Large-Scale Optimization Methods for Data-Science Applications

by

Haihao Lu

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Mathematics and Operations Research
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

Signature redacted

Author ..................................................

Department of Mathematics and Operations Research Center

May 6, 2019

Signature redacted

Certified by.....

Robert M. Freund

Theresa Seley Professor in Management Science

Thesis Supervisor

Signature redacted

Accepted by .................................

Jonathan Kelner

Chairman, Department of Mathematics Committee on Graduate Thesis

Accepted by ..............

Dimitris Bertsimas

Boeing Professor of Operations Research

Co-director, Operations Research Center
Large-Scale Optimization Methods for Data-Science Applications

by

Haihao Lu

Submitted to the Department of Mathematics and Operations Research Center on Apr 26, 2019, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mathematics and Operations Research

Abstract

In this thesis, we present several contributions of large scale optimization methods with the applications in data science and machine learning.

In the first part, we present new computational methods and associated computational guarantees for solving convex optimization problems using first-order methods. We consider general convex optimization problem, where we presume knowledge of a strict lower bound (like what happened in empirical risk minimization in machine learning). We introduce a new functional measure called the growth constant for the convex objective function, that measures how quickly the level sets grow relative to the function value, and that plays a fundamental role in the complexity analysis. Based on such measure, we present new computational guarantees for both smooth and non-smooth convex optimization, that can improve existing computational guarantees in several ways, most notably when the initial iterate is far from the optimal solution set.

The usual approach to developing and analyzing first-order methods for convex optimization always assumes that either the gradient of the objective function is uniformly continuous (in the smooth setting) or the objective function itself is uniformly continuous. However, in many settings, especially in machine learning applications, the convex function is neither of them. For example, the Poisson Linear Inverse Model, the $D$-optimal design problem, the Support Vector Machine problem, etc. In the second part, we develop a notion of relative smoothness, relative continuity and relative strong convexity that is determined relative to a user-specified “reference function” (that should be computationally tractable for algorithms), and we show that many differentiable convex functions are relatively smooth or relatively continuous with respect to a correspondingly fairly-simple reference function. We extend the mirror descent algorithm to our new setting, with associated computational guarantees.

Gradient Boosting Machine (GBM) introduced by Friedman is an extremely powerful supervised learning algorithm that is widely used in practice — it routinely fea-
tures as a leading algorithm in machine learning competitions such as Kaggle and the KDDCup. In the third part, we propose the Randomized Gradient Boosting Machine (RGBM) and the Accelerated Gradient Boosting Machine (AGBM). RGBM leads to significant computational gains compared to GBM, by using a randomization scheme to reduce the search in the space of weak-learners. AGBM incorporate Nesterov’s acceleration techniques into the design of GBM, and this is the first GBM type of algorithm with theoretically-justified accelerated convergence rate. We demonstrate the effectiveness of RGBM and AGBM over GBM in obtaining a model with good training and/or testing data fidelity.

Thesis Supervisor: Robert M. Freund
Title: Theresa Seley Professor in Management Science
Acknowledgments

First and foremost, I would like to thank my advisor, Rob Freund. Rob has been a tremendous mentor and perfect role model for me. During my PhD time, Rob not only guided me into the research area in optimization, taught me how to conduct research step-by-step, but also showed me the academic integrity. I feel truly blessed to have Rob as my advisor, and I hope to be as generous and encouraging with my future students as Rob was with me.

I feel deeply indebted to my undergraduate advisor, David Cai, who brought me to the field of broad mathematical science. Without him, I would never have been able to pursue a PhD at MIT, and I would have missed the past five wonderful years as well as future academic life.

I would like to thank several other people who have had a significant impact for the work appeared in this thesis and my professional development during my PhD time. Thanks especially Rahul Mazumder for motivating me to study and work on statistical machine learning, and for funding me for my last year. There are never lack of interesting ideas we can work together, and I have enjoyed our broad discussions on research and life. I was very fortunate to have the opportunity to collaborate with Yurii Nesterov. Every meeting/chat with Yurii can inspire me with deeper understandings of old works or with new research directions.

I would like to thank my collaborators and co-workers at Google. I really cherish my experience at Google working on computational tools and real applications. In particular, I am very grateful for Vahab Mirrokni’s help for referring me to Google and for helping me find interesting projects at Google. I am very grateful for Miles Lubin’s guidance on how to do high-quality coding with Google’s standard. And I learnt a lot on large scale machine learning problems from Natalia Ponomareva.

I would like to thank my great thesis committee, Rob Freund, Rahul Mazumder, Ankur Moitra and Jon Kelner, for their valuable advices and comments. In addition, I would also like to thank Suvrit Sra, Asu Ozdaglar and Alex Townsend to serve
my general exam committee in Maths Department or my general exam committee in Operations Research Center.

I am extremely fortunate to have the chance to interact with many great faculty members at MIT, from whom I have learnt a lot. Thanks to Dimitris Bertsimas, David Gamarnik, Georgia Perakis, David Simchi-Levi, Juan Pablo Vielma, Pablo Parrilo, Dimitri Bertsekas, Asu Ozdaglar, Philippe Rigollet, Nikos Trichakis. I would also like to thank Barbara Peskin, Michele Gallarelli, Laura Rose, Andre Carvalho and David Merrill, the Maths and ORC staff members who helped me along the way.

I am very grateful to all the friends who have supported me – Ruixun, Chenjie, Sylvaner, Ben, Li, Yuchen, Will, Hanzhang, Hong, Paul, Qiang, Brad, Scott, Jing, Yimin, Jane, Thao, Kenji, Louis, among many others. They have become an essential part for my life, research and study during the past five years.

Last but by no means least, I want to dedicate this thesis to my family family – my father Dianzhao Lu, my mother Caie Liu, and my wife Ruiting Liu – for their endless and unconditional love and support.
## Contents

1 Introduction 17

2 New Computational Guarantees for Solving Convex Optimization Problems with First Order Methods, via a Function Growth Condition Measure 21

2.1 Problem Statement and Overview of Results .......................... 21

2.1.1 Problem Statement, Strict Lower Bound, and Function Growth Constant ........................................... 21

2.1.2 Overview of Results .................................................. 26

2.2 Review of Subgradient Descent and an Accelerated Gradient Method 28

2.2.1 Subgradient Descent ................................................ 28

2.2.2 Accelerated Gradient Method for Smooth Optimization ....... 30

2.3 Computational Guarantees when \( f(\cdot) \) is Non-Smooth ......... 31

2.3.1 Subgradient Descent using Two Step-Size Rules Running Simultaneously ............................................. 31

2.3.2 Subgradient Descent when \( f^* \) is known ......................... 41

2.3.3 Non-Smooth Optimization using a New Smooth Approximations Method .................................................. 44

2.4 Computational Guarantees when \( f(\cdot) \) is Smooth ................ 53
3 Relatively Smooth Convex Optimization by First-Order Methods, and Applications

3.1 Introduction, Definition of “Relative-Smoothness,” and Basic Properties 63

3.1.1 Traditional Set-up for Smooth First-Order Methods ............. 63
3.1.2 Relative Smoothness and Relative Strong Convexity ........... 67
3.1.3 Constructive Algorithmic Set-up ............................. 70

3.2 Examples of Relatively Smooth Optimization Problems ............ 72

3.2.1 Optimization over $\mathbb{R}^n$ with $\|\nabla^2 f(x)\|$ growing as a polynomial in $\|x\|_2$ ......................... 72
3.2.2 $D$-Optimal Design Problem ................................... 76
3.2.3 Generalized Volumetric Function Optimization .................. 78
3.2.4 Optimization over $Q \subset (0,u]^n$ with $\|\nabla^2 f(x)\|$ growing as a polynomial in $\sum_{i=1}^{n} \frac{1}{x_i}$ ...................... 80

3.3 Computational Analysis for the Primal Gradient Scheme and the Dual Averaging Scheme .................................. 83

3.3.1 Analysis of Primal Gradient Scheme (Algorithm 6) ............ 83
3.3.2 Dual Averaging Scheme and Analysis ............................ 86
3.3.3 On Optimization Problems with a Composite Function ........ 89
3.3.4 Questions: Accelerated Methods, Conjugate Duality, Choosing the Reference Function .................................. 89

3.4 $D$-Optimal Design Revisited: Computational Guarantees using the Primal Gradient or Dual Averaging Scheme .................... 90

4 Relatively Continuous Convex Optimization by First-Order Methods, and Applications 95

4.1 Introduction ......................................................... 95
4.2 Traditional Mirror Descent ........................................... 98
  4.2.1 Deterministic Setting ............................................. 99
  4.2.2 Stochastic Setting ................................................. 100
4.3 Relative Continuity .................................................. 102
  4.3.1 Deterministic Setting ............................................. 103
  4.3.2 Stochastic Setting ................................................ 106
4.4 Computational Analysis for Stochastic Mirror Descent and (Deterministic) Mirror Descent ........................................... 107
4.5 Specifying a Reference Function $h(\cdot)$ with Relative Continuity for Mirror Descent ................................................ 115
  4.5.1 Deterministic Setting ............................................. 115
  4.5.2 Stochastic Setting ................................................ 117
  4.5.3 Relative Continuity for instances of SVM and IEP ........... 118

5 Randomized Gradient Boosting Machine ................................ 123
  5.1 Introduction ............................................................ 123
    5.1.1 Gradient Boosting Machine .................................... 124
    5.1.2 Randomized Gradient Boosting Machine ....................... 126
    5.1.3 Related Literature on Convergence Guarantees for GBM ... 130
    5.1.4 Contributions .................................................... 131
  5.2 Random-then-Greedy Coordinate Descent in the Coefficient Space ... 132
  5.3 Machinery: Structured Norms and Random Selection Rules ...... 135
    5.3.1 Infinity Norm, Slope Norm, Group Norm and a Combined Norm ... 135
    5.3.2 Random-then-Greedy Procedure ................................ 137
  5.4 Computational Guarantees for RGBM ................................ 142
5.4.1 Minimal Cosine Angle .................................................. 144
5.4.2 Computational Guarantees: Strongly Convex Loss Function . 150
5.4.3 Computational Guarantees: Non-Strongly Convex Loss Function155
5.5 Numerical Experiments .................................................. 158

6 Accelerated Gradient Boosting Machine .................................. 161
  6.1 Introduction ........................................................................ 162
  6.1.1 Our contributions ......................................................... 162
  6.1.2 Related Literature .......................................................... 163
  6.2 Gradient Boosting Machine .................................................. 164
  6.2.1 Best Fit Weak Learners .................................................. 165
  6.3 Accelerated Gradient Boosting Machine (AGBM) .................... 167
    6.3.1 Boosting with strong learners ...................................... 167
    6.3.2 Boosting with weak learners ....................................... 169
  6.4 Convergence Analysis of AGBM ........................................... 170
    6.4.1 Assumptions ............................................................. 171
    6.4.2 Computational Guarantees .......................................... 172
  6.5 Extensions and Variants .................................................... 183
    6.5.1 Restart and Linear Convergence .................................... 183
    6.5.2 A Vanilla Accelerated Gradient Boosting Method .............. 184
  6.6 Numerical Experiments .................................................... 185
    6.6.1 Evidence that VAGBM May Diverge .............................. 186
    6.6.2 AGBM Sensitivity to the hyperparameters ...................... 187
    6.6.3 Experiments with Fine Tuning ..................................... 188
6.7 Additional Discussions ........................................... 189
  6.7.1 Line search in Boosting .................................... 190
  6.7.2 Use of Hessian .................................................. 190
  6.7.3 Out-of-sample Performance ................................. 190
6.8 Conclusion .......................................................... 191

