_{s \in T_x M, |s|_x \leq r} \sigma_{\min}(D\text{Exp}_x(s)) - \sup_{x \in M} \sup_{s \in T_x M, |s|_x \leq r} \|DR_x(s) - D\text{Exp}_x(s)\|_{op}. $$

All sectional curvatures of $M$ are bounded above by some finite $C$ owing to compactness of $M$ [30] S9.3, p166]. Thus, Lemma 5.6.3 covers the first part of the bound. This motivates the following lemma, which addresses the second part of the bound.

**Lemma 5.6.8.** If the manifold $M$ is compact, then the function $h$ on $\mathbb{R}^+$ defined by

$$h(r) \triangleq \sup_{x \in M} \sup_{s \in T_x M, |s|_x \leq r} \|DR_x(s) - D\text{Exp}_x(s)\|_{op}$$

is upper-semicontinuous.

We prove this lemma later. Combining, we find

- If $C \leq 0$, $t(r) \geq 1 - h(r)$.
- If $C = \frac{1}{R^2} > 0$ and $0 \leq r \leq \pi R$, $t(r) \geq \frac{\sin(r/R)}{r/R} - h(r)$.

Since $DR_x(0)$ and $D\text{Exp}_x(0)$ coincide for all $x$ (they are the identity operator), we have that $h(0) = 0$. Therefore, using Lemma 5.6.8, we have that $t(0) = 1$ and $t(r)$ is lower bounded by a lower semi-continuous function achieving value 1 at $r = 0$. This implies that, for every $\delta > 0$ and defining $b = 1 - \delta$, there exists an $a > 0$ such that for all $r \leq a$, $t(r) \geq b$. \qed

We now turn to proving Lemma 5.6.8 crucial to the proof above. To this end, it is helpful to introduce a few results of topology first. We follow [25], including the blanket assumption that all topological spaces are assumed Hausdorff (page 65 in that reference).

Manifolds (including tangent bundles) are equipped with their natural topology.

**Definition 5.6.9** (Upper semi-continuous (u.s.c.) mapping). A correspondence $\Gamma: Y \to Z$ between two topological spaces $Y, Z$ is defined to be a u.s.c. mapping if, for all $y$ in $Y$, $\Gamma(y)$
is a compact subset of $Z$ and if, for any neighborhood $V$ of $\Gamma(y)$, there exists a neighborhood $U$ of $y$ such that for all $u \in U$, $\Gamma(u) \subseteq V$.

**Theorem 5.6.10** ([25, Thm. VI.2, p116]). *If $\phi$ is an upper semi-continuous, real-valued function in $Y \times Z$ and $\Gamma$ is a u.s.c. mapping of $Y$ into $Z$ (two topological spaces) such that, for each $Y$, $\Gamma(y)$ is non-empty, then the real-valued function $M$ defined by

$$M(y) = \max_{z \in \Gamma(y)} \phi(y, z)$$

is upper semi-continuous. ($M$ is well defined since, under the assumptions, the maximum is attained.)

The strategy to prove Lemma 5.6.8 is to apply Theorem 5.6.10 twice. To do so, we must establish that certain maps are u.s.c. To ease this task, we introduce the following lemma, which allows to transpose the question between homeomorphic topological spaces.

**Lemma 5.6.11.** Let $Y, Z$ be topological spaces homeomorphic to $Y'$, $Z'$ (respectively) through homeomorphisms $h_Y : Y \to Y'$, $h_Z : Z \to Z'$. Let $\Gamma : Y \to Z$ be a correspondence from $Y$ to $Z$. Further let $\Gamma' : Y' \to Z'$ be a correspondence from $Y'$ to $Z'$ related to $\Gamma$ though the following property:

$$\forall y \in Y, \quad h_Z(\Gamma(y)) = \Gamma'(h_Y(y)). \quad (5.6.9)$$

This property is best illustrated as a diagram:

$$
\begin{array}{c}
Y \downarrow h_Y \quad \Gamma \quad h_Z \quad \downarrow h_Y(Y') \quad \Gamma' \quad \downarrow h_Z(Z')
\end{array}
$$

Then, $\Gamma$ is u.s.c. if and only if $\Gamma'$ is u.s.c.
Proof of Lemma 5.6.11. By definition of homeomorphism, \( h_Y \) is invertible, and both \( h_Y \) and \( h_Y^{-1} \) are continuous. By continuity of \( h_Y^{-1} \), for any subset \( S \subseteq Y \) and any neighborhood \( N \) of \( S \) in \( Y \), we have that \( h_Y(N) \) is a neighborhood of \( h_Y(S) \) in \( Y' \) (and similarly for \( h_Z \)).

We now proceed with the proof: assume \( \Gamma' \) is u.s.c.; we show \( \Gamma \) is u.s.c. The other direction follows immediately by noting that, since \( h_Y, h_Z \) are homeomorphisms, so are \( h_Y^{-1}, h_Z^{-1} \).

First, we note that, for any \( y \in Y \), \( \Gamma(y) \) is compact. Indeed, by (5.6.9) we have that \( \Gamma(y) = h_Z^{-1}(\Gamma'(h_Y(y))) \).

Since \( \Gamma' \) is u.s.c, \( \Gamma'(h_Y(y)) \) is compact, and since \( h_Z^{-1} \) is continuous, \( \Gamma(y) \) is compact.

Therefore, all we need to show is that given any \( y \in Y \) and an arbitrary neighborhood \( V \) of \( \Gamma(y) \), there exists a neighborhood \( U \) of \( y \) such that

\[
\bigcup_{u \in U} \Gamma(u) \subseteq V.
\]

To this end, consider the set \( h_Z(V) \), which is a neighborhood of \( h_Z(\Gamma(y)) \) by continuity. Furthermore, owing to (5.6.9), we have that \( h_Z(\Gamma(y)) = \Gamma'(h_Y(y)) \). Therefore, \( h_Z(V) \) is a neighborhood of \( \Gamma'(h_Y(y)) \). Since \( \Gamma' \) is u.s.c., there exists a neighborhood \( U' \) of \( h_Y(y) \) such that

\[
\bigcup_{u' \in U'} \Gamma'(u') \subseteq h_Z(V). \tag{5.6.10}
\]

Now define \( U \subseteq Y \) as \( U = h_Y^{-1}(U') \). Note that \( U \) is a neighborhood of \( y \). Using (5.6.9), the definition of \( U \) and (5.6.10) (in that order) yields

\[
\bigcup_{u \in U} \Gamma(u) = \bigcup_{u \in U} h_Y^{-1}(\Gamma'(h_Y(u))) = \bigcup_{u' \in U'} h_Z^{-1}(\Gamma'(u')) \subseteq V,
\]

which finishes the proof. \( \square \)

We use the above lemma to prove the following statement.
Lemma 5.6.12. The correspondence $\Gamma$ between $\mathbb{R}^+ \times \mathcal{M}$ and $T\mathcal{M}$ defined by

$$\Gamma(r, x) = \{ s \in T_x\mathcal{M} : \|s\|_x \leq r \}$$  \hspace{1cm} (5.6.11)

is a u.s.c. mapping.

Proof of Lemma 5.6.12. The proof is in three parts: we first restrict $\Gamma$ using the domain of a chart of $\mathcal{M}$; then we argue that if all such restrictions are u.s.c. then $\Gamma$ itself is u.s.c.; and finally we show each restriction is u.s.c. using Lemma 5.6.11.

Part 1. Let $(V, \varphi)$ be a chart of $\mathcal{M}$. Let $Y = \mathbb{R}^+ \times V$ and $Z = TV = \{(x, s) : x \in V, s \in T_x\mathcal{M}\}$ be two topological spaces (with the natural topologies inherited from the smooth manifold structure of $\mathcal{M}$.) We consider the correspondence $\Gamma_V : Y \to Z$ obtained by restricting $\Gamma$ to $Y$.

Part 2. Assume each such $\Gamma_V$ is u.s.c.; then, $\Gamma$ is u.s.c. Indeed, for any $(r, x) \in \mathbb{R}^+ \times \mathcal{M}$, there exists a chart $(V, \varphi)$ such that $x \in V$. Since $\Gamma(r, x) = \Gamma_V(r, x)$ is compact by assumption on $\Gamma_V$, all images of $\Gamma$ are compact. Furthermore, let $N$ be any neighborhood of $\Gamma_V(r, x)$ in $T\mathcal{M}$. Restrict this neighborhood to $N_V = N \cap TV$: this is a neighborhood of $\Gamma_V(r, x)$ in $TV$. By assumption on $\Gamma_V$, there exists a neighborhood $U$ of $(r, x)$ in $\mathbb{R}^+ \times V$ such that $\bigcup_{u \in U} \Gamma_V(u) \subseteq N_V$. Thus, a fortiori, $\bigcup_{u \in U} \Gamma(u) \subseteq N$. In other words, $\Gamma$ is u.s.c.