A Growth Conditions .................................................. 193
  A.1 Growth Constant $G$ and the Modulus of Weak Sharp Minima ........................ 193
  A.2 Proof of Theorem 2.1.1 ........................................... 194

B Relative Conditions .................................................. 197
  B.1 Solving the subproblem (4.7) when $h(x)$ is a convex function of $\|x\|_2^2$
     and $Q$ has simple constraints ..................................... 197
  B.2 Extension to Composite Optimization .......................... 198
  B.3 Criteria for choosing the reference function $h(\cdot)$ ...................... 200
  B.4 Finite Radius Bound for SVM .................................... 201
List of Figures

2-1 Illustration of $G$ and $f_{slb}$ for a function with multiple optimal solutions. 25

5-1 Plots showing the training [top panel] and testing [bottom panel] loss versus number of RGBM iterations and the associated computational cost (measured by "epochs") for RGBM with different $t$ values. We consider the a9a dataset (for a classification task) from the LIBSVM library (see text for details). A smaller value of $t$ corresponds to a smaller cost per iteration. As expected we see overall computational savings for a value of $t$ that is smaller than the maximum $t = 123$, which corresponds to GBM. ................................. 129

5-2 Figure shows the profiles of $K^{-1}(j)$ (i.e., the probability distribution of the quantile $q = j/K$ for the RtG procedure, as described in the text) as a function of $q$. We consider three profiles (of $K^{-1}(j)$) for three different values of $K$, and the Beta(1,10) density function (we fix $t = 10$). We observe that for $K \approx 40$, the profile of $K^{-1}(j)$ and that of the Beta(1,10) distribution are indistinguishable. ................................. 140

5-3 Illustration of the relationship between $\Theta$ and density of weak-learners in a 2D example. (a), (b), (c) represent poorly-spread weak-learners, moderately spread weak-learners and densely spread weak-learners, respectively. When $\mathcal{F}$ is the infinity norm, the values of $\Theta$ are given by: (a) $\Theta^2 \approx 0$; (b) $\Theta^2 = 1/2$; and (c) $\Theta^2 \approx 0.933$ — the weak-learners are more spread out for higher values of $\Theta$. ................................. 145
5-4 Plot shows how $\Theta$ varies with $p$ (log-log plot) when the weak-learners are orthogonal and $\mathcal{F}$ corresponds to the slope norm with parameter $\gamma = [\gamma_i^f(j)]_j$ (see (5.10)). We show three profiles for three different values of $t$. Note that $\Theta$ is defined only for $p \geq t$. This MCA corresponds to the Type 2 random selection rule.

5-5 Plots showing the training and testing loss versus number of standardized iterations aka epochs for four different datasets. We consider RGBM for different $t$ values (with the largest corresponding to GBM). The general observations are similar to that in Figure 5-1 — we get significant computational savings by using a smaller value of $t$, without any loss in training/testing error.

6-1 Training loss versus number of trees for VAGBM (which doesn’t converge) and AGBM with different parameters $\gamma$.

6-2 Training and testing loss versus number of trees for logistic regression on a la.
List of Tables

3.1 Comparison of the order of computational guarantees for the Frank-Wolfe Method [53], [1] and the Primal Gradient and Dual Averaging Schemes (Theorem 3.4.1 and Remark 3.4.1) for $D$-optimal design. All constants have been suppressed in order to highlight the dependencies on particular quantities of interest. It also follows from [53] that $f(x^0) - f^* \leq m \ln(n/m)$ for $x^0 = (1/n)e$, which can be inserted in the above bounds as well. ........................................... 94

5.1 Basic statistics of the (real) datasets used in numerical experiments. The training/testing datasets are obtained by a 80%/20% (random) split on these sample sizes. ........................................... 158

6.1 List of notations used. ........................................... 167

6.2 Basic statistics of the (real) datasets used. ........................................... 186

6.3 Performance of after tuning hyper-parameters. ........................................... 189
Chapter 1

Introduction

During the past two decades, there has been significant development in machine learning, motivated by a wide array of application domains in the “big data” era. Continuous optimization methods, as the tool for solving the machine learning models, in many cases determine the computational bottlenecks for the capacity of machine learning applications. As the size of the problems arising in machine learning grow dramatically, first-order methods become more appealing mainly due to two reasons: (1) the cost per iteration is low, so that it scales better with the size of the problems, and (2) low-accuracy solutions are good enough as the data usually is noisy.

In this thesis, we present several contributions of first-order methods, and how to improve machine learning algorithms from optimization perspectives. Among several themes in the thesis, we are interested in: (i) gaining better understandings and improving the computational complexity of first-order methods, and (ii) developing new machine learning methods with improved computational complexity.

The results in chapter 2, 3, 4, 5, 6 are included in the papers [36], [61], [62], [66] and [64]. Here we provide a brief summary for each of the remaining chapters of this thesis.

Chapter 2: New Computational Guarantees for Solving Convex Optimization Problems with First Order Methods, via a Function Growth
Condition Measure. Motivated by recent work of Renegar [91], we present new computational methods and associated computational guarantees for solving convex optimization problems using first-order methods. Our problem of interest is the general convex optimization problem $f^* = \min_{x \in Q} f(x)$, where we presume knowledge of a strict lower bound $f_{\text{slb}} < f^*$. [Indeed, $f_{\text{slb}}$ is naturally known when optimizing many loss functions in statistics and machine learning (least-squares, logistic loss, exponential loss, total variation loss, etc.) as well as in Renegar’s transformed version of the standard conic optimization problem [91]; in all these cases one has $f_{\text{slb}} = 0 < f^*$.]

We introduce a new functional measure called the growth constant $G$ for $f(\cdot)$, that measures how quickly the level sets of $f(\cdot)$ grow relative to the function value, and that plays a fundamental role in the complexity analysis. When $f(\cdot)$ is non-smooth, we present new computational guarantees for the Subgradient Descent Method and for smoothing methods, that can improve existing computational guarantees in several ways, most notably when the initial iterate $x^0$ is far from the optimal solution set. When $f(\cdot)$ is smooth, we present a scheme for periodically restarting the Accelerated Gradient Method that can also improve existing computational guarantees when $x^0$ is far from the optimal solution set, and in the presence of added structure we present a scheme using parametrically increased smoothing that further improves the associated computational guarantees.

Chapter 3: Relatively Smooth Convex Optimization by First-Order Methods, and Applications. The usual approach to developing and analyzing first-order methods for smooth convex optimization assumes that the gradient of the objective function is uniformly smooth with some Lipschitz constant $L$. However, in many settings the differentiable convex function $f(\cdot)$ is not uniformly smooth – for example in $D$-optimal design where $f(x) := -\ln \det(HXH^T)$ and $X := \text{Diag}(x)$, or even the univariate setting with $f(x) := -\ln(x) + x^2$. In this chapter we develop a notion of “relative smoothness” and relative strong convexity that is determined relative to a user-specified “reference function” $h(\cdot)$ (that should be computationally tractable for algorithms), and we show that many differentiable convex functions are relatively smooth with respect to a correspondingly fairly-simple reference function.
We extend two standard algorithms— the primal gradient scheme and the dual averaging scheme—to our new setting, with associated computational guarantees. We apply our new approach to develop a new first-order method for the $D$-optimal design problem, with associated computational complexity analysis. Some of our results have a certain overlap with the recent work [8].

Chapter 4: Relatively Continuous Convex Optimization by First-Order Methods, and Applications. The usual approach to developing and analyzing first-order methods for non-smooth (stochastic or deterministic) convex optimization assumes that the objective function is uniformly Lipschitz continuous with parameter $M_f$. However, in many settings the non-differentiable convex function $f$ is not uniformly Lipschitz continuous—for example (i) the classical support vector machine (SVM) problem, (ii) the problem of minimizing the maximum of convex quadratic functions, and even (iii) the univariate setting with $f(x) := \max\{0, x\} + x^2$. Herein we develop a notion of “relative continuity” that is determined relative to a user-specified “reference function” $h$ (that should be computationally tractable for algorithms), and we show that many non-differentiable convex functions are relatively continuous with respect to a correspondingly fairly-simple reference function $h$. We also similarly develop a notion of “relative stochastic continuity” for the stochastic setting. We analyze two standard algorithms—the (deterministic) mirror descent algorithm and the stochastic mirror descent algorithm—for solving optimization problems in these new settings, providing the first computational guarantees for instances where the objective function is not uniformly Lipschitz continuous.

Chapter 5: Randomized Gradient Boosting Machine. Gradient Boosting Machine (GBM) introduced by Friedman [41] is an extremely powerful supervised learning algorithm that is widely used in practice—it routinely features as a leading algorithm in machine learning competitions such as Kaggle and the KDDCup. In spite of the usefulness of GBM in practice, our current theoretical understanding of this method is rather limited. In this chapter, we propose Randomized Gradient Boosting Machine (RGBM) which leads to significant computational gains compared
to GBM, by using a randomization scheme to reduce the search in the space of weak-learners. We derive novel computational guarantees for RGBM. We also provide a principled guideline towards better step-size selection in RGBM that does not require a line search. Our proposed framework is inspired by a special variant of coordinate descent that combines the benefits of randomized coordinate descent and greedy coordinate descent; and may be of independent interest as an optimization algorithm. As a special case, our results for RGBM lead to superior computational guarantees for GBM. Our computational guarantees depend upon a curious geometric quantity that we call Minimal Cosine Angle, which relates to the density of weak-learners in the prediction space. On a series of numerical experiments on real datasets, we demonstrate the effectiveness of RGBM over GBM in terms of obtaining a model with good training and/or testing data fidelity with a fraction of the computational cost.

**Chapter 6: Accelerating Gradient Boosting Machine.** In this chapter, we propose Accelerated Gradient Boosting Machine (AGBM) by incorporating Nesterov’s acceleration techniques into the design of GBM. The difficulty in accelerating GBM lies in the fact that weak (inexact) learners are commonly used, and therefore the errors can accumulate in the momentum term. To overcome it, we design a “corrected pseudo residual” and fit best weak learner to this corrected pseudo residual, in order to perform the z-update. Thus, we are able to derive novel computational guarantees for AGBM. This is the first GBM type of algorithm with theoretically-justified accelerated convergence rate. Finally we demonstrate with a number of numerical experiments the effectiveness of AGBM over conventional GBM in obtaining a model with good training and/or testing data fidelity.

At the end of this chapter, I will give a brief overview on the other works I did during my PhD time but are not included in this thesis. In [63], we study ways to accelerate greedy coordinate descent in theory and in practice. In [65], we showed local minima are global for deep linear models. In [108], we propose efficient algorithms for leave-one-out cross-validation for high-dimensional non-differentiable learning problems. In [3], we propose a near-optimal bidding strategy in online advertisement.
Chapter 2

New Computational Guarantees for Solving Convex Optimization Problems with First Order Methods, via a Function Growth Condition Measure

2.1 Problem Statement and Overview of Results

2.1.1 Problem Statement, Strict Lower Bound, and Function Growth Constant

Motivated by recent work of Renegar [91], we present new computational methods and associated computational guarantees for solving convex optimization problems using
first-order methods. Our problem of interest is the following optimization problem:

$$P: f^* := \min_{x \in \mathbb{R}} f(x),$$

where $Q \subseteq \mathbb{R}^n$ is a closed convex set and $f(\cdot): Q \to \mathbb{R}$ is a convex function. Let the set of optimal solutions of (4.1) be denoted as $\text{Opt} := \{x \in Q : f(x) = f^*\}$. For $x \in Q$, let $\text{Dist}(x, \text{Opt})$ denote the distance from $x$ to the set of optimal solutions, namely $\text{Dist}(x, \text{Opt}) := \min_y \{\|y - x\| : y \in \text{Opt}\}$.

**Strict Lower Bound $f_{\text{slb}}$ and Function Growth Constant $G$.** Let $f_{\text{slb}}$ be a known and given strict lower bound on the optimal value $f^*$ of (4.1), namely $f_{\text{slb}} < f^*$. Such a known strict lower bound arises naturally when optimizing many loss functions in statistics and machine learning (least-squares loss, logistic loss, exponential loss, total variation loss, etc.) perhaps with the addition of a regularization term; in all these cases $f_{\text{slb}} = 0 < f^*$. A known strict lower bound also arises in Renegar’s transformed version of the standard conic optimization problem [91].

Let $\varepsilon' > 0$ be given. Given the knowledge of the strict lower bound $f_{\text{slb}}$, it is natural to work with the notion of a relative measure of optimality. Let us define an $\varepsilon'$-relative solution of (4.1) to be a point $\hat{x}$ that satisfies:

$$\hat{x} \in Q \quad \text{and} \quad \frac{f(\hat{x}) - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon'.$$

Note that (2.2) is a relative error measure, relative to the optimal bound gap $f^* - f_{\text{slb}}$. We focus on an $\varepsilon'$-relative solution rather than on an $\varepsilon$-absolute solution ($f(\hat{x}) \leq f^* + \varepsilon$), as the former seems more natural in the setting where a strict lower bound is part of the problem description. Indeed, consider the context of loss functions $f(\cdot)$ in statistics and machine learning where $f_{\text{slb}} = 0$, in which case an $\varepsilon'$-relative solution $\hat{x}$ corresponds to $\frac{f(\hat{x})}{f^*} \leq (1 + \varepsilon')$, and hence is a multiplicative measure of optimality tolerance.
Let $G$ denote the smallest scalar $\bar{G}$ satisfying:

$$\text{Dist}(x, \text{Opt}) \leq \bar{G} \cdot (f(x) - f_{\text{sib}}) \text{ for all } x \in Q.$$ (2.3)

By its definition one sees that $G$ measures how fast the distances from the optimal solution set $\text{Opt}$ grow relative to the bound gap $f(x) - f_{\text{sib}}$. Therefore $G$ is a measure of the growth rate of the level sets of $f(\cdot)$. We call $G$ the “growth constant” of the function $f(\cdot)$ for the given strict lower bound $f_{\text{sib}}$. Note that an equivalent definition of $G$ is given by:

$$G = \sup_{x \in Q} \left\{ \frac{\text{Dist}(x, \text{Opt})}{f(x) - f_{\text{sib}}} \right\}. \quad (2.4)$$

Unlike the strict lower bound $f_{\text{sib}}$, we do not assume that $G$ is known, nor do we need any upper bounds on $G$. Indeed, neither knowledge of $G$ nor the finiteness of $G$ are needed in order to implement the computational methods presented herein; however the finiteness of $G$ is needed for the analysis of the methods to be meaningful.

We will see in Sections 2.3 and 2.4 that the knowledge of the fixed strict lower bound $f_{\text{sib}}$ and the concept of the function growth constant $G$ lead to different versions of first-order methods with different computational guarantees than the traditional analysis of first-order methods would dictate. Furthermore, these different computational guarantees can dominate the traditional guarantees in many cases but most notably when the initial iterate $x^0$ is far from the optimal solution. Roughly speaking, for several of the algorithms developed herein our computational guarantees grow like $\ln(1 + \text{Dist}(x^0, \text{Opt}))$ in contrast to traditional guarantees where the growth is proportional to $\text{Dist}(x^0, \text{Opt})$ and $\text{Dist}(x^0, \text{Opt})^2$ (in the smooth and nonsmooth settings, respectively).

In a departure from typical optimization approaches to lower bounds such as those arising from duality theory wherein one desires as tight a lower bound as possible, herein the lower bound $f_{\text{sib}}$ is strict, namely $f_{\text{sib}} < f^*$, and it is fixed, i.e., it is not updated as part of a computational procedure. It is best to think of this lower bound as a structural lower bound that is easily connected to known properties of
the function $f(\cdot)$. Such a strict lower bound on $f(\cdot)$ arises naturally in the settings of statistics and machine learning in the case of loss functions and/or regularization functions, see for example [47]. Consider when $f(\cdot)$ is the logistic loss function $f(x) = \frac{1}{m} \sum_{i=1}^{m} \ln (1 + e^{-Ax})$ or the exponential loss function $f(x) = \ln \left( \sum_{i=1}^{m} e^{-Ax} \right)$, perhaps with the addition of a regularization term $\lambda \|x\|_p^r$ for some $p \geq 1$, $r \geq 1$, and $\lambda \geq 0$. If the sample data is not strictly separable, which translated herein means that there is no $x$ satisfying $Ax \geq 0$ unless $Ax = 0$, then it follows that $f^* > 0$ and so $f_{slb} := 0$ is a strict lower bound and is quite natural in this setting. Another example is regularized least-squares regression such as the LASSO and its cousins, wherein $f(\beta) = \frac{1}{2} \|y - X\beta\|^2 + \lambda \|\beta\|^r_p$; it follows that $f^* \geq 0$ and one can assert that $f^* > 0 =: f_{slb}$ under a variety of mild assumptions involving either $\lambda$ or the data matrix $X$. Other classes of examples for which $f_{slb} = 0$ is a strict lower bound on $f^*$ include total variation (TV) loss functions which are used in image de-noising, as well as the broad class of minimum norm problems in general, under mild assumptions. Another class of problems for which there is a natural strict lower bound on $f^*$ is the class of projectively transformed conic convex optimization problems under a particularly clever projective transformation, as developed by Renegar [91]; indeed it was this problem class and the results in [91] that gave rise to the line of research described herein.

We can interpret $G$ as connected to a lower estimator of $f(\cdot)$: rearranging (2.3), we obtain:

$$f(x) \geq \bar{f}(x) := f_{slb} + G^{-1} \Dist(x, \Opt) \text{ for all } x \in Q .$$  \hspace{1cm} (2.5)

Therefore the convex function $\bar{f}(x) = f_{slb} + G^{-1} \Dist(x, \Opt)$ is a lower estimator of the function $f(\cdot)$ on $Q$. This interpretation is illustrated in Figure 2-1. As Figure 2-1 illustrates, the concept of the growth constant $G$ is somewhat related to the notion of the modulus of weak sharp minima for (4.1), see Polyak [85] and Burke and Ferris [23]; this relationship is discussed further in Appendix A.1.
A natural question to ask is under what circumstances is the growth constant $G$ finite? Roughly speaking, it holds that $G$ is finite except when the objective function level sets are ill-behaved relative to their recession cone. This is made precise in the following theorem, whose proof is given in Appendix A.2. For $\varepsilon > 0$, let
\[ \text{Opt}_\varepsilon := \{ x \in Q : f(x) \leq f^* + \varepsilon \} \]
denote the $\varepsilon$-optimal level set of $f(\cdot)$ on $Q$, and let $S$ denote the recession cone of $\text{Opt}_\varepsilon$, namely
\[ S := \{ d \in \mathbb{R}^n : x + \theta d \in Q \text{ and } f(x + \theta d) \leq f^* + \varepsilon, \text{ for all } x \in \text{Opt}_\varepsilon \text{ and } \theta \geq 0 \}. \]
Note that $S$ is the (common) recession cone of $\text{Opt}_\varepsilon$ for all $\varepsilon \geq 0$.

**Theorem 2.1.1.** Suppose that for some $\varepsilon > 0$ there exists a bounded set $E_\varepsilon$ for which $\text{Opt}_\varepsilon \subset E_\varepsilon + S$ where $S$ is the recession cone of $\text{Opt}_\varepsilon$. Then for any given strict lower bound $f_{\text{slb}} < f^*$, the growth constant $G$ is finite. \( \square \)

Let us briefly examine special cases of Theorem 2.1.1. Consider the case when $\text{Opt} = E + T$ where $E$ is a bounded convex set and $T$ is a subspace. Then for any $\varepsilon > 0$ it is easy to show that $\text{Opt}_\varepsilon = E_\varepsilon + T$ for some bounded set $E_\varepsilon$, in which case Theorem 2.1.1 implies that $G$ is finite. In particular, when $\text{Opt}$ itself is a bounded set, then we can set $T = \{0\}$, and so Theorem 2.1.1 implies that $G$ is finite.
For an example wherein $G = \infty$, consider the function $f(x_1, x_2) := \frac{x_2^2}{x_1}$ on $Q := \{(x_1, x_2) : x_1 \geq 1\}$. It is straightforward to check that the Hessian matrix $\nabla^2 f(x)$ is positive semidefinite on $Q$ and hence $f(\cdot)$ is convex on $Q$. We have $f^* = 0$ and $\text{Opt} = \{(x_1, 0) : x_1 \geq 1\}$. However, the growth constant $G = \infty$ for any strict lower bound $f_{\text{slb}}$, since by letting $(x_1, x_2) = (\beta^2, \beta)$ for any $\beta \geq 1$ we obtain using (2.4) that

$$G \geq \lim_{\beta \to +\infty} \frac{\text{Dist}((\beta^2, \beta), \text{Opt})}{f(\beta^2, \beta) - f_{\text{slb}}} = \lim_{\beta \to +\infty} \frac{\beta}{1 - f_{\text{slb}}} = +\infty.$$ 

### 2.1.2 Overview of Results

We use the knowledge of the fixed strict lower bound $f_{\text{slb}}$ and the concept of the function growth constant $G$ to design and develop computational guarantees for new versions of first-order methods for solving the optimization problem (4.1). In Section 2.3 we present such methods when $f(\cdot)$ is non-smooth and Lipschitz continuous with Lipschitz constant $M$. In Theorem 2.3.1 we present an iteration complexity of $O \left( M^3 G^2 \left[ \ln \left( 1 + \frac{f(x^0) - f^*}{f^* - f_{\text{slb}}} \right) + \frac{1}{(\varepsilon^2)^2} \right] \right)$ for a version of Subgradient Descent that simultaneously runs with two step-sizes and occasional re-starting, which strictly improves the standard computational complexity bound for Subgradient Descent when $x^0$ is a "cold start," i.e., $\text{Dist}(x^0, \text{Opt})$ is large. In the special case when the optimal objective function value $f^*$ is known, Theorem 2.3.2 shows that the standard step-size rule for Subgradient Descent yields the same result. And when $f(\cdot)$ can be smoothed, we present further improved computational guarantees for a new method (Algorithm 4) that successively smooths and restarts the Accelerated Gradient Method, see Theorem 2.3.3 herein.

In Section 2.4 we present computational guarantees for new first-order methods when $f(\cdot)$ is smooth and has Lipschitz gradient with Lipschitz constant $L$. We present a new first-order method (Algorithm 5) based on periodically restarting the Accelerated Gradient Method, that leads to an iteration complexity of $O \left( G\sqrt{L} \left[ \sqrt{f(x^0) - f_{\text{slb}}} + \frac{\sqrt{f^* - f_{\text{slb}}}}{\sqrt{\varepsilon^2}} \right] \right)$ (Theorem 2.4.1), which in many cases can improve the standard computational complexity bound for the Accelerated Gradient Method, most notably when $f(x^0)$ is far
from the optimal value $f^*$ and $\varepsilon'$ is small. And when $f(\cdot)$ has appropriate adjoint structure, we use parametric increased smoothing and restarting of the Accelerated Gradient Method to achieve a further improvement in the above computational guarantee (Theorem 2.4.2).