Part 3. Consider a particular $\Gamma_V$ obtained from a chart $(V, \varphi)$ as in Part 1. To show it is u.s.c., we invoke Lemma 5.6.11. Following the notation of that lemma, define:

\begin{align*}
Y &= \mathbb{R}^+ \times V, & Y' &= \mathbb{R}^+ \times \varphi(V), & h_Y(r, x) &= (r, \varphi(x)) \\
Z &= TV, & Z' &= \varphi(V) \times \mathbb{R}^n, & h_Z(x, s) &= (\varphi(x), D\varphi(x)[s]).
\end{align*}
Here, $\varphi(V)$ is an open subset of $\mathbb{R}^n$ equipped with the usual topology. That the maps $h_Y, h_Z$ are homeomorphisms follows tautologically from the definitions of the natural topologies on $\mathcal{M}$ and $T\mathcal{M}$. By Lemma 5.6.11, we need only show that $\Gamma'_V = h_Z \circ \Gamma_V \circ h_Y^{-1}$ is u.s.c. Given $r \geq 0$ and $\hat{x} = \varphi(x)$ arbitrary,

$$\Gamma'_V(r, \hat{x}) = h_Z(\Gamma_V(r, x))$$

$$= h_Z(\{(x, s) \in TV : \|s\|_x \leq r\})$$

$$= \{(\hat{x}, \hat{s}) : \hat{s} = D\varphi(x)[s], \|s\|_x \leq r\}$$

$$= \{(\hat{x}, \hat{s}) : \hat{s}^\top G_{\hat{x}} \hat{s} \leq r^2\},$$

where $G_{\hat{x}}$ is a positive definite matrix which represents the Riemannian metric at $x$ in the coordinates inherited from $\varphi$. Owing to positive definiteness of $G_{\hat{x}}$, images of $\Gamma'_V$ are compact.

It remains to show that, for any neighborhood $N$ of $\Gamma'_V(r, \hat{x})$, there exists a neighborhood $U$ of $(r, \hat{x})$ such that, for all $u \in U$, the inclusion $\Gamma'_V(u) \subseteq N$ holds. To this end, note that, owing to compactness of $\Gamma'_V(r, \hat{x})$, there exist $\varepsilon_1, \varepsilon_2 > 0$ such that:

For all $\hat{x}' \in \varphi(V), \hat{s}' \in \mathbb{R}^n$, if $\|\hat{x} - \hat{x}'\|_2 < \varepsilon_1$ and $(\hat{s}')^\top G_{\hat{x}} \hat{s}' < (r + \varepsilon_2)^2$, then $(\hat{x}', \hat{s}') \in N$.

Consider the set

$$\Gamma'_V(\hat{x} + \delta \hat{x}, r + \delta r) = \{(\hat{x} + \delta \hat{x}, \hat{s}') : (\hat{s}')^\top G_{\hat{x} + \delta \hat{x}} \hat{s}' < (r + \delta r)^2\}.$$ 

To ensure this set is included in $N$, we first require $\|\delta \hat{x}\|_2 < \varepsilon_1$. Second, we want to require

$$(\hat{s}')^\top G_{\hat{x}} \hat{s}' < (r + \varepsilon_2)^2.$$
We know the following:

\[(s')^\top G_{\dot{x}} s' = (s')^\top G_{\dot{x} + \delta \dot{x}} s' + (s')^\top (G_{\dot{x}} - G_{\dot{x} + \delta \dot{x}}) s' \leq (r + \delta r)^2 \left[ 1 + \left\| G_{\dot{x} + \delta \dot{x}} G_{\dot{x}}^{-1/2} - I \right\|_{op} \right].\]

The term in brackets is a continuous function of \( \delta \dot{x} \) which evaluates to 1 at \( \delta \dot{x} = 0 \). Thus, possibly after reducing \( \varepsilon_1 \), we can ensure the term between brackets is strictly less than \( 1 + \varepsilon_3 \) for any \( \varepsilon_3 > 0 \). Hence, we require the following condition on \( \delta r \):

\[(1 + \varepsilon_3)(r + \delta r)^2 < (r + \varepsilon_2)^2.\]

Pick \( \varepsilon_3 \) small enough so that \( \delta r = 0 \) satisfies this condition. Combining, we showed there exists an open set around \( \dot{x} \) in \( \varphi(V) \) and an open set around \( r \) in \( \mathbb{R}^+ \) such that \( \Gamma' \) maps to a subset of \( \mathcal{N} \). The product of these open sets defines an adequate neighborhood \( U \) of \( (r, \dot{x}) \) in \( \mathbb{R}^+ \times \varphi(V) \), which concludes the proof.

\[\square\]

We are now ready to prove the remaining lemma.

**Proof of Lemma 5.6.8**  Consider the following continuous function:

\[f : \mathbb{R}^+ \times \mathcal{T}M \to \mathbb{R} : (r, x, s) \mapsto f(r, x, s) = \| DR_x(s) - DExp_x(s) \|_{op}.\]

Note that \( f \) is constant with respect to \( r \); formally allowing \( f \) to be a function of \( r \) nonetheless is convenient for technical reasons below. Further consider the function

\[g : \mathbb{R}^+ \times \mathcal{M} \to \mathbb{R} : (r, x) \mapsto g(r, x) = \max_{s \in \Gamma(r, x)} f(r, x, s),\]

where \( \Gamma \) is as defined by (5.6.11). Note that \( h(r) = \sup_{x \in \mathcal{M}} g(r, x) \).

165
Continuity of \( f \) and the fact that \( \Gamma \) is u.s.c. (established by Lemma 5.6.12) along with Theorem 5.6.10 (where \( y = (r, x) \) and \( z \in \Gamma(r, x) \)) imply that the function \( g(r, x) \) is upper semi-continuous on the space \( \mathbb{R}^+ \times \mathcal{M} \). We invoke Theorem 5.6.10 once more, this time with \( Y \) as \( \mathbb{R}^+ \), \( Z \) as \( \mathcal{M} \), \( \phi \) as \( g \) and with the correspondence \( \Gamma(r) = \mathcal{M} \) for all \( r \geq 0 \)—this requires compactness of \( \mathcal{M} \). This implies that the function \( h(r) \) is upper semi-continuous.

\[ \square \]

5.7 Numerical experiments

We implement Algorithm 6 within the Manopt framework [35] and compare the performance of our implementation against some existing solvers in that toolbox, namely, the Riemannian trust-region method (RTR) [2] and the conjugate gradients method (CG). We consider a suite of nine Riemannian optimization problems from the collection of problems provided with Manopt, and we run our method with the choices of \( \theta = 1, 0.1, 0.01 \) (see A5.3.3).

Figure 5.1 displays performance profiles for the various algorithms on these same problems. Figure (a) considers running time, whereas figure (b) shows (outer) iteration count. If a curve passes through the point \((\alpha, k)\), it means it terminated for \( k \) out of nine problems in at most \( \alpha \) times the best result of any algorithm for each of these \( k \) problems. As can be seen from the profiles, our method is competitive with RTR (which has been fine-tuned over many years) in terms of the number of iterations, and it shows promise for its overall runtime performance.

5.8 Conclusion

In this chapter, we considered the minimization of an appropriately smooth function on a Riemannian manifold. We extended algorithms based adaptive regularization of cubics to this setting. We showed that under appropriate smoothness assumptions on the function as well as the underlying manifold and the retraction being used, one can recover essentially
Figure 5.1: Performance profiles of ARC for $\theta = 1, 0.1, 0.01$, as compared to the Riemannian trust-region method (RTR) and the conjugate gradients method (CG) in terms of (a) time and (b) iterations.

the same rate of convergence as the Euclidean case. An interesting direction for future work would be to derive a second-order analysis, namely, an upper bound on the iteration complexity of computing an approximate local minima, i.e. an approximate critical point which also approximately satisfies second-order necessary optimality conditions.
Chapter 6

Lower Bounds for Higher-Order Convex Optimization

6.1 Introduction

Linearly-converging (methods for which the dependence on the additive error $\varepsilon$ is logarithmic) convex optimization algorithms fall into two categories. The first are methods whose iteration complexity scales with the dimension. This includes the ellipsoid [88, 77], cutting plane [139, 92] and random-walk [26, 86, 98] based methods. They solve convex optimization in the general membership oracle model.

The other category of linearly-converging algorithms is iterative derivative-based methods. These achieve fast dimension-free iteration rates for certain types of convex functions, namely those that are strongly convex and smooth. Indeed (as discussed in the first three chapters) gradient descent and its various extensions have therefore been extremely successful for optimization especially in machine learning.

However, as established in Chapters 3 and 4, state-of-the-art optimization methods for machine learning is achieved by methods that are based on higher derivatives. Notably, the fastest theoretical running times for both convex (Theorem 3.1.5 and other references
in Section 2.1.3) and non-convex (Theorem 4.3.1 and other references in Section 4.1.1) optimization are attained by algorithms that either explicitly or implicitly exploit second-order information and third-order smoothness.

Of particular interest is Newton’s method, due to recent efficient implementations that run in near-linear time in the input representation. A question of interest prior to the work was that whether Newton’s method or higher-order methods can under appropriate smoothness conditions, and without requiring curvature assumptions, achieve logarithmic in error iteration complexity which is independent of the dimensionality, which is extremely high for many large-scale applications.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dim. Free</th>
<th>Order</th>
<th>Assumptions</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ellipsoid</td>
<td>No</td>
<td>NA</td>
<td>None</td>
<td>(\text{poly}(d, \log(1/\varepsilon)))</td>
</tr>
<tr>
<td>Random Walks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interior Point</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradient Descent</td>
<td>Yes</td>
<td>k=1</td>
<td>Bounded (L_2) (\lambda_{\text{min}} &gt; 0)</td>
<td>(O\left(\frac{L_2}{\lambda_{\text{min}}}\right)^{1/2} \log(\frac{1}{\varepsilon}))</td>
</tr>
<tr>
<td>Newton’s method</td>
<td>Yes</td>
<td>k=2</td>
<td>Bounded (L_3) (\lambda_{\text{min}} &gt; 0)</td>
<td>(\tilde{O}\left(\frac{L_3}{\lambda_{\text{min}}}^{2/7}\right) + \log \log(\frac{1}{\varepsilon}))</td>
</tr>
<tr>
<td>Higher-Order</td>
<td>Yes</td>
<td>(k &gt; 2)</td>
<td>Bounded (L_k)</td>
<td>(O\left(\frac{L_{k+1}}{\varepsilon}^{1/(k+1)}\right))</td>
</tr>
</tbody>
</table>

Table 6.1: Table summarizing known results for convex optimization.

Table 6.1 surveys the known algorithms for convex optimization and their iteration complexity. Polynomial time algorithms admit linear convergence, i.e. \(O(\log \frac{1}{\varepsilon})\), and invariably depend on the dimension. Dimension free polynomial-time algorithms require both an upper bound on the smoothness and a lower bound on the strong convexity.

The main question we provide an answer to in this chapter is whether there exists linearly-converging iterative algorithms for (high-order) smooth functions that are not strongly convex?
In this chapter we show that unfortunately, these hopes cannot be attained without stronger assumptions on the underlying optimization problem. In particular Theorem 6.1.2 shows that even if the functions are \( k \)th-order smooth, for arbitrarily large \( k \), and the iterative algorithm uses \( k \)th-order derivative information, the answer is negative. To the best of our knowledge, our results are the first lower bound for \( k \)th-order optimization for \( k \geq 2 \) that include higher-order smoothness.

6.1.1 Statement of Result

We consider the problem of \( k \)th-order optimization. We model a \( k \)th-order algorithm as follows. Given a \( k \)-times differentiable function \( f : \mathbb{R}^d \to \mathbb{R} \), at every iteration \( i \), the algorithm outputs a point \( x_i \) and receives as input the tuple \([f(x_i), \nabla f(x_i), \nabla^2 f(x_i), \ldots \nabla^k f(x_i)]\), i.e. the value of the function and its \( k \) derivatives at \( x_i \). The goal of the algorithm is to output a point \( x_T \) such that

\[
    f(x_T) - \min_{x \in \mathbb{R}^d} f(x) \leq \varepsilon
\]

For the \( k \)th-order derivatives to be informative, one needs to bound their rate of change, or equivalently the Lipschitz constant of its derivative. This is called \( k \)th-order smoothness, and we denote it by \( L_{k+1} \). In particular consider the following definition

**Definition 6.1.1.** A \( k \)-times differentiable function \( f \) is called \( k \)th-order smooth if there exists a constant \( L_{k+1} \) such that for all \( x, y \)

\[
    \| \nabla^k f(x) - \nabla^k f(y) \| \leq L_{k+1} \| x - y \|
\]

In the above \( \| \cdot \| \) is defined as the induced operator norm with respect to the Euclidean norm. Our main theorem shows the limitation of \( k \)th-order iterative optimization algorithms:

\( ^1 \)After the writing of the first manuscript [9], we were made aware of the work by [22] which provides lower bounds for these settings as well.

\( ^2 \)An iteration is equivalent to an oracle call to \( k \)th-order derivatives in this model.
Theorem 6.1.2. For every number $L_{k+1}$ and $k^{th}$-order algorithm $ALG$ (deterministic or randomized), there exists a number $\varepsilon_0(L_{k+1})$ such that for all $\varepsilon \leq \varepsilon_0(L_{k+1})$, there exists a $k^{th}$-order smooth convex function $f : B_d \rightarrow \mathbb{R}$ with smoothness coefficient $L_{k+1}$ such that $ALG$ cannot output a point $x_T$ such that

$$f(x_T) \leq \min_{x \in B_d} f(x) + \varepsilon$$

in number of iterations $T$ fewer than

$$c_k \left( \frac{L_{k+1}}{\varepsilon} \right)^{\Omega(1/k)}$$

where $c_k$ is a constant depending on $k$ and $B_d$ is defined to be the unit ball in $\mathbb{R}^d$.

The above lower bound is known to be tight up to constants in the exponent: for $k > 2$, [23] proves an upper bound of $O \left( \left( \frac{L_{k+1}}{\varepsilon} \right)^{\frac{1}{k+1}} \right)$.

Although the bound is stated for constrained optimization over the unit ball, it can be extended to an unconstrained setting via the addition of an appropriate scaled multiple of $\|x\|^2$. We leave this adaptation for future work. Further as is common with lower bounds the underlying dimension $d$ is assumed to be large enough and differs for the deterministic vs randomized version. Theorems 6.5.1 and 6.6.1 make the dependence precise.

Table 6.2 surveys the known lower bounds for derivative based optimization. For the case of $k = 2$, the most efficient methods known are the cubic regularization technique proposed by [113] and an accelerated hybrid proximal extra-gradient method proposed by [104]. The best known upper bound in this setting is $O \left( \frac{L_3}{\varepsilon} \right)^{2/7}$ [104]. We show a lower bound of $\Omega \left( \left( \frac{L_3}{\varepsilon} \right)^{2/11} \right)$ (c.f. Theorem 6.2.1) demonstrating that the upper bound is nearly tight.
### 6.1.2 Related work.

The literature on convex optimization is too vast to survey; the reader is referred to [36, 112].

Lower bounds for convex optimization were studied extensively in the seminal work of [109]. In particular, tight first-order optimization lower bounds were established assuming first-order smoothness. (Also see [112] for a concise presentation of the lower bound). In a recent work, [21] presented a lower bound when given access to second-order derivatives. However a key component (as remarked by the authors themselves) missing from the bound established by [21] was that the constructed function was not third-order smooth. Indeed the lower bound established by [21] can be overcome when the function is third-order smooth (ref. [113]). The upper bounds for higher-order oracles (assuming appropriate smoothness) was established by [23].

Higher order smoothness has been leveraged recently in the context of non-convex optimization [6, 41, 14]. In a surprising new discovery, [43, 17] show that assuming higher-order smoothness the bounds for first-order optimization can be improved without having explicit access to higher-order oracles. This is a property observed in our lower bound too. Indeed as shown in the proof the higher order derivatives at the points queried by the algorithm are always 0. For further details regarding first-order lower bounds for various different settings we refer the reader to [5, 143, 21, 20] and the references therein. The work of [78] studies a different kind of smoothing namely inf-convolution to obtain first-order smoothness in arbitrary norms, however it does not provide guarantees with higher-order smoothness.
In parallel and independently, Arjevani et al. [22] also obtain lower bounds for deterministic higher-order optimization. In comparison, their lower bound is stronger in terms of the exponent than the ones proved in this chapter, and matches the upper bound for $k = 2$. However, our construction and proof are simple (based on the well known technique of ball smoothing) and our bounds hold for randomized algorithms as well, as opposed to their deterministic lower bounds.