Algorithm $A$ in Renegar [92] provides an interesting approach to the general convex optimization setting, that bears comparison to the approach and results contained herein – which are also designed for the general convex optimization setting. Both Algorithm $A$ in [92] and the algorithms herein generalize the methodology for conic optimization developed in Renegar [89, 91] to the general convex optimization problems, but they do so in different ways. Herein the generalization is obtained by introducing the new function measure $G$ based on the strict lower bound $f_{slb}$, while in Algorithm $A$ in [92] the original problem is transformed (implicitly or explicitly) to a conic optimization problem in a slightly lifted space. The resulting algorithms appear to be very different, and have different computational requirements and convergence bounds – Algorithm $A$ in [92] requires a 1-dimensional root finding procedure each iteration, whereas Algorithm 3 herein requires orthogonal projection onto the feasible region. (And indeed it is rather remarkable that Algorithm $A$ of [92] does not require such projection.) Algorithm $A$ does not need a Lipschitz constant; however in the case of a smooth objective function Algorithm $A$ cannot take advantage of such smoothness, unlike Algorithm 5 (and also Algorithm 4) herein.

The paper is organized as follows. Section 2.2 contains a brief review of the Subgradient Descent and an Accelerated Gradient Method. Section 2.3 contains first-order methods and computational guarantees when $f(\cdot)$ is non-smooth. Section 2.4 contains first-order methods and computational guarantees when $f(\cdot)$ is smooth.

**Notation.** Unless otherwise specified, the norm is the Euclidean (inner product) norm $\|x\| := \sqrt{x^T x}$. We occasionally refer to the $\ell_p$ norm of a vector $v$, which is denoted by $\|v\|_p$. For $Q \subset \mathbb{R}^n$, let $\Pi_Q(\cdot)$ denote the Euclidean projection operator onto $Q$, namely $\Pi_Q(x) := \arg \min_{y \in Q} \|y - x\|$. We define $\text{Dist}(x, S) := \min_y \{\|x - y\| : y \in S\}$. The set of optimal solutions of (4.1) is denoted by $\text{Opt} := \{x \in Q : f(x) = f^*\}$. 27
2.2 Review of Subgradient Descent and an Accelerated Gradient Method

We briefly review the Subgradient Descent Method and an Accelerated Gradient Method (as analyzed in Tseng [105]) for solving the convex optimization problem (4.1).

2.2.1 Subgradient Descent

Recall that \( g \) is a subgradient of \( f(\cdot) \) at \( x \) if the following subgradient inequality holds:

\[
    f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y \in Q.
\]

Let \( \partial f(x) \) denote the set of subgradients of \( f(\cdot) \) at \( x \). Here we assume that \( f(\cdot) \) is Lipschitz continuous on a relatively open set \( \hat{Q} \) containing \( Q \), namely, there is a scalar \( M \) for which

\[
    |f(y) - f(x)| \leq M||y - x|| \quad \text{for all } x, y \in \hat{Q}. \tag{2.6}
\]

It follows from (2.6) that for all \( x \in Q \) and \( g \in \partial f(x) \) it holds that \( ||g|| \leq M \).

Algorithm 1 presents the standard subgradient scheme. In this method \( x^k \) is the iterate at iteration \( k \), the best objective value among the first \( k \) iterates is \( f^*_b \), and the best iterate among the first \( k \) iterates is \( x^*_b \).

The following theorem summarizes well-known computational guarantees associated with the subgradient descent method.

**Theorem 2.2.1. (Convergence Bounds for Subgradient Descent [86, 76])**

(i) Consider the subgradient descent method (Algorithm 1). Then for all \( k \geq i \geq 0 \), the following inequality holds:

\[
    f^*_b \leq f^* + \frac{\text{Dist}(x^i, \text{Opt})^2 + \sum_{i=1}^k ||g_i||^2 \alpha_i^2}{2 \sum_{i=1}^k \alpha_i} \leq f^* + \frac{\text{Dist}(x^i, \text{Opt})^2 + M^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i}
\]
Algorithm 1 Subgradient Method for Non-Smooth Optimization

**Initialize.** Initialize with $x^0 \in Q$, $f_b^0 \leftarrow f(x^0)$, $x_b^0 \leftarrow x^0$. $i \leftarrow 0$.

At iteration $i$:
1. **Compute Subgradient.** Compute $g_i \in \partial f(x^i)$.
2. **Determine Step-size.** Determine $\alpha_i \geq 0$.
3. **Perform Updates.** Compute 
   \[
   x_{i+1} = \Pi_Q(x^i - \alpha_i g_i), \\
   f_{b_{i+1}} = \min\{f_{b_i}, f(x_{i+1})\}, \\
   x_{b_{i+1}} = \arg\min_{x \in \{x_i^b, x_{i+1}^b\}} \{f(x)\}.
   \]

(ii) Suppose that $f^*$ is known, and let the step-sizes for Algorithm 1 be $\alpha_i = (f(x^i) - f^*)/\|g_i\|^2$. Then for all $k \geq i \geq 0$, the following inequality holds:

\[
f_{b_{k}} \leq f^* + \frac{M \text{Dist}(x^i, \text{Opt})}{\sqrt{k-i} + 1}. \quad \square
\]

Suppose that we seek to bound the number of iterations $N$ of the Subgradient Descent method required to compute an (absolute) $\varepsilon$-optimal solution of (4.1), which is a point $\hat{x} \in Q$ that satisfies $f(\hat{x}) \leq f^* + \varepsilon$. If $\varepsilon > 0$ is given, and the step-sizes are chosen as $\alpha_i = \varepsilon/\|g_i\|^2$, then it follows from part (i) of Theorem 2.2.1 that $f_{b_{N}}^N \leq f^* + \varepsilon$ for all

\[
N \geq \bar{N} := \frac{M^2 \text{Dist}(x^0, \text{Opt})^2}{\varepsilon^2} - 1. \quad (2.7)
\]

If instead we know (or can bound from above) $\text{Dist}(x^0, \text{Opt})$, and the step-sizes are chosen as $\alpha_i = \text{Dist}(x^0, \text{Opt})/(\sqrt{N+1} \|g_i\|)$ where $N$ satisfies (2.7), then it also follows from part (i) of Theorem 2.2.1 that $f_{b_{N}}^N \leq f^* + \varepsilon$. And if $f^*$ is known, then the bound (2.7) is also sufficient to guarantee $f_{b_{N}}^N \leq f^* + \varepsilon$ if the steps-sizes are chosen as in part (ii) of Theorem 2.2.1. Furthermore, it follows from [71] that the bound (2.7) cannot in general be improved in the black-box oracle model of computation with complexity bounds depending only on $M, \text{Dist}(x^0, \text{Opt})$, and $\varepsilon$. In this regard, we note that the dependence on additional parameters, namely the strict lower bound $f_{\text{slb}}$ and the function growth constant $G$, which are used throughout this paper, shows how we can achieve different (and better in many cases) complexity bounds by including...
additional parameters and appropriately amending algorithms and their analysis.

2.2.2 Accelerated Gradient Method for Smooth Optimization

Here we assume that \( f(\cdot) \) is differentiable on an open set containing \( Q \), and that \( \nabla f(\cdot) \) is Lipschitz on \( Q \) with scalar \( L \), namely:

\[
\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\| \quad \text{for all } x, y \in Q .
\]  

(2.8)

Algorithm 2 presents a standard Accelerated Gradient Method as in Tseng [105].

Algorithm 2 Accelerated Gradient Method

**Initialize.** Initialize with \( x^0 \in Q \) and \( z^0 := x^0 \), and \( i \leftarrow 0 \). Define step-size parameters \( \theta_i \in (0, 1] \) recursively by \( \theta_0 := 1 \) and \( \theta_{i+1} \) satisfies \( \frac{1}{\theta_{i+1}} - \frac{1}{\theta_{i}} = \frac{1}{\theta_i^2} \).

At iteration \( i \):

1. **Perform Updates.** Define \( y^i := (1 - \theta_i)x^i + \theta_iz^i \), and compute \( \nabla f(y^i) \),

\[
z^{i+1} \leftarrow \text{arg min}_{x \in Q} \{ f(y^i) + \nabla f(y^i)^T(x - z^i) + \frac{1}{2}\theta_iL(x - z^i)^T(x - z^i) \},
\]

\[
x^{i+1} \leftarrow (1 - \theta_i)x^i + \theta_iz^{i+1} .
\]

For \( \delta \geq f^* \) define the level set \( S_\delta := \{ x \in Q : f(x) \leq \delta \} \). For \( x \in Q \), let \( \text{Dist}(x, S_\delta) \) denote the distance from \( x \) to the level set \( S_\delta \), namely \( \text{Dist}(x, S_\delta) := \min_y \{ \|y - x\| : y \in S_\delta \} \). The following theorem is a computational guarantee for the Accelerated Gradient Method due to Tseng [105].

**Theorem 2.2.2. (Convergence Bound for Accelerated Gradient Method [105]):** Consider the Accelerated Gradient Method (Algorithm 2). Let \( \delta \geq f^* \) and \( S_\delta := \{ x \in Q : f(x) \leq \delta \} \). Then for all \( k \geq 0 \), the following inequality holds:

\[
f(x^k) - \delta \leq \frac{2L\text{Dist}(x^0, S_\delta)^2}{(k+1)^2} . \quad \Box
\]

Note that in the case when \( \delta = f^* \), then \( S_\delta = \text{Opt} \) whereby Theorem 2.2.2 specializes to the standard result for the Accelerated Gradient Method. We will
utilize the more general result in Theorem 2.2.2 in the context of smoothing of a non-smooth function, in Sections 2.3.3 and 2.4 herein.

2.3 Computational Guarantees when $f(\cdot)$ is Non-Smooth

Let $\varepsilon' > 0$ be given. We aspire to compute an $\varepsilon'$-relative solution of (4.1), which recall from (2.2) is a point $\hat{x} \in Q$ satisfying: $\frac{f(\hat{x}) - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon'$. In this section we present three new computational guarantees for first-order methods applied to computing a $\varepsilon'$-relative solution of problem (4.1) that are based on the strict lower bound $f_{\text{slb}}$ and growth constant $G$. The first guarantee is for a new algorithm based on Subgradient Descent that runs two different step-sizes simultaneously with occasional re-starts. The second guarantee is for the standard Subgradient Method using a standard step-size rule in the case when the optimal value $f^*$ is known. The third guarantee is for the case when the function $f(\cdot)$ can be smoothed and then solved using an algorithm based on the Accelerated Gradient Method.

2.3.1 Subgradient Descent using Two Step-Size Rules Running Simultaneously

We consider solving (4.1) using a version of subgradient descent that simultaneously runs two versions of the Subgradient Descent Method – each with a different step-size rule – with occasional simultaneous re-starts of both versions. The formal description of our method is given in Algorithm 3. In the algorithm, the notation $a(x_{ij+1}, f^i_{b,j+1}, x_{ij}^{i,j+1}) \leftarrow \text{SDM}(f(\cdot), x_{ij}, \alpha_{ij}, g_{ij})$ denotes assigning to $x_{i,j+1}$ the next value of the Subgradient Descent Method applied to the optimization problem (4.1) with objective function $f(\cdot)$ with current point $x_{i,j} \in Q$ using the step-size $\alpha_{ij}$ and the subgradient $g_{ij}$, along with updates of the best objective function value obtained thus far $f^i_{b,j+1}$ with the corresponding best iterate computed $x_{b}^{i,j+1}$.

We now walk through the structure of Algorithm 3. The algorithm requires as
Algorithm 3 Non-Smooth Method with Two Step-Size Rules Running Simultaneously

**Initialize.** Initialize with \( x^0 \in Q \) and \( \varepsilon' > 0 \).
Define constants \( \bar{\varepsilon}' := 0.9 , \varepsilon := \frac{\varepsilon'}{1+\bar{\varepsilon}} , \bar{\varepsilon} := \frac{\varepsilon'}{1+\bar{\varepsilon}} , B := 1/\sqrt{\bar{\varepsilon}} , F := \sqrt{\bar{\varepsilon}} \).
Set \( x_{1,0} \leftarrow x^0 , \bar{x}_{1,0} \leftarrow x^0 , i \leftarrow 1 \).