6.1.3 Overview of Techniques

Our lower bound is inspired by the lower bound presented in [49]. In particular we construct the function as a piece-wise linear convex function defined by $f(x) = \max_i \{ a_i^T x \}$ with carefully constructed vectors $a_i$ and restricting the domain to be the unit ball. The key idea here is that querying a point reveals information about at most one hyperplane. The optimal point however can be shown to require information about all the hyperplanes.

Unfortunately the above function is not differentiable. We now smooth the function by the ball smoothing operator (defined in Definition 6.3.2) which averages the function in a small Euclidean ball around a point. We show (c.f. Corollary 6.3.4) that iterative application of the smoothing operator ensures $k$-differentiability as well as boundedness of the $k^{th}$-order derivative.

Two key issues arise due to smoothing described above. Firstly although the smoothing operator leaves the function unchanged around regions far away from the intersection of the hyperplanes, it is not the case for points lying near the intersection. Indeed querying a point near the intersection of the hyperplanes can potentially lead to leak of information about multiple hyperplanes at once. To avoid this, we carefully shift the linear hyperplanes making them affine and then arguing that this shifting indeed forces sufficient gap between the points queried by the algorithm and the intersections leaving sufficient room for smoothing.

Secondly such a smoothing is well known to introduce a dependence on the dimension $d$ in the smoothness coefficients. Our key insight here is that for the class of functions being
considered for the lower bound (c.f. Definition 6.3.1) smoothing can be achieved without a
dependence on the dimension (c.f. Theorem 6.3.3). This is essential to achieving dimension
free lower bounds and we believe this characterization can be of intrinsic interest.

6.2 Main Result

The main result we prove in the chapter is given by the following theorem. The theorem is a
restatement of Theorem 6.1.2.

**Theorem 6.2.1.** For every number $L_{k+1}$ and $k^{th}$-order algorithm ALG (deterministic or
randomized), there exists an $\varepsilon_0(L_{k+1})$ such that for all $\varepsilon \leq \varepsilon_0(L_{k+1})$, there exists a $k$-
differentiable convex function $f \in B_d \to \mathbb{R}$ with $k^{th}$-order smoothness coefficient $L_{k+1}$ such
that ALG cannot output a point $x_T$ such that

$$f(x_T) \leq \min_{x \in B_d} f(x) + \varepsilon$$

in number of iterations $T$ fewer than

$$c_k \left( \frac{L_{k+1}}{\varepsilon} \right)^{\frac{2}{5k+1}}$$

where $c_k$ is a constant depending on $k$.

**Roadmap** : The rest of the chapter is dedicated to the proof of the above theorem.

We begin by providing requisite notation and definitions for the smoothing operator and
providing the relevant lemmas regarding smoothing in Section 6.3. In Section 6.4 we provide
the construction of our hard function. In Section 6.5 we state and prove our main theorem
(Theorem 6.5.1) showing the lower bound against deterministic algorithms. We also prove
Theorem 6.1.2 based on Theorem 6.5.1 in this Section. In Section 6.6 we state and prove
Theorem 6.6.1 showing the lower bound against randomized algorithms.
6.3 Preliminaries

6.3.1 Notation

We use $B_d$ to refer to the $d$-dimensional $\ell_2$ unit ball. We suppress the $d$ from the notation when it is clear from the context. Let $\Gamma$ be an $r$-dimensional linear subspace of $\mathbb{R}^d$. We denote by $M_\Gamma$, an $r \times d$ matrix which contains an orthonormal basis of $\Gamma$ as rows. Let $\Gamma^\perp$ denote the orthogonal complement of $\Gamma$. Given a vector $v$ and a subspace $\Gamma$, let $v \perp \Gamma$ denote the perpendicular component of $v$ w.r.t $\Gamma$. We now define the notion of a $\Gamma$-invariant function.

**Definition 6.3.1 ($\Gamma$-invariance).** Let $\Gamma$ be an $r$ dimensional linear subspace of $\mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be $\Gamma$-invariant if for all $x \in \mathbb{R}^d$ and $y$ belonging to the subspace $\Gamma^\perp$, i.e. $M_\Gamma y = 0$, we have that

$$f(x) = f(x + y)$$

Equivalently there exists a function $g : \mathbb{R}^r \to \mathbb{R}$ such that for all $x$, $f(x) = g(M_\Gamma x)$.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is defined to be $c$-Lipschitz with respect to a norm $\| \cdot \|$ if it satisfies

$$f(x) - f(y) \leq c\|x - y\|$$

Lipschitzness for the rest of the chapter will be measured in the $\ell_2$ norm.

6.3.2 Smoothing

In this section we define the smoothing operator and derive the requisite properties.

**Definition 6.3.2 (Smoothing operator).** Given an $r$-dimensional subspace $\Gamma \in \mathbb{R}^d$ and a parameter $\delta > 0$, define the operator $S_{\delta, \Gamma} : (\mathbb{R}^d \to \mathbb{R}) \to (\mathbb{R}^d \to \mathbb{R})$ (referred henceforth
as the smoothing operator) as

\[ S_{\delta, \Gamma} f(x) \triangleq \mathbb{E}_{\nu \in \Gamma, \|\nu\| \leq 1}[f(x + \delta \nu)] \]

where the expectation is over sampling a unit vector from the subspace \( \Gamma \) uniformly randomly.

As a shorthand we define \( f_{\delta, \Gamma} \triangleq S_{\delta, \Gamma} f \). Further for any \( t \in \mathbb{N} \) define \( S_{\delta, \Gamma}^{t} f \triangleq S_{\delta, \Gamma}(\ldots S_{\delta, \Gamma}(f)) \) i.e. the smoothing operator applied on \( f \) iteratively \( t \) times.

When \( \Gamma = \mathbb{R}^{d} \) we suppress the notation from \( f_{\delta, \Gamma} \) to \( f_{\delta} \). Following is the main lemma we prove regarding the smoothing operator.

**Lemma 6.3.3.** Let \( \Gamma \) be an \( r \)-dimensional linear subspace of \( \mathbb{R}^{d} \) and \( f : \mathbb{R}^{d} \to \mathbb{R} \) be \( \Gamma \)-invariant and \( G \)-Lipschitz. Let \( f_{\delta, \Gamma} \triangleq S_{\delta, \Gamma} f \) be the smoothing of \( f \). Then we have the following properties.

1. \( f_{\delta, \Gamma} \) is differentiable and also \( G \)-Lipschitz and \( \Gamma \)-invariant.
2. \( \nabla f_{\delta, \Gamma} \) is \( \frac{rG}{\delta} \)-Lipschitz.
3. \( \forall x : |f_{\delta, \Gamma}(x) - f(x)| \leq \delta G \)

Following is a corollary of the above lemma.

**Corollary 6.3.4.** Given a \( G \)-Lipschitz continuous function \( f \) and an \( r \)-dimensional subspace \( \Gamma \) such that \( f \) is \( \Gamma \)-invariant, we have that the function \( S_{\delta, \Gamma}^{k} f \) is \( k \)-times differentiable \( \forall k \). Moreover we have that for any \( x, y \)

\[ \forall i \in [k] \| \nabla^{i} S_{\delta, \Gamma}^{k} f(x) - \nabla^{i} S_{\delta, \Gamma}^{k} f(y) \| \leq \left( \frac{r}{\delta} \right)^{i} G \| x - y \| \]

\[ |S_{\delta, \Gamma}^{k} f(x) - f(x)| \leq G\delta k \]

We prove the above now.
Proof of Lemma 6.3.3. As stated before $f$ being $\Gamma$-invariant implies that there exists a function $g$ such that $f(x) = g(\Gamma x)$. Therefore we have that

$$f_{\delta,\Gamma}(x) = \mathbb{E}_{v \sim |v| \leq 1} [f(x + \delta v)] = \mathbb{E}_{v \sim \Gamma |v| \leq 1} [g(M_{\Gamma} x + \delta M_{\Gamma} v)] = g_{\delta}(M_{\Gamma} x)$$

where $g_{\delta}(x) = S_{\delta,\mathbb{R}} g(x)$. The representation of $f_{\delta,\Gamma}$ as $g_{\delta}(M_{\Gamma} x)$ implies that $f_{\delta,\Gamma}$ is $\Gamma$-invariant. Further the above equality implies that $\nabla f_{\delta,\Gamma}(x) = M_{\Gamma}^T \nabla g_{\delta}(M_{\Gamma} x)$. A standard argument using Stokes’ theorem shows that $g_{\delta}$ is differentiable even when $g$ is not differentiable in a measure 0 set which is always the case with our constructions.

$$\nabla g_{\delta}(x) = \frac{r}{\delta} \mathbb{E}_{v \sim S_r} [g(y + \delta v) v]$$

for $i \in [s]$ and for a unit vector $v$. We will first consider the case $i < k$. Using the inductive hypothesis and the fact that smoothing and derivative commute for $g$ to be not differentiable in a measure 0 set which is always the case with our constructions.

Proof of Corollary 6.3.4. We will argue inductively. The base case ($k = 0$) is a direct consequence of the function $f$ being $G$-Lipschitz. Suppose the theorem holds for $k - 1$. To argue about $\|\nabla^i S_{\delta,\Gamma}^k f(x) - \nabla^i S_{\delta,\Gamma}^k f(y)\|$ we will consider the function $q_{i,v}(x) = \nabla^i S_{\delta,\Gamma}^{k-1} f(x) [v^{\otimes i}]$ for $i \in [k]$ and for a unit vector $v$. We will first consider the case $i < k$.

The first inequality follows from Jensen’s inequality and the second inequality follows from noticing that $f$ being $G$-Lipschitz implies that $g$ is $G$-Lipschitz. We now have that

$$\|\nabla f_{\delta,\Gamma}(x) - \nabla f_{\delta,\Gamma}(y)\| \leq \|M_{\Gamma}(\nabla g_{\delta}(M_{\Gamma} x) - \nabla g_{\delta}(M_{\Gamma} y))\| \leq \frac{rG}{\delta} \|M_{\Gamma}\| \|x - y\|$$

$f$ being $G$-Lipschitz immediately gives us $\forall x : |f_{\delta,\Gamma}(x) - f(x)| \leq \delta G$.

---

3We need $g$ to be not differentiable in a measure 0 set which is always the case with our constructions.
differentiable functions we have that

$$S_{\delta, \Gamma} q_{i,v}(x) = \nabla^i S_{\delta, \Gamma}^k f(x)[v^\otimes i]$$

Note that the inductive hypothesis implies that $q_{i,v}(x)$ is $(\frac{r}{\delta})^i G$-Lipschitz and so is $S_{\delta, \Gamma} q_{i,v}(x)$ via Lemma 6.3.3. Therefore we have that

$$\forall i \in [k-1] \| \nabla^i S_{\delta, \Gamma}^k f(x) - \nabla^i S_{\delta, \Gamma}^k f(y) \| \leq \left( \frac{r}{\delta} \right)^i G \| x - y \|$$

We now consider the case when $i = k$. By Lemma 6.3.3 we know that $S_{\delta, \Gamma} q_{k-1,v}(x) = \nabla^{k-1} S_{\delta, \Gamma}^k f(x)[v^\otimes i]$ is differentiable and therefore we have that $S_{\delta, \Gamma}^k f(x)$ is $k$ times differentiable. Further we have that

$$\nabla S_{\delta, \Gamma} q_{k-1,v}(x) = \nabla^k S_{\delta, \Gamma}^k f(x)[v^\otimes k]$$

A direct application of Lemma 6.3.3 and the inductive hypothesis which implies that $q_{k-1,v}$ is $(\frac{r}{\delta})^{k-1} G$-Lipschitz gives that

$$\| \nabla^k S_{\delta, \Gamma}^k f(x) - \nabla^k S_{\delta, \Gamma}^k f(y) \| \leq \left( \frac{r}{\delta} \right)^k G \| x - y \|$$

Further it is immediate to see that

$$\inf_{y : \| y - x \| \leq k \delta} f(y) \leq S_{\delta, \Gamma}^k f(x) \leq \sup_{y : \| y - x \| \leq k \delta} f(y)$$

which implies using the fact that $f$ is $G$ Lipschitz that

$$| S_{\delta, \Gamma}^k f(x) - f(x) | \leq G \delta k$$
6.4 Construction of the Hard Function

In this section we describe the construction of our hard function $f^\dagger$. Our construction is inspired by the information-theoretic hard instance based on zero-sum games proposed by [49]. The construction of the function will be characterized by a sequence of vectors $X^{1\rightarrow r} = \{x_1 \ldots x_r\}$, $x_i \in B_d$ and parameters $k, \gamma, \delta, m$. We assume $d > m \geq r$. To make the dependence explicit we denote the hard function as

$$f^\dagger(X^{1\rightarrow r}, \gamma, \delta, m) : B_d \rightarrow \mathbb{R}$$

For brevity in the rest of the section we suppress $X^{1\rightarrow r}, \gamma, k, \delta, m$ from the notation, however all the quantities defined in the section depend on them. To define $f^\dagger$ we will define auxiliary vectors $\{a_1 \ldots a_r\}$ and auxiliary functions $f, \tilde{f}$.

Given a sequence of vectors $\{x_1, x_2, \ldots x_r\}, x_i \in B_d$, let $X_i$ for $i \leq r$ be defined as the subspace spanned by the vectors $\{x_1 \ldots x_i\}$. Further inductively define vectors $\{a_1 \ldots a_r\}$ as follows.

If $x_i \notin X_{i-1}$, define

$$a_i = \frac{\hat{a}_i}{\|\hat{a}_i\|} \text{ where } \hat{a}_i = x_i \perp X_{i-1}$$

If indeed $x_i \in X_i$, then $a_i$ is defined to be an arbitrary unit vector in the orthogonal component $X_i \perp X_{i-1}$. Further define an auxiliary function

$$f(x) = \max_{i \in [r]} f_i(x) \text{ where } f_i(x) = a_i^T x$$

Given the parameter $\gamma$, now define the following functions

$$\tilde{f}(x) = \max_{i \in [r]} \tilde{f}_i(x) \text{ where } \tilde{f}_i(x) = f_i(x) + \left(1 - \frac{i}{m}\right) \gamma \approx a_i^T x + \left(1 - \frac{i}{m}\right) \gamma$$
With these definitions in place we can now define the hard function parametrized by $k, \delta$.

Let $A_r$ be the subspace spanned by $\{a_1 \ldots a_r\}$

$$f^\dagger(X^{1-r}, k, \gamma, \delta, m) = S_{\delta, A_r}^k \tilde{f}(X^{1-r}, \gamma, m)$$

(6.4.1)

i.e. $f^\dagger$ is constructed by smoothing $\tilde{f}$ $k$-times with respect to the parameters $\delta, A_r$. We now collect some important observations regarding the function $f^\dagger$.

**Observation 6.4.1.** $f^\dagger$ is convex and continuous. Moreover it is 1-Lipschitz and is invariant with the respect to the $r$-dimensional subspace $A_r$.

Note that $\tilde{f}$ is a max of linear functions and hence convex. Since smoothing preserves convexity we have that $f^\dagger$ is convex. 1-Lipschitzness follows by noting that by definition $\|a_i\| = 1$ and it can be seen that $\tilde{f}$ is $A_r$-invariant and therefore by Theorem 6.3.3 we get that $f^\dagger$ is $A_r$-invariant.

**Observation 6.4.2.** $f^\dagger$ is $k$-differentiable with the Lipschitz constants $L_{i+1} \leq \left(\frac{l}{\delta}\right)^i$ for all $i \leq k$.