At outer iteration \( i \):

1. **Initialize inner iterations.** \( f^{i,0}_b \leftarrow f(x_{i,0}) \), \( f^{i,0}_{\bar{b}} \leftarrow f(\bar{x}_{i,0}) \)
\( x^{i,0}_b \leftarrow x_{i,0} \), \( x^{i,0}_{\bar{b}} \leftarrow \bar{x}_{i,0} \)
\( K_i \leftarrow +\infty , j \leftarrow 0 \).

2. **Test/update current iterates.** At inner iteration \( j \):
   If \( \frac{f(x_{i,j}) - f_{\text{sib}}}{f(x_{i,0}) - f_{\text{sib}}} \geq B \) and \( \frac{f(\bar{x}_{i,j}) - f_{\text{sib}}}{f(\bar{x}_{i,0}) - f_{\text{sib}}} \geq B \), then
   **Compute subgradients.** Compute \( g_{ij} \in \partial f(x_{i,j}) , \bar{g}_{ij} \in \partial f(\bar{x}_{i,j}) \)
   **Set step-sizes.** \( \alpha_{ij} \leftarrow \frac{\varepsilon(f(x_{i,j}) - f_{\text{sib}})}{F\|g_{ij}\|^2} , \bar{\alpha}_{ij} \leftarrow \frac{\varepsilon(f(\bar{x}_{i,j}) - f_{\text{sib}})}{F\|\bar{g}_{ij}\|^2} \)
   **Update:**
   \( (x_{i,j+1}, f^{i,j+1}_b, x^{i,j+1}_b) \leftarrow \text{SDM}(f(\cdot), x_{i,j}, \alpha_{ij}, g_{ij}) \)
   \( (\bar{x}_{i,j+1}, f^{i,j+1}_{\bar{b}}, \bar{x}^{i,j+1}_{\bar{b}}) \leftarrow \text{SDM}(f(\cdot), \bar{x}_{i,j}, \bar{\alpha}_{ij}, \bar{g}_{ij}) \)

   Else if \( \frac{f(x_{i,j}) - f_{\text{sib}}}{f(x_{i,0}) - f_{\text{sib}}} < B \), then:
   \( K_i \leftarrow j , x_{i+1,0} \leftarrow x_{i,j} , \bar{x}_{i+1,0} \leftarrow \bar{x}_{i,j} , i \leftarrow i + 1 \), and
   Goto Step 1.

   Else
   \( \frac{f(\bar{x}_{i,j}) - f_{\text{sib}}}{f(\bar{x}_{i,0}) - f_{\text{sib}}} < B \), and:
   \( K_i \leftarrow j , x_{i+1,0} \leftarrow \bar{x}_{i,j} , \bar{x}_{i+1,0} \leftarrow \bar{x}_{i,j} , i \leftarrow i + 1 \), and
   Goto Step 1.

input the starting point \( x^0 \) and the desired relative accuracy value \( \varepsilon' \) used to define an \( \varepsilon' \)-relative solution, see (2.2). The algorithm then defines an absolute constant \( \bar{\varepsilon}' := 0.9 \). The two values \( \varepsilon' \) and \( \bar{\varepsilon}' \) are then used as aspirational goals for simultaneously running the standard Subgradient Descent Method in search of either an \( \varepsilon' \)-relative solution of (4.1) or an \( \bar{\varepsilon}' \)-relative solution of (4.1). For notational ease, both \( \varepsilon' \) and \( \bar{\varepsilon}' \) are converted to a slightly different form by defining \( \varepsilon \) and \( \bar{\varepsilon} \). At the start of the \( i^{\text{th}} \) outer iteration, Algorithm 3 runs the Subgradient Descent Method simultaneously using two different step-size rules (but starting at the same point \( x_{i,0} = \bar{x}_{i,0} \), and
so generates inner iterations \( \{x_{i,j}\} \) and \( \{\bar{x}_{i,j}\} \) for \( j = 0, 1, \ldots \) based on computed subgradients \( \{g_{ij}\} \) and \( \{\bar{g}_{ij}\} \) and step-sizes \( \{\alpha_{ij}\} \) and \( \{\bar{\alpha}_{ij}\} \), respectively. The only structural difference between the two instantiations of Subgradient Descent is that the steps-sizes \( \{\alpha_{ij}\} \) use \( \varepsilon \) in their definition whereas \( \{\bar{\alpha}_{ij}\} \) use \( \bar{\varepsilon} \) in their definition.

The number of inner iterations \( j \) that are run in the \( i^{th} \) outer iteration is initially set to be \( K_i \leftarrow +\infty \). If either \( f(x_{i,j}) \) or \( f(\bar{x}_{i,j}) \) makes sufficient progress relative to the starting value \( f(x_{i,0}) (= f(\bar{x}_{i,0})) \) as determined in the ratio test at the start of Step (2.), then the outer iteration \( i \) is concluded and \( K_i \), which counts the number of inner iterations therein, is updated to \( K_i \leftarrow j \). Finally, the next outer iteration starting values \( x_{i+1,0} = \bar{x}_{i+1,0} \) are re-set to either \( x_{i,j} \) or \( \bar{x}_{i,j} \), depending on which of \( x_{i,j} \) or \( \bar{x}_{i,j} \) satisfies the ratio test.

Many of the ideas used in the construction of Algorithm 3 were motivated from similar notions developed in Algorithm 2 of [91] as well as the algorithm “MainAlgo” in [90] (which uses the construct of running two algorithms simultaneously with different parameters).

Regarding counting of iterates \( x_{i,j}, \bar{x}_{i,j} \) that are computed by Algorithm 3, we will say that the algorithm has computed an iterate whenever it computes a subgradient and then calls SDM(\( \cdot, \cdot, \cdot \)). There are therefore two iterates computed at each inner iteration. We have:

\textbf{Theorem 2.3.1. (Complexity Bound for Algorithm 3)} \textit{Within a total number of iterates computed that does not exceed}

\[
18M^2G^2 \left( 2.7 \ln \left( 1 + \frac{f(x^0) - f^*}{f^* - f_{\text{slb}}} \right) + \left( \frac{1 + \varepsilon'}{\varepsilon'} \right)^2 \right),
\]

Algorithm 3 will compute an iterate \( x_{i,j} \) for which

\[
\frac{f(x_{i,j}) - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon' . \quad \Box
\]

Since \( f(x^0) \leq f^* + M\text{Dist}(x^0, \text{Opt}) \), the computational guarantee in Theorem 2.3.1
can itself be bounded by:

\[
18M^2G^2 \left( 2.7 \ln \left( 1 + \frac{MDist(x^0, Opt)}{f^* - f_{slb}} \right) + \left( \frac{1 + \varepsilon'}{\varepsilon'} \right)^2 \right),
\]  

(2.9)

which is qualitatively different from the guarantee of the standard Subgradient Descent Method (Algorithm 1) in (2.7) in two interesting ways. First, the dependence in (2.7) on Dist(x^0, Opt) is quadratic, whereas in (2.9) it is logarithmic. Second, although both guarantees are linear in the inverse square of the desired relative accuracy \( \varepsilon' \) (from (2.2) an \( \varepsilon' \)-relative solution corresponds to an absolute \( \varepsilon' \cdot (f^* - f_{slb}) \) solution of (4.1)), however \( x^0 \) affects this factor multiplicatively through \( Dist(x^0, Opt)^2 \) in (2.7), whereas the factor is independent of \( x^0 \) in (2.9).

Let us also quantitatively compare the computational guarantee of Theorem 2.3.1 with the standard guarantee for Subgradient Descent given by (2.7). The standard computational guarantee (2.7) can be written as:

\[
\frac{M^2 Dist(x^0, Opt)^2}{\varepsilon'^2 (f^* - f_{slb})^2}.
\]

Let us presume that \( \varepsilon' \) is small, whereby \( \frac{1 + \varepsilon'}{\varepsilon'} \approx \frac{1}{\varepsilon'} \). Then the ratio of the new guarantee (2.9) from Theorem 2.3.1 to the standard guarantee (2.7) is at most

\[
\frac{\text{Guarantee of Theorem 2.3.1}}{\text{Standard Guarantee (2.7)}} \leq 18(f^* - f_{slb})^2G^2 \left( \frac{2.7(\varepsilon')^2 \ln \left( 1 + \frac{MDist(x^0, Opt)}{f^* - f_{slb}} \right) + 1}{\text{Dist}(x^0, Opt)^2} \right).
\]  

(2.10)

Notice from (2.10) that for any instance of (4.1), when Dist(x^0, Opt) is sufficiently large the right-hand side of (2.10) can be made arbitrarily small, thereby showing that in these cases the computational guarantee in Theorem 2.3.1 can be made arbitrarily better than the standard guarantee (2.7) for Subgradient Descent.

We will prove Theorem 2.3.1 by first establishing eight propositions. The reader familiar with [91] will notice certain resemblances between aspects of the proof constructs below and the proof of Theorem 3.8 of [91], see also [92]. Throughout, for
notational convenience, we will work with three constants $B$, $F$, and $\varepsilon'$ that must be chosen to satisfy the conditions:

$$B \in (0,1) \; , \; F > \frac{1}{2B} \; , \; \text{and} \; \varepsilon' > 0 \; ,$$

and whose specific values in Algorithm 3 are set to $B = 1/\sqrt{e}$, $F = \sqrt{e}$, and $\varepsilon' = 0.9$, where $e$ is the base of the natural logarithm.

Let $\delta' > 0$ play the role of either $\varepsilon'$ or $\varepsilon''$, and also define $\delta := \frac{\delta'}{1 + \delta'}$ (analogous to the definitions of $\varepsilon$ and $\varepsilon$).

The first two propositions below apply to the generic setting of the Subgradient Descent Method.

**Proposition 2.3.1.** Let $\delta \in (0,1)$ be given, and suppose we run the Subgradient Descent Method (Algorithm 1) with starting iterate $\hat{x}^0$, using step-sizes:

$$\alpha_j := \frac{\delta(f(\hat{x}^0) - f_{slb})}{F \|g_j\|^2}$$

for all iterations $j$. Then for all $j \geq 0$ it holds that

$$f^j - f_{slb} \leq f^* - f_{slb} + \left[ \frac{G^2M^2F}{2\delta(j + 1)} + \frac{\delta}{2F} \right] (f(\hat{x}^0) - f_{slb}) .$$

**Proof:** Define $\alpha := \frac{\delta(f(\hat{x}^0) - f_{slb})}{F}$. Then $\alpha_j = \frac{\alpha}{\|g_j\|^2} \geq \frac{\alpha}{M^2}$. It follows from part (i) of Theorem 2.2.1 that

$$f^j - f_{slb} \leq f^* - f_{slb} + \frac{\text{Dist}(\hat{x}^0, \text{Opt})^2}{2 \sum_{i=0}^{j} \alpha_i} + \frac{\sum_{i=0}^{j} \|g_i\|^2 \alpha_i^2}{2 \sum_{i=0}^{j} \alpha_i}$$

$$\leq f^* - f_{slb} + \frac{M^2 \text{Dist}(\hat{x}^0, \text{Opt})^2}{2\alpha(j + 1)} + \frac{\alpha}{2}$$

$$\leq f^* - f_{slb} + \frac{M^2 \varepsilon^2 (f(\hat{x}^0) - f_{slb})^2}{2(j + 1)\delta (f(\hat{x}^0) - f_{slb})} + \frac{\delta (f(\hat{x}^0) - f_{slb})}{2F} ,$$
where the second inequality uses the definition of $\alpha_i$ and the inequality $\|g_i\| \leq M$, and the third inequality uses the definitions of $\alpha$ and $G$. Simplifying the last expression completes the proof. \qed

**Proposition 2.3.2.** Under the identical set-up as Proposition 2.3.1, let $\delta' := \delta/(1 - \delta)$, and define:

$$W := \frac{FM^2C^2}{2\delta^2 [B - \frac{1}{2F}]}.$$ 

Then either $\frac{f^W - f^*}{f^* - f_{\text{slb}}} \leq \delta'$, or $f^W_b - f_{\text{slb}} \leq B(f(\hat{x}^0) - f_{\text{slb}})$, or both.

**Proof:** Suppose that $\frac{f^W - f^*}{f^* - f_{\text{slb}}} > \delta'$. This rearranges to: $\delta' < \frac{f^W - f_{\text{slb}}}{f^* - f_{\text{slb}}} - 1$, whereby

$$\frac{f^* - f_{\text{slb}}}{f^W_b - f_{\text{slb}}} < \frac{1}{1 + \delta'} = 1 - \delta'. \quad (2.11)$$

Invoking Proposition 2.3.1 we have:

$$f_b^W - f_{\text{slb}} \leq f^* - f_{\text{slb}} + \left[ \frac{G^2M^2F}{2\delta(W + 1)} + \frac{\delta}{2F} \right] (f(\hat{x}^0) - f_{\text{slb}})$$

$$< f^* - f_{\text{slb}} + \left[ \delta \left( B - \frac{1}{2F} \right) + \frac{\delta}{2F} \right] (f(\hat{x}^0) - f_{\text{slb}})$$

$$= f^* - f_{\text{slb}} + \delta B(f(\hat{x}^0) - f_{\text{slb}})$$

$$< (1 - \delta)(f_b^W - f_{\text{slb}}) + \delta B(f(\hat{x}^0) - f_{\text{slb}}),$$

where the second inequality follows since $W + 1 > \frac{FM^2C^2}{2\delta^2 [B - 1/(2F)]}$, and the last inequality uses (2.11). Rearranging the final inequality and dividing by $\delta$ then yields $f_b^W - f_{\text{slb}} \leq B(f(\hat{x}^0) - f_{\text{slb}})$, which completes the proof. \qed

In the next two propositions we apply Proposition 2.3.2 directly to the setting of Algorithm 3.
Proposition 2.3.3. Consider outer iteration \( i \) of Algorithm 3. Define:

\[
U := \left\lfloor \frac{FM^2G^2}{2\varepsilon^2 \left[ B - \frac{1}{2F} \right]} \right\rfloor .
\]

If \( K_i > U \), then \( \frac{f^{ij}_b - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon' \) for all \( j = U, \ldots, K_i \).

**Proof:** Let us apply Proposition 2.3.2 with \( \delta' := \varepsilon' \), \( W := U \), and \( \alpha^0 := x_{i,0} \). If \( K_i > U \), then by definition of \( K_i \) it holds that \( f^{ij}_b - f_{\text{slb}} > B(f^{i,0}_b - f_{\text{slb}}) \). Thus, from Proposition 2.3.2 it holds that \( \frac{f^{ij}_b - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon' \). Therefore \( \frac{f^{ij}_b - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon' \) for all \( j = U, \ldots, K_i \), since \( f^{ij}_b \) is by definition monotonically nonincreasing in \( j \). \( \square \)

Proposition 2.3.4. Consider outer iteration \( i \) of Algorithm 3. Define:

\[
V := \left\lfloor \frac{FM^2G^2}{2\varepsilon^2 \left[ B - \frac{1}{2F} \right]} \right\rfloor .
\]

If \( K_i > V \), then \( \frac{f(x_{i,0}) - f_{\text{slb}}}{f^* - f_{\text{slb}}} \leq \frac{1 + \varepsilon'}{B} \).

**Proof:** Let us similarly apply Proposition 2.3.2 with \( \delta' := \varepsilon' \), \( W := V \), and \( \alpha^0 := x_{i,0} = \tilde{x}_{i,0} \). Let us suppose \( K_i > V \). First, notice that for \( 0 \leq j < K_i \), it holds that

\[
f(\tilde{x}_{i,j}) - f_{\text{slb}} > B(f(\tilde{x}_{i,0}) - f_{\text{slb}}) = B(f(x_{i,0}) - f_{\text{slb}}) .
\]

Therefore the left-hand term can be replaced by \( f^{ij}_b - f_{\text{slb}} \), and setting \( j = V \) we obtain \( f^{i,V}_b - f_{\text{slb}} \geq B(f(x_{i,0}) - f_{\text{slb}}) \). Therefore from Proposition 2.3.2 it holds that \( \frac{f^{i,V}_b - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon' \). Combining these inequalities we obtain:

\[
B(f(x_{i,0}) - f_{\text{slb}}) \leq f^{i,V}_b - f_{\text{slb}} = f^{i,V}_b - f^* + f^* - f_{\text{slb}} \leq \varepsilon'(f^* - f_{\text{slb}}) + f^* - f_{\text{slb}} ,
\]

and rearranging yields the result. \( \square \)

In the next proposition we use the standard notation \( a^+ \) for the nonnegative part of a scalar \( a \).
Proposition 2.3.5. Let $m$ denote the number of outer iterations $i$ of Algorithm 3 for which
\[ \frac{f(x_i,0) - f_{sib}}{f^* - f_{sib}} > \frac{1 + \varepsilon'}{B}. \]

Then
\[ m \leq \left\lceil \frac{\ln \left( 1 + \frac{f(x_i,0) - f^*}{f^* - f_{sib}} \right) - \ln \left( \frac{1 + \varepsilon'}{B} \right)}{\ln(1/B)} \right\rceil. \]

Proof: If $m = 0$ then the result holds trivially, so let us suppose that $m \geq 1$. It then follows using induction on $f(x_{i+1,0}) - f_{sib} \leq B(f(x_i,0) - f_{sib})$ that
\[ \frac{1 + \varepsilon'}{B} < \frac{f(x_m,0) - f_{sib}}{f^* - f_{sib}} \leq \frac{B^{m-1}(f(x_1,0) - f_{sib})}{f^* - f_{sib}}, \]
and taking logarithms yields
\[ m - 1 < \frac{\ln \left( \frac{f(x_1,0) - f_{sib}}{f^* - f_{sib}} \right) - \ln \left( \frac{1 + \varepsilon'}{B} \right)}{\ln(1/B)} = \frac{\ln \left( 1 + \frac{f(x_0) - f^*}{f^* - f_{sib}} \right) - \ln \left( \frac{1 + \varepsilon'}{B} \right)}{\ln(1/B)}, \]
from which the result follows. \qed

In the following proposition, as well as others later on, we use the standard notational convention that $\sum_{i=1}^n \cdot := 0$ for $n \leq 0$.

Proposition 2.3.6. Let $V$ and $m$ be as defined in Propositions 2.3.4 and 2.3.5. Then $x_{m+1,0}$ exists, and let $T_m$ denote the total number of iterates computed prior to and including $x_{m+1,0}$. It holds that:
\[ \frac{f(x_{m+1,0}) - f_{sib}}{f^* - f_{sib}} \leq \frac{1 + \varepsilon'}{B}, \]
and furthermore $T_m \leq 2mV$.

Proof: If $m = 0$ then the results holds trivially from the definition of $m$. 38
Next suppose that \( m \geq 1 \), and consider any outer iteration \( i \leq m \). Then since

\[
\frac{f(x_{i,0}) - f_{slb}}{f^* - f_{slb}} > \frac{1 + \varepsilon'}{B},
\]

it follows from Proposition 2.3.4 that \( K_i \leq V \). This also implies that \( x_{m+1,0} \) exists and therefore must satisfy

\[
\frac{f(x_{m+1,0}) - f_{slb}}{f^* - f_{slb}} \leq \frac{1 + \varepsilon'}{B}.
\]

Finally, since \( T_m = 2 \sum_{i=1}^{m} K_i \), it therefore follows that \( T_m \leq 2mV \).

**Proposition 2.3.7.** Let \( p \) denote the number of outer iterations \( i \) for which \( K_i \) is finite. Then

\[
p \leq m + \left\lfloor \frac{\ln \left( \frac{1 + \varepsilon'}{B} \right)}{\ln(1/B)} \right\rfloor,
\]

where \( m \) is as defined in Proposition 2.3.5.

**Proof:** It follows from Proposition 2.3.6 that \( p \geq m \). Therefore \( f^* - f_{slb} \leq f(x_{p,K_p}) - f_{slb} = f(x_{p+1,0}) - f_{slb} \leq B^{p-m}(f(x_{m+1,0}) - f_{slb}) \leq B^{p-m} \left( \frac{1 + \varepsilon'}{B} \right) (f^* - f_{slb}) \),

where we have used the properties of \( x_{m+1,0} \) in Proposition 2.3.6. Taking logarithms yields

\[
p - m \leq \frac{\ln \left( \frac{1 + \varepsilon'}{B} \right)}{\ln(1/B)},
\]

from which the result follows.

**Proposition 2.3.8.** Let \( U, m, \) and \( p \) be as defined in Propositions 2.3.3, 2.3.5, and 2.3.7. Within a total number of computed iterates after \( x_{m+1,0} \) that does not exceed \( 2(p - m + 1)U \), Algorithm 3 will compute an iterate \( x_{i,j} \) for which

\[
\frac{f(x_{i,j}) - f^*}{f^* - f_{slb}} \leq \varepsilon' .
\]

**Proof:** Let \( \hat{i} \) denote the index of the first outer iteration \( i \in \{m + 1, \ldots, p + 1\} \) for which \( K_i > U \). Notice that since \( K_{p+1} = +\infty \) it must hold that \( \hat{i} \leq p + 1 \). It
follows from Proposition 2.3.3 that \( \frac{f_{x_i}^j - f^*}{f^* - f_{x_i}^0} \leq \epsilon' \) and hence for some \( j \leq U \) it holds that \( \frac{f(x_{i,j}) - f^*}{f^* - f_{x_i}^0} \leq \epsilon' \). Let us now count the number of iterates computed after \( x_{m+1,0} \) and prior to and including \( x_{i,j} \). This number is bounded above by:

\[
2 \left( \sum_{i=m+1}^{i=m+1} K_i + U \right) \leq 2 \left( (\hat{i} - m - 1)U + U \right) = 2(\hat{i} - m)U \leq 2(p - m + 1)U ,
\]

where the first inequality follows since \( K_i \leq U \) for \( i < \hat{i} \), and the last inequality uses \( \hat{i} \leq p + 1 \). \( \square \)

We now use these propositions to prove Theorem 2.3.1.

**Proof of Theorem 2.3.1:** Utilizing the definitions of \( U, V, m, p, \) and \( x_{i,j} \) in Propositions 2.3.3, 2.3.4, 2.3.5, 2.3.7, and 2.3.8, it follows from Propositions 2.3.6 and 2.3.8 that the total number of iterates computed prior to and including \( x_{i,j} \) is at most \( 2[mV + (p - m + 1)U] \). Substituting the values of \( U \) and \( V \) and using the bounds on \( m \) and \( p \) in Propositions 2.3.5 and 2.3.7 yields:

\[
2 \left[ \frac{\ln \left( 1 + \frac{f(x_i^0) - f^*}{f^* - f_{x_i}^0} \right)}{\ln(1/B)} \right] + \frac{FM^2G^2}{2e^2[B-1/F]} + 2 \left[ \frac{\ln \left( 1 + \frac{\epsilon'}{B} \right)}{\ln(1/B)} \right] \leq \frac{FM^2G^2}{2e^2[B-1/F]},
\]

\[
\leq M^2G^2 \left( 48.5 \ln \left( 1 + \frac{f(x_i^0) - f^*}{f^* - f_{x_i}^0} \right) + 18 \left( \frac{1+\epsilon'}{\epsilon'} \right)^2 \right) ,
\]

where the second inequality follows from substituting in the values \( B = 1/\sqrt{e} \), \( F = \sqrt{e} \), and \( \epsilon' = 0.9 \), and rounding terms upward. This last expression then is rounded upward to yield the desired iteration bound. \( \square \)
### 2.3.2 Subgradient Descent when $f^*$ is known

In the special case when $f^*$ is known, we can obtain a computational guarantee that is of the same order as that of Theorem 2.3.1 by directly using the standard Subgradient Descent Method (Algorithm 1) with the (standard) step-size rule $\alpha_i := (f(x^i) - f^*)/\|g_i\|^2$. This is shown in the following theorem.