Above is a direct consequence of Corollary 6.3.4 and the fact that $\tilde{f}$ is 1-Lipschitz and invariant with respect to the $r$-dimensional subspace $A_r$. Corollary 6.3.4 also implies that

$$\forall x \quad \|f^\dagger(x) - \tilde{f}(x)\| \leq k\delta$$

(6.4.2)

Setting $\hat{x} = -\sum_{i=1}^r a_i$, we get that $f(\hat{x}) = \frac{-1}{\sqrt{r}}$. Therefore the following inequality follows from Equation (6.4.2) and by noting that $\|f(x) - \tilde{f}(x)\|_\infty \leq \gamma$

$$\min_{x \in B_d} f^\dagger(x) \leq f^\dagger(\hat{x}) \leq \frac{-1}{\sqrt{r}} + \gamma + k\delta$$

(6.4.3)

The following lemma provides a characterization of the derivatives of $f^\dagger$ at the points $x_i$. 180
Lemma 6.4.3. Given a sequence of vectors \( \{x_1 \ldots x_r\} \) and parameters \( \delta, \gamma, r, m, \) let \( \{g_1 \ldots g_r\} \) be a sequence of functions defined as

\[
\forall i \quad g_i = f^i(X^{1-i}, k, \gamma, \delta, m)
\]

If the parameters are such that \( 2k\delta \leq \frac{\gamma}{m} \) then we have that

\[
\forall i \in [r] \quad \forall j \in [k] \quad g_i(x_i) = g_r(x_i), \quad \nabla^j g_i(x_i) = \nabla^j g_r(x_i)
\]

Proof. We will first note the following about the smoothing operator \( S^k_\delta \). At any point \( x \), all the \( k \) derivatives and the function value of \( S^k_\delta f \) for any function \( f \) depend only on the value of the function \( f \) in a ball of radius at most \( k\delta \) around the point \( x \). Consider the function \( g_r \) and \( g_i \) for any \( i \in [r] \). Note that by definition of the functions \( g_i \), for any \( x \) such that

\[
\arg\max_{j \in [r]} a_j^T x + \left( 1 - \frac{j}{m} \right) \gamma \leq i
\]

we have that \( g_i(x) = g_r(x) \). Therefore to prove the lemma it is sufficient to show that

\[
\forall i, x \in \|x - x_i\| \leq k\delta \quad \arg\max_{j \in [r]} a_j^T x + \left( 1 - \frac{j}{m} \right) \gamma \leq i
\]

Let us first note the following facts. By construction we have that \( \forall j > i, a_j^T x_i = 0 \). This immediately implies that

\[
\max_{j > i} a_j^T x_i + \left( 1 - \frac{j}{m} \right) \gamma = \left( 1 - \frac{i + 1}{m} \right) \gamma
\]

Further using the fact that \( \|a_j\| \leq 1, \forall j \in [r] \) we have that

\[
\forall x \text{ s.t. } \|x - x_i\| \leq k\delta \text{ we have } \max_{j > i} a_j^T x + \left( 1 - \frac{j}{m} \right) \gamma \leq \left( 1 - \frac{i + 1}{m} \right) \gamma + k\delta
\]
Further note that by construction $a_i^T x_i \geq 0$ which implies $a_i^T x + (1 - \frac{i}{m}) \gamma \geq (1 - \frac{i}{m}) \gamma$. Again using the fact that $\|a_j\| \leq 1, \forall j \in [r]$ we have that

$$\forall x \text{ s.t. } \|x - x_i\| \leq k\delta \text{ we have } \max_{j \leq i} a_j^T x + \left(1 - \frac{j}{m}\right) \geq \left(1 - \frac{i}{m}\right) \gamma - k\delta \quad (6.4.6)$$

The above equations in particular imply that as long as $2k\delta < \frac{\gamma}{m}$, we have that

$$\forall x \text{ s.t. } \|x - x_i\| \leq k\delta \argmax_{j \in [r]} a_j^T x + \left(1 - \frac{j}{m}\right) \leq i \quad (6.4.7)$$

which as we argued before is sufficient to prove the lemma. \hfill \Box

### 6.5 Lower Bound for Deterministic Algorithms

The following theorem (Theorem 6.5.1) proves the existence of the required hard function. Theorem 6.2.1 for the deterministic case is a simple derivation which we provide after the theorem statement.

**Theorem 6.5.1.** For any integer $k$, any $T > 5k$, and $d > T$ and any $k^{th}$-order deterministic algorithm, there exists a convex function $f^\dagger : B_d \to \mathbb{R}$ for every $d > T$, such that for $T$ steps of the algorithm every point $y \in B_d$ queried by the algorithm is such that

$$f^\dagger(y) \geq \min_{x \in B_d} f^\dagger(x) + \frac{1}{2\sqrt{T}}.$$  

Moreover the function is guaranteed to be $k$-differentiable with Lipschitz constants $L_{i+1}$ bounded as

$$\forall i \leq k \quad L_{i+1} \leq (10k)^i T^{2.5i} \quad (6.5.1)$$

We first prove Theorem 6.2.1 in the deterministic case using Theorem 6.5.1.
Proof of Theorem 6.2.1 Deterministic case. Given an algorithm ALG and numbers \( L_{k+1}, k \) define \( \varepsilon_0(L_{k+1}, k) = L_{k+1}/(10k)^k \). For any \( \varepsilon \ll \varepsilon_0 \) pick a number \( T \) such that
\[
\varepsilon = \frac{L_{k+1}}{2(10k)^kT(2.5k+0.5)}
\]

Let \( f^\dagger \) be the function constructed in Theorem 6.5.1 for parameters \( k, T, ALG \) and define the hard function \( h : B_d \to \mathbb{R} \)
\[
h(x) = \frac{L_{k+1}}{(10k)^kT^{2.5k}} f^\dagger(x)
\]

Note that by the guarantee in Equation (6.5.1) we get that \( h(x) \) is \( k^{th} \)-order smooth with coefficient at most \( L_{k+1} \). Note that since this is a scaling of the original hard function \( f^\dagger \) the lower bound applies directly and therefore ALG cannot achieve accuracy
\[
\frac{L_{k+1}}{2(10k)^kT^{2.5k} \sqrt{T}} = \varepsilon
\]
in less than \( T = c_k \left( \frac{L_{k+1}}{\varepsilon} \right)^{\frac{2}{5k+1}} \) iterations where \( c_k \) is a constant only depending on \( k \). This finishes the proof of the theorem.

We now provide the proof of Theorem 6.5.1

Proof of Theorem 6.5.1 Define the following parameters \( \gamma \triangleq \frac{1}{3\sqrt{T}} \) and \( \delta_T \triangleq \frac{\gamma}{3kT} \).

Consider a deterministic algorithm ALG. Since ALG is deterministic let the first point played by the algorithm be fixed to be \( x_1 \). We now define a series of functions \( f_i^\dagger \) inductively for all \( i = \{1, \ldots T\} \) as follows
\[
X^{1-i} \triangleq \{x_1 \ldots x_i\} \quad f_i^\dagger \triangleq f^\dagger(X^{1-i}, \gamma, k, \delta_T, T) \quad (6.5.2)
\]
\[
Inp_i^x \triangleq \{f_i^\dagger(x_i), \nabla f_i^\dagger(x_i) \ldots \nabla^k f_i^\dagger(x_i)\} \quad x_{i+1} \triangleq ALG(Inp_1^x, \ldots Inp_i^x) \quad (6.5.3)
\]
The above definitions simulate the deterministic algorithm ALG with respect to changing functions $f_i$. $\text{Inp}^p_i$ is the input the algorithm will receive if it queried point $x_i$ and the function was $f_i^p$. $x_{i+1}$ is the next point the algorithm ALG will query on round $i + 1$ given the inputs $\{\text{Inp}^p_1 \ldots \text{Inp}^p_i\}$ over the previous rounds. Note that thus far these quantities are tools defined for analysis. Since ALG is deterministic these quantities are all deterministic and well defined. We will now prove that the function $f_T^\dagger$ defined in the series above satisfies the properties required by the Theorem 6.5.1.

**Bounded Lipschitz Constants** Using Corollary 6.3.4 the fact that $f$ has Lipschitz constant bounded by 1 and that $f_T^\dagger$ is invariant with respect to a $T$ dimensional subspace, we get that the function $f_T^\dagger$ has higher order Lipschitz constants bounded above as

$$\forall i \leq k \quad L_{i+1} \leq \left(\frac{T}{\delta_T}\right)^i \leq \left(10kT^{2.5}\right)^i$$

**Suboptimality**

Let $\{y_0 \ldots y_T\}$ be the points queried by the algorithm ALG when executed on $f_T^\dagger$. We need to show that

$$\forall i \in [1 \ldots T] \quad f_T^\dagger(y_i) \geq \min_{x \in B_d} f_T^\dagger(x) + \frac{1}{2\sqrt{T}} \quad (6.5.4)$$

Equation (6.5.4) follows as a direct consequence of the following two claims.

**Claim 6.5.2.** We have that for all $i \in [1, T]$, $y_i = x_i$ where $x_i$ is defined by Equation (6.5.3).

**Claim 6.5.3.** We have that

$$\forall i \in [1 \ldots T] \quad f_T^\dagger(x_i) \geq \min_{x \in B_d} f_T^\dagger(x) + \frac{1}{2\sqrt{T}}$$

To remind the reader, $x_i$ (Equation (6.5.3)) are variables which were defined by simulating the algorithm on a changing function where as $y_i$ are the points played by the algorithm.
ALG when run on $f_T^t$. Claim 6.5.2 shows that even though $f_T^t$ was constructed using $x_i$ the outputs produced by the algorithm does not change.