**Theorem 2.3.2. (Complexity Bound for standard Subgradient Descent when $f^*$ is known)** Let the step-sizes for the Subgradient Descent Method (Algorithm 1) applied to solve problem (4.1) be chosen as:

$$\alpha_i := \frac{f(x^i) - f^*}{\|g_i\|^2},$$

and suppose that $N \geq 0$ and satisfies

$$N \geq 2M^2G^2 \left(1 + 2.9 \ln \left(\frac{f(x^0) - f^*}{f^* - f_{slb}}\right) + 2.9 \ln \left(\frac{1}{\varepsilon'}\right) + 6.8 \left(\frac{1}{\varepsilon'}\right) + 2 \left(\frac{1}{\varepsilon'}\right)^2\right).$$

Then:

$$\frac{f(x^N_b) - f^*}{f^* - f_{slb}} \leq \varepsilon'. \quad \Box \quad (2.12)$$

The computational guarantee above is an almost-exact generalization of Theorem 3.7 of Renegar [91], which therein pertains to a specific transformed conic optimization problem. The proof of this theorem follows the logic for the proof of Theorem 3.7 of [91] in many respects as well.

Notice that up to an absolute constant, the computational guarantee of Theorem 2.3.2 is essentially the same as that of Theorem 2.3.1 in the worst case.

**Proof of Theorem 2.3.2:** We will presume that $\frac{f(x^0) - f^*}{f^* - f_{slb}} > \varepsilon'$, since otherwise (2.12) is satisfied trivially for all $N \geq 0$. Let $B \in (0, 1)$ be a given fractional quantity. Define $K_0 := 0$, and for all $i$ such that $f_b^{K_i} - f^* > 0$ define $K_{i+1}$ inductively as the smallest iteration index of Subgradient Descent for which $f_b^{K_{i+1}} - f^* \leq B(f_b^{K_i} - f^*)$. Notice that so long as $f_b^{K_i} - f^* > 0$ it follows using part (ii) of Theorem 2.2.1 that

41
$K_{i+1}$ exists (i.e., is finite). Let $i'$ be the smallest sub-index $i$ for which $\frac{\delta^{K_{i'}} - f^*}{f^* - f_{slb}} \leq \varepsilon'$.

It follows from the initial presumption above that $i' \geq 1$, and it holds for any $i \geq 0$ satisfying $i < i'$ that $\varepsilon'(f^* - f_{slb}) < f(x^{K_i}) - f^* \leq B^i(f(x^{K_0}) - f^*) = B^i(f(x^0) - f^*)$, from which it follows that $i$, and hence also $i'$, is finite. Furthermore, it holds for any $i \geq 0$ satisfying $i < i'$ that:

$$\varepsilon'(f^* - f_{slb}) < f(x^{K_{i'-1}}) - f^* \leq B^{i'-1-i}(f(x^{K_i}) - f^*)$$

(2.13)

since $x^{K_i} = x_b^{K_i}$ by the definition of $K_i$. Using $i = 0$ in (2.13) and taking logarithms yields:

$$i' < 1 + \frac{\ln \left( \frac{1}{\varepsilon'} \right) + \ln \left( \frac{f(x^0) - f^*}{f^* - f_{slb}} \right)}{\ln(B^{-1})}.$$

(2.14)

If $K_{i+1}$ exists (i.e., is finite), then it follows from part (ii) of Theorem 2.2.1 that:

$$f(x_b^{K_{i+1}-1}) - f^* \leq \frac{MDist(x^{K_i}, \text{Opt})}{\sqrt{K_{i+1} - 1 - K_i + 1}}.$$

This last inequality can be rearranged to yield:

$$K_{i+1} - K_i \leq \frac{M^2 \text{Dist}(x^{K_i}, \text{Opt})^2}{(f(x_b^{K_{i+1}-1}) - f^*)^2} < \frac{B^{-2}M^2G^2 \left( f(x^{K_i}) - f_{slb} \right)^2}{(f(x^{K_i}) - f^*)^2} = B^{-2}M^2G^2 \left( 1 + \frac{f^* - f_{slb}}{f(x^{K_i}) - f^*} \right)^2$$

(2.15)

where the second inequality uses the definition of the growth constant $G$ as well as the fact that $f(x_b^{K_{i+1}-1}) - f^* > B(f(x^{K_i}) - f^*)$. Now putting all of this together we
obtain:

\[
K_{i'} = \sum_{i=0}^{i'-1} (K_{i+1} - K_i)
\]

\[
\leq B^{-2}M^2G^2 \sum_{i=0}^{i'-1} \left(1 + \frac{f^* - f_{slb}}{f(x^{K_i}) - f^*}\right)^2
\]

\[
\leq B^{-2}M^2G^2 \sum_{i=0}^{i'-1} \left(1 + \frac{1}{\varepsilon^i} B^{i'-1-i}\right)^2
\]

\[
= B^{-2}M^2G^2 \sum_{i=0}^{i'-1} \left(1 + \left(\frac{2}{\varepsilon^i}\right) B^i + \left(\frac{1}{\varepsilon^i}\right)^2 (B^2)^i\right)
\]

\[
\leq B^{-2}M^2G^2 \left(i' + \left(\frac{2}{\varepsilon^i}\right) \frac{1}{1-B} + \left(\frac{1}{\varepsilon^i}\right)^2 \frac{1}{1-B^2}\right)
\]

\[
\leq B^{-2}M^2G^2 \left(1 + \frac{\ln \left(\frac{1}{\varepsilon^i}\right) + \ln \left(\frac{f(x^0) - f^*}{f^* - f_{slb}}\right)}{\ln(B^{-1})} + \left(\frac{2}{\varepsilon^i}\right) \frac{1}{1-B} + \left(\frac{1}{\varepsilon^i}\right)^2 \frac{1}{1-B^2}\right),
\]

where the first inequality is from (2.15), the second inequality uses (2.13), the third inequality replaces the two finite geometric series with corresponding infinite series, and the fourth inequality uses (2.14). Finally, using the value of \( B = 1/\sqrt{2} \) and substituting into the above yields the result. \( \square \)

We remark that one obtains the precise constants of Theorem 3.7 of [91] by using \( B = 1/2 \). Choosing \( B \) to optimize the absolute constant of the \((1/\varepsilon^i)^2\) term yields \( B = 1/\sqrt{2} \) and the absolute constants as presented in the statement of the theorem. Choosing \( B \) to optimize the absolute constant of the \( \ln \left(\frac{f(x^0) - f^*}{f^* - f_{slb}}\right) \) term would yield \( B = 1/\sqrt{e} \) with the coefficient of 2 in the \( \ln(\cdot) \) terms.

To conclude this subsection, consider the case when \( f^* \) is known and \( G \) can be upper-bounded by a constant for any \( f_{slb} \) (as is the case when \( f(\cdot) \) is piecewise-linear or, more generally, when \( f(\cdot) \) has weak sharp minima (A.1)). Then given an absolute
tolerance $\varepsilon$, we can set $\varepsilon' = 1$ and $f_{slb} = f^* - \varepsilon$, whereby Theorem 2.3.2 implies linear convergence of Algorithm 1 in term of the absolute tolerance $\varepsilon$. However, this is not as favorable a result as that in Yang and Lin [110], which obtains linear convergence without requiring that $f^*$ is known. Also, this result can be considered a slight variation of Gilpin, Peña and Soheli [44], which assumes $f(\cdot)$ is piecewise-linear but does not require that $f^*$ is known.

2.3.3 Non-Smooth Optimization using a New Smooth Approximations Method

As first proposed by Nesterov [78], there are many practical settings wherein one can approximate the non-smooth convex function $f(\cdot)$ by a smooth convex function $f_\mu(\cdot)$, where the sense of the approximation depends on the parameter $\mu$. If the smooth approximation $f_\mu(\cdot)$ is computationally easy to work with, one can then use the Accelerated Gradient Method (Algorithm 2) to approximately optimize $f_\mu(\cdot)$ (thereby also approximately optimizing $f(\cdot)$) on the feasible set $Q$. There are a variety of techniques that can be used to construct a parametric family of smooth functions $f_\mu(\cdot)$ depending on the known structure of $f(\cdot)$ and $Q$, see [78] as well as [79] and Beck and Teboulle [10] among others. For our purposes herein, we will suppose that there is a smoothing technique with the following two properties:

(i) there is a known constant $\bar{D} > 0$ such that for any given $\mu > 0$ we can construct a smooth convex function $f_\mu(\cdot) : Q \to \mathbb{R}$ which is not far from $f(\cdot)$, namely:

$$f(x) - \bar{D}\mu \leq f_\mu(x) \leq f(x) \quad \text{for all } x \in Q \text{, and} \quad (2.16)$$

(ii) $f_\mu(\cdot)$ has Lipschitz continuous gradient on $Q$ with Lipschitz constant $L_\mu$ satisfying

$$L_\mu \leq \bar{A}/\mu \quad (2.17)$$

for some known positive constant $\bar{A}$. 

44
These properties can be used to design an implementation of the Accelerated Gradient Method (Algorithm 2) applied to $f_\mu(\cdot)$, that can be used to compute an absolute $\varepsilon$-optimal solution of the original optimization problem (4.1). The scheme developed in [78] in conjunction with the Accelerated Gradient Method (Algorithm 2) yields an iteration complexity bound of

$$N := \left\lceil \frac{\sqrt{8AD\text{Dist}(x^0, \text{Opt})}}{\varepsilon} \right\rceil - 1$$

(2.18)

to obtain an (absolute) $\varepsilon$-optimal solution of (4.1) for a suitably designed version of the basic method.

Herein we develop a variant of the basic smoothing method to solve the optimization problem (4.1) that yields a new computational guarantee that can improve on (2.18) in many cases. Algorithm 4 presents parametric smoothing and restarting method for computing an $\varepsilon'$-relative solution of the optimization problem (4.1) for the non-smooth objective function $f(\cdot)$ based on successive smooth approximations and re-starting of the Accelerated Gradient Method (Algorithm 2). In the description of Algorithm 4 the general notation “$x_{i,j} \leftarrow \text{AGM}(f_\mu(\cdot), x_{i,0}, j)$” denotes assigning to $x_{i,j}$ the $j^{th}$ iterate of the Accelerated Gradient Method applied to the optimization problem (4.1) with objective function $f_\mu(\cdot)$ using the initial point $x_{i,0} \in Q$.