Claim 6.5.2 and Claim 6.5.3 derive Equation 6.5.4 in a straightforward manner thus finishing the proof of Theorem 6.5.1.

We now provide the proofs of Claim 6.5.2 and Claim 6.5.3.

**Proof of Claim 6.5.2**  Note that since the algorithm is deterministic $y_1$ is fixed and $y_i$ for $i \geq 2$ is defined inductively as follows.

$$
\text{Inp}_i^y \triangleq \{ f_T^t(y_i), \nabla f_T^t(y_i), \ldots, \nabla^k f_T^t(y_i) \} \quad y_{i+1} = \text{ALG}(\text{Inp}_1^y, \ldots, \text{Inp}_T^y) \quad (6.5.5)
$$

We will prove the claim via strong induction. The base case $x_1 = y_1$ is immediate because ALG is deterministic and therefore the first point queried by it is always the same.

Assume now that the claim holds for all $j \leq i$. Since by definition $2k\delta_T \leq \gamma / T$, we can see as a direct consequence of Lemma 6.4.3 that

$$
\{ \forall j \leq i \ x_j = y_j \} \Rightarrow \{ \forall j \leq i \ \text{Inp}_i^y = \text{Inp}_i^x \} \quad (6.5.6)
$$

where $\text{Inp}_i^x$ is as defined in Equation (6.5.3). Note that $\text{Inp}_i^x$ is the set of derivatives of $f_i^t$ at $x_i$ and $\text{Inp}_i^y$ is the set of derivatives of $f_i^t$ at $y_i$. Also by definition we have that

$$
\{ \forall j \leq i \ \text{Inp}_i^y = \text{Inp}_i^x \} \Rightarrow \{ x_{i+1} = y_{i+1} \}
$$

Putting the above two together we have that $\{ \forall j \leq i \ x_j = y_j \} \Rightarrow \{ x_{i+1} = y_{i+1} \}$ which finishes the induction.

\[\square\]
Proof of Claim 6.5.3  Using Lemma 6.4.3 we have that \( f^i(x_i) = f^i(x_i) \). Further Equation (6.4.6) implies that

\[
f^i(x_i) \geq \left(1 - \frac{i}{T}\right) \gamma - k\delta_T
\]

Now using (6.4.3) using we get that every point in \( \{x_1 \ldots x_T\} \) is such that

\[
f^i_T(x_i) - \min_{x \in B} f^i_T(x) \geq \left(\frac{1}{\sqrt{T}} - \frac{i\gamma}{T} - 2k\delta_T\right) \geq \frac{1}{2\sqrt{T}}
\]

The above follows by the choice of parameters and \( T \) being large enough. This finishes the proof of Claim 6.5.3.

6.6 Lower Bound for Randomized Algorithms

In this section we prove the version of Theorem 6.5.1 for randomized algorithms. The key idea underlying the proof remains the same. However since we cannot simulate the algorithm anymore we choose the vectors \( \{a_i\} \) forming the subspace randomly from \( \mathbb{R}^d \) for a large enough \( d \). This ensures that no algorithm with few queries can discover the subspace in which the function is non-invariant with reasonable probability. Naturally the dimension required for Theorem 6.5.1 now is larger than the tight \( d > T \) we achieved as in the case of deterministic algorithms.

The proof of Theorem 6.2.1 for randomized algorithms follows in a similar way as the proof for the deterministic case using Theorem 6.5.1.

**Theorem 6.6.1.** For any integer \( k \), any \( T > 5k \), \( \delta \in [0, 1] \), and any \( k \)-order (potentially randomized algorithm), there exists a \( k \)-differentiable convex function \( f^\dagger : B_d \to \mathbb{R} \) for \( d = \Omega(T^3 \log(T^2/\delta)) \), such that with probability at least \( 1 - \delta \) (over the randomness of the algorithm) for \( T \) steps of the algorithm every point \( y \) queried by the algorithm is such that

\[
f^\dagger(y) \geq \min_{x \in B_d} f^\dagger(x) + \frac{1}{2\sqrt{T}}.
\]
Moreover the function $f^\dagger$ is guaranteed to be $k$-differentiable with Lipschitz constants $L_i$ bounded as

$$\forall i \leq k \quad L_{i+1} \leq (20kT^{2.5})^i$$

**Proof.** We provide a randomized construction for the function $f^\dagger$. The construction is the same as in Section 6.4 but we repeat it here for clarity. We sample a random $T$ dimensional orthonormal basis $\{a_1 \ldots a_T\}$. Let $A_i$ be the subspace spanned by $\{a_1 \ldots a_i\}$ and $A_i^\perp$ be the orthogonal complement of $A_i$. Further define an auxiliary function

$$f(x) \triangleq \max_i f_i(x) \text{ where } f_i(x) \triangleq a_i^T x$$

Given a parameter $\gamma$, now define the following functions

$$\tilde{f}(x) \triangleq \max_i \tilde{f}_i(x) \text{ where } \tilde{f}_i(x) \triangleq f_i(x) + \left(1 - \frac{i}{T}\right) \gamma + a_i^T x$$

$$f^\dagger(k, \gamma, \delta_T) \triangleq S_{\delta_T, A_T}^k \tilde{f}$$

(6.6.1)

i.e. smoothing $\tilde{f}$ with respect to $\delta_T, A_T$. The hard function we propose is the random function $f^\dagger$ with parameters set as $\gamma = \frac{1}{3\sqrt{T}}$ and $\delta_T = \frac{1}{20kT^{1.5}}$. We restate facts which can be derived analogously to those derived in Section 6.4 (c.f. Equations (6.4.2), (6.4.3)).

$$\forall x \quad |f^\dagger(x) - \tilde{f}(x)| \leq k\delta \quad \text{and} \quad \min_{x \in B_d} f^\dagger(x) \leq \frac{-1}{\sqrt{T}} + \gamma + k\delta$$

(6.6.2)

The following key lemma will be the main component of the proof.

**Lemma 6.6.2.** Let $\{x_1 \ldots x_T\}$ be the points queried by a randomized algorithm throughout its execution on the function $f^\dagger$. With probability at least $1 - \delta$ (over the randomness of the algorithm and the selection of $f^\dagger$) the following event $\mathcal{E}$ happens

$$\mathcal{E} = \left\{ \forall i \in [T] \quad \forall j \geq i \quad |a_j^T x_i| \leq \frac{1}{20T^{1.5}} \right\}$$

187
Using the above lemma we first demonstrate the proof of Theorem 6.6.1. We will assume the event $E$ in Lemma 6.6.2 happens.

**Bounded Lipschitz Constants** Using Corollary 6.3.4, the fact that $f^\dagger$ has Lipschitz constant bounded by 1 and that $\tilde{f}$ is invariant with respect to the $T$ dimensional subspace $A_T$, we get that the function $f^\dagger_T$ has higher order Lipschitz constants bounded above as

$$\forall i \leq k \quad L_{i+1} \leq \left( \frac{T}{\delta_T} \right)^i \leq (20kT^{2.5})^i$$

**Sub-optimality** : The event $E$ in the lemma implies that $\tilde{f}_i(x_i) \geq -\frac{1}{20T^{1.5}} + (1 - \frac{i}{T})\gamma$ which implies that $\tilde{f}(x_i) \geq -\frac{1}{20T^{1.5}} + (1 - \frac{i}{T})\gamma$ and from Equation (6.6.2) we get that

$$\forall i \in [T] \quad f^\dagger(x_i) \geq -\frac{1}{20T^{1.5}} + \left(1 - \frac{i}{T}\right)\gamma - k\delta_T$$

Now using Equation (6.6.2) we get that every $x_i$ is such that

$$\forall i \in [T] \quad f^\dagger(x_i) - \min_{x \in B} f^\dagger(x) \geq \frac{1}{\sqrt{T}} - \frac{i}{T} \gamma - 2k\delta_T - \frac{1}{20T^{1.5}} \geq \frac{1}{2\sqrt{T}}$$

The last inequality follows by the choice of parameters. This finishes the proof of Theorem 6.6.1.

---

**Proof of Lemma 6.6.2** We will use the following claims to prove the lemma. For any vector $x$, define the event $E_i(x) = \{ \forall j \geq i \mid |a_j^T x| \leq \frac{1}{20T^{1.5}} \}$. The event we care about then is

$$\mathcal{E} \triangleq \bigcap_{i=1 \rightarrow T} E_i(x_i)$$

**Claim 6.6.3.** Let $x \in \mathbb{R}^d$. If $E_i(x)$ holds, then $[f^\dagger(x), \nabla f^\dagger(x) \ldots \nabla^k f^\dagger(x)]$ all depend only on $\{a_1 \ldots a_i\}$. 

188
Claim 6.6.4. Let $i \in [T]$. Assume that $\forall j < i$, $\mathcal{E}_j(x_j)$ holds. Then $\mathcal{E}_i(x_i)$ holds with probability at least $1 - \frac{\delta}{T}$ (over the choice of $a_i$ and the randomness of the algorithm).

Claim 6.6.3 is a robust version of the argument presented in the proof of Theorem 6.5.1. Claim 6.6.4 is a byproduct of the fact that in high dimensions the correlation between a fixed vector and a random small basis is small. Claim 6.6.3 is used to prove Claim 6.6.4.

Lemma 6.6.2 now follows via a simple inductive argument using Claim 6.6.4 which is as follows

$$
\Pr(\mathcal{E}) = \Pr \left( \bigcap_{i=1\rightarrow T} \mathcal{E}_i(x_i) \right) = \prod_{i=1\rightarrow T} \Pr \left( \mathcal{E}_i(x_i) \bigcap \mathcal{E}_j(x_j) \right) \geq \left( 1 - \frac{\delta}{T} \right)^T \geq 1 - \delta
$$

Proof of Claim 6.6.3 As noted before the smoothing operator $S^k_\delta$ is such that at any point $x$ all the $k$ derivatives of $S^k_\delta f$ depend only on the value of $f$ in a ball of radius $k\delta$ around the point $x$. Therefore it is sufficient to show that for the function $\tilde{f}$, for every $y$ such that $\|y - x\| \leq k\delta_T$ we have that $\tilde{f}(y)$ depends only on $\{a_1 \ldots a_i\}$. To ensure this, it is enough to ensure that for every such $y$ we have that $\arg\min_{j \in [T]} \tilde{f}_j(y) \leq i$ which is what we prove next.

Let us first note the following facts. By the definition of $\mathcal{E}_i(x)$ we have that $\forall j > i$, $f_j(x) \leq \frac{1}{20T^{1.5}}$. This immediately implies that

$$
\max_{j > i} \tilde{f}_j(x_i) \leq \frac{1}{20T^{1.5}} + \left( 1 - \frac{i + 1}{T} \right) \gamma \tag{6.6.3}
$$

Now since we know each $\tilde{f}_i$ is 1-Lipschitz$^4$ this also gives us

$$
\forall y \text{ s.t. } \|y - x\| \leq k\delta_T \text{ we have } \max_{j > i} \tilde{f}_j(y) \leq \frac{1}{20T^{1.5}} + \left( 1 - \frac{i + 1}{T} \right) \gamma + k\delta_T \tag{6.6.4}
$$

$^4\|a_i\| = 1$
By the event $\mathcal{E}_i(x)$ we also know that $\tilde{f}_i(x) \geq -\frac{1}{20T^{1.5}} + (1 - \frac{i}{T})\gamma$. This implies as above

$$\forall y \text{ s.t. } \|y - x\| \leq k\delta_T \quad \text{we have} \quad \max_{j \leq i} \tilde{f}_j(y) \geq -\frac{1}{20T^{1.5}} + \left(1 - \frac{i}{T}\right)\gamma - k\delta_T \quad (6.6.5)$$

The above equations imply that as long as $2k\delta_T + \frac{1}{10T^{1.5}} < \frac{\gamma}{T}$ (which is true by the choice of parameters), we have that

$$\forall y \text{ s.t. } \|y - x\| \leq k\delta_T \quad \arg\min_{j \in [i]} \tilde{f}_j(y) \leq i \quad (6.6.6)$$

which is sufficient to prove Claim 6.6.3.

**Proof of Claim 6.6.4** Consider any $i \in [T]$. Given $\mathcal{E}_j(x_j)$ is true for all $j < i$, applying Claim 6.6.3 for all $j < i$, implies that all the information that the algorithm possesses is only a function of $\{a_1 \ldots a_{i-1}\}$ and the internal randomness of the algorithm. Since $\{a_1 \ldots a_T\}$ was sampled from the uniform distribution on the orthonormal group, we can assume that it was sampled by the inductive process which picks $a_i$ as a uniformly random unit vector from the subspace $A_{i-1}^\perp$, which is defined to be the orthogonal component of $A_{i-1}$, the subspace spanned by $\{a_1, \ldots, a_{i-1}\}$.

The above inductive procedure for sampling implies that conditioned on $\{a_1 \ldots a_{i-1}\}$, the distribution of the remaining vectors $\{a_i \ldots a_T\}$ is uniform on the orthogonal group $O(T - i + 1)$, lying in the $d - i + 1$-dimensional subspace $A_{i-1}^\perp$. The above implies that the distribution over $\{a_i \ldots a_T\}$ is conditionally independent of any $x_i$ the algorithm might play. Since we wish to bound the absolute value of the inner product we can assume $\|x_i\| = 1$.

Following the above arguments, proving the lemma now reduces to the following. Consider a fixed unit vector $y$ in a $\mathbb{R}^{d-i+1}$. The vector $y$ corresponds to the vector $x_i$ played by the algorithm. Further, consider picking a uniformly random $T - i + 1$ dimensional subspace of $\mathbb{R}^{d-i+1}$ represented by the basis $\{y_1 \ldots y_{T-i+1}\}$. $\{y_1 \ldots y_{T-i+1}\}$ represent the

---

5Otherwise the absolute value of the inner product is only lower
vectors \{a_i \ldots a_T\}. The lemma now reduces to bounding \( \forall j \), the probability

\[
\Pr \left( |\langle y, y_j \rangle| > \frac{1}{20T^{1.5}} \right).
\]

The rest of the argument follows the argument by [143](Proof of Lemma 7). Note that for \( y_1 \) this probability amounts to the ratio between the surface area of a sphere above the caps of radius \( \sqrt{1 - \left(\frac{1}{20T^{1.5}}\right)^2} \) and the surface area of the unit sphere. This surface area is smaller than the ratio between the surface area of a sphere of radius \( \sqrt{1 - \left(\frac{1}{20T^{1.5}}\right)^2} \) and the surface area of a unit sphere. Formally this gives us

\[
\Pr \left( |y_1^T y| \geq \frac{1}{20T^{1.5}} \right) \leq \sqrt{\left(1 - \left(\frac{1}{20T^{1.5}}\right)^2\right)^{d-i+1}}
\]

Applying the argument inductively we get that

\[
\forall j \in [1, T - i + 1] \quad \Pr \left( |y_j^T y| \geq \frac{1}{20T^{1.5}} \right) \leq \sqrt{\left(1 - \left(\frac{1}{20T^{1.5}}\right)^2\right)^{d-i-j+2}}
\]

Using the union bound we have that

\[
\Pr(\mathcal{E}_i(x_i)) \geq 1 - \left( \Pr \left( \bigcup_{j=1-i+1} \left( |y_j^T y| \geq \frac{1}{20T^{1.5}} \right) \right) \right) \\
\geq 1 - (T - i) \left( 1 - \left(\frac{1}{20T^{1.5}}\right)^2 \right)^{d-i+1} \\
\geq 1 - (T - i) e^{-\left(\frac{1}{20T^{1.5}}\right)^2 \frac{d-T}{2}} \geq 1 - \frac{\delta}{T}
\]

The last line follows from the choice of \( d = \Omega \left( T^3 \log(T^2/\delta) \right) \). □
6.7 Conclusion

We have considered the problem of achieving dimension free polynomial time algorithms for minimizing convex functions where the function is guaranteed to be $k$-differentiable and $k^{th}$-order smooth and the algorithm is allowed to have access to $k$ derivatives at every iteration. We showed an oracle complexity lower bound for convex optimization under these conditions demonstrating that the number of points queried by any deterministic/randomized algorithm should have at least an inverse polynomial dependence on the desired error. This rules out linearly-converging derivative-based algorithms even under these assumptions.

While we provide precise guarantees for the dependence on the exponent, our bounds are weaker than those proved independently and concurrently by [22] (which only applies to deterministic algorithms). We believe that our construction (or potentially a similar one) might be able to achieve the improved bounds and leave this direction as immediate future work. Furthermore we remark that while the known upper and lower bounds are tight for first and second order optimization, they are not tight for $k > 2$ and this is an intriguing open question.
Bibliography