At the $i^{th}$ outer iteration of Algorithm 4, the algorithm sets two different smoothing parameters in Step (1.), namely $\mu^1_i$ and $\mu^2_i$, where $\mu^2_i$ differs from $\mu^1_i$ by the relative accuracy input value $\varepsilon'$. The algorithm then runs the Accelerated Gradient Method with starting point $x_{i,0}$ simultaneously on the two smoothed functions $f_{\mu^1_i}(\cdot)$ and $f_{\mu^2_i}(\cdot)$, using the double indexing notation of $x_{i,j}$ and $y_{i,j}$ to denote iteration $j$ of the Accelerated Gradient Method initialized at the point $x_{i,0}$ for optimizing $f_{\mu^1_i}(\cdot)$ and $f_{\mu^2_i}(\cdot)$ on $Q$, respectively. Notice that the smoothing parameters $\mu^1_i$ and $\mu^2_i$ decrease over the course of the outer iterations, as it makes more sense to set these values higher at first and then decrease them as the solution is approached. The outer iteration $i$ runs until the ratio test in Step (3a.) fails, at which point the current point...
Algorithm 4 Parametric Smoothing/Restarting Method using $f_{\mu}(\cdot)$

**Initialize.** Initialize with $x^0 \in Q$ and $\varepsilon > 0$.
Define $B := \frac{1}{2}$, $t := \frac{1}{8}$.
Set $x_{1,0} \leftarrow x^0$, $i \leftarrow 1$.

At outer iteration $i$:

1. **Set smoothing parameters.** $\mu_1^i \leftarrow \frac{t \cdot (f(x_{i,0}) - f_{\text{slb}})}{D}$, $\mu_2^i \leftarrow \frac{t\varepsilon' \cdot (f(x_{i,0}) - f_{\text{slb}})}{D}$

2. **Initialize inner iteration.** $K_i \leftarrow +\infty$, $j \leftarrow 0$

3. **Run inner iterations.** At inner iteration $j$:
   
   (3a.) If $\frac{f(x_{i,j}) - f_{\text{slb}}}{f(x_{i,0}) - f_{\text{slb}}} \geq B$, then
   
   $x_{i,j+1} \leftarrow \text{AGM}(f_{\mu_1^i}(\cdot), x_{i,0}, j + 1)$,
   
   $y_{i,j+1} \leftarrow \text{AGM}(f_{\mu_2^i}(\cdot), x_{i,0}, j + 1)$,
   
   $j \leftarrow j + 1$, and Goto (3a.)

   Else $K_i \leftarrow j$, $x_{i+1,0} \leftarrow x_{i,j}$, $i \leftarrow i + 1$, and Goto Step 1.

$x_{i,j}$ becomes the starting point of the next outer iteration, namely $x_{i+1,0} \leftarrow x_{i,j}$. The counter $K_i$ records the number of inner iterations $j$ of outer iteration $i$. Regarding counting of iterates computed in Algorithm 4, we will say that the algorithm has computed an iterate whenever it calls $\text{AGM}(\cdot, \cdot, \cdot)$. There are therefore two computed iterates at each inner iteration.

Restarting for accelerated gradient methods for strongly convex functions has been studied in [82] and [98]. To the best of our knowledge, restarting of accelerated methods in the absence of strong convexity was first used in Renegar [89], and Algorithm 4 exploits this and other ideas from [89] and [90] as well. We have the following computational guarantee associated with Algorithm 4.

**Theorem 2.3.3.** (Complexity Bound for Parametric Smoothing/Restarting Method (Algorithm 4) for Non-smooth Optimization) Suppose that $f_{\mu}(\cdot)$ satisfies the smoothing conditions (2.16) and (2.17). Within a total number of computed
Algorithm 4 will compute an iterate $y_{i,j}$ for which

$$\frac{f(y_{i,j}) - f^*}{f^* - f_{slb}} \leq \varepsilon'.$$

Similar to Theorem 2.3.1, the dependence in Theorem 2.3.3 on the quality of the initial iterate is logarithmic in the initial optimality gap $f(x^0) - f^*$. Also, the factor involving $1/\varepsilon'$ in Theorem 2.3.3 is independent of the quality of the initial iterate, unlike that of the standard bound for the smoothing method given in (2.18). We will prove Theorem 2.3.3 by first establishing several propositions. Throughout, for notational convenience, we will work with two constants $B$ and $t$ that must be chosen to satisfy

$$B > 0, \ t > 0, \ B - B^2 \geq 2t, \text{ and } B \geq 4t,$$

and whose specific values are set to $B = 1/2$ and $t = 1/8$ in Algorithm 4.

The following proposition applies to the generic setting of the Accelerated Gradient Method applied to the smoothed function $f_\mu(\cdot)$. Recall that $L_\mu$ denotes the Lipschitz constant of the gradient of $f_\mu(\cdot)$ on $Q$.

**Proposition 2.3.9.** Given the smoothing parameter $\mu > 0$ and a given constant $\beta > 0$, define $Y := \lceil G\sqrt{2\beta} - 1 \rceil$. Let $x_k \leftarrow \text{AGM}(f_\mu(\cdot), \hat{x}^0, k)$ denote the $k$th iterate of the Accelerated Gradient Method applied to the function $f_\mu(\cdot)$ with starting point $\hat{x}^0$. For $k \geq Y$ it holds that:

$$f(x_k) - f^* \leq \frac{L_\mu}{\beta} (f(\hat{x}^0) - f_{slb})^2 + \mu \bar{D}.$$  \hspace{1cm} (2.19)

**Proof:** Note that for any $x \in \text{Opt}$ it holds that $f_\mu(x) \leq f^*$, whereby $\text{Opt} \subset S := \{x \in Q : f_\mu(x) \leq f^*\}$. It then follows from Theorem 2.2.2 applied to the function
Using $\delta = f^*$ that for any $k \geq Y$ we have:

\[
f(x_k) - f^* \leq \frac{2L_n}{(Y+1)^2} \text{Dist}(\hat{x}^0, S)^2
\]

\[
\leq \frac{2L_n}{(Y+1)^2} \text{Dist}(\hat{x}^0, \text{Opt})^2 \leq \frac{2L_n}{(Y+1)^2} G^2 (f(\hat{x}^0) - f_{\text{slb}})^2 \leq \frac{L_n}{\beta} (f(\hat{x}^0) - f_{\text{slb}})^2,
\]

where the second inequality uses the fact that $\text{Opt} \subset S$, the third inequality uses the definition of $G$, and the last inequality uses the value of $Y$.

Note from (2.16) that $f(x) \leq f_{\mu}(x) + \mu \bar{D}$, whereby:

\[
f(x_k) - f^* \leq f_{\mu}(x_k) - f^* + \mu \bar{D} \leq \frac{L_n}{\beta} (f(\hat{x}^0) - f_{\text{slb}})^2 + \mu \bar{D}.
\]

We now apply Proposition 2.3.9 to the setting of the Parametric Smoothing/Restarting Method (Algorithm 4).

**Proposition 2.3.10.** Let $i$ be the index of an outer iteration of Algorithm 4. Define $T := \left[ \frac{G\sqrt{2}A\bar{D}}{\epsilon t} - 1 \right]$. If $k \geq T$ and $x_{i,k}$ exists, then it holds that:

\[
f(x_{i,k}) - f^* \leq 2t(f(x_{i,0}) - f_{\text{slb}}).
\]

**Proof:** The proof follows by applying Proposition 2.3.9 with $\mu = \mu_i = \frac{t(f(x_{i,0}) - f_{\text{slb}})}{\bar{D}}$, $\beta = \frac{\bar{D}}{\epsilon t}$, $Y = T$, and $\hat{x}^0 = x_{i,0}$. It then follows that

\[
f(x_{i,k}) - f^* \leq \frac{L_n}{\beta} (f(\hat{x}^0) - f_{\text{slb}})^2 + \mu \bar{D} \leq \frac{A}{\mu \beta} (f(\hat{x}^0) - f_{\text{slb}})^2 + \mu \bar{D} = 2t(f(x_{i,0}) - f_{\text{slb}}),
\]

where the second inequality uses $L_n \leq \bar{A}/\mu$ from (2.17) and the last equality uses the values of $\mu$ and $\beta$.

**Proposition 2.3.11.** Let $i$ be the index of an outer iteration of Algorithm 4. Define $U := \left[ \frac{G\sqrt{2}A\bar{D}}{\epsilon t} - 1 \right]$. If $k \geq U$ and $y_{i,k}$ exists, then it holds that:

\[
f(y_{i,k}) - f^* \leq 2\epsilon t(f(x_{i,0}) - f_{\text{slb}}).
\]
Proof: The proof follows by applying Proposition 2.3.9 with \( \mu = \mu_i^2 = \frac{t \epsilon' (f(x_i,0) - f_{slb})}{\beta} \), \( \beta = \frac{\overline{A} D}{t^2 (\epsilon')^2} \), \( Y = U \), and \( \hat{x}^0 = x_{i,0} \). It then follows that

\[
\begin{align*}
f(y_i,k) - f^* &\leq \frac{L \mu}{\beta} (f(x_i^0) - f_{slb})^2 + \mu \overline{D} \leq \frac{\overline{A}}{\mu \beta} (f(x_i^0) - f_{slb})^2 + \mu \overline{D} = 2t \epsilon' (f(x_i,0) - f_{slb}) ,
\end{align*}
\]

where the second inequality uses \( L \mu \leq \overline{A}/\mu \) from (2.17) and the final equality derives from substituting in the values of \( \mu \) and \( \beta \).

The next three propositions pertain to Algorithm 4 as well as to a more general setting which will be used in Section 2.4 to prove computational guarantees for algorithms when \( f(\cdot) \) is smooth. The more general setting is described in the body of the following proposition.

**Proposition 2.3.12.** Let \( B, v > 0 \) be constants satisfying \( B - B^2 \geq v, B \geq 2v \). Consider an algorithm with outer and inner iterations indexed with counters \( i \) and \( j \), respectively (such as Algorithm 4), with initial iterate \( x^0 \) that is used to set \( x_{i,0} = y_{i,0} \leftarrow x^0 \) in simultaneous running of the Accelerated Gradient Method using the same indexing notation as in Algorithm 4, and where \( x_{i+1,0} = y_{i+1,0} \leftarrow x_{i,K_i} \) where \( K_i \leftarrow j \) denotes the first index \( j \) for which \( \frac{f(x_{i,j}) - f_{slb}}{f(x_{i,0}) - f_{slb}} < B \). Suppose that there are sequences \( \{J_i\} \) and \( \{I_i\} \) indexed over the outer iteration counter \( i \) such that the following conditions are satisfied:

(i) for all \( k \geq J_i \) it holds that \( f(x_{i,k}) - f^* \leq v (f(x_{i,0}) - f_{slb}) \), and

(ii) for all \( k \geq I_i \) it holds that \( f(y_{i,k}) - f^* \leq v \epsilon' (f(x_{i,0}) - f_{slb}) \).

Let \( p \) denote the number of outer iterations \( i \) for which \( K_i \) is finite. Then

\[
p \leq \left\lfloor \frac{\ln \left( 1 + \frac{f(x^0) - f^*}{f(x^0) - f_{slb}} \right)}{\ln(1/B)} \right\rfloor .
\]

Furthermore, if \( i \geq 1 \) and \( i \leq p - 1 \), then \( K_i \leq J_i \).

**Proof:** If \( p = 0 \) the results follow trivially, so let us suppose that \( p \geq 1 \), whereby \( K_p \) is finite and \( x_{p+1,0} \) exists. It then follows that \( f^* - f_{slb} \leq f(x_{p+1,0}) - f_{slb} \leq
\(B^p(f(x_{1,0}) - f_{slb})\), and taking logarithms yields the proof of the bound on \(p\).

Suppose additionally that \(i \geq 1\) and \(i \leq p - 1\). Let us assume that \(K_i \geq J_i + 1\), from which we will derive a contradiction. We have

\[
f(x_{i,K_i-1}) - f* \leq v(f(x_{i,0}) - f_{slb}) \leq (B - B^2)(f(x_{i,0}) - f_{slb}) ,
\]

where the first inequality uses condition (i) and the second inequality uses \(B - B^2 \geq v\). Also, \(i + 2 \leq p + 1\), whereby \(x_{i+2,0}\) exists and therefore satisfies \(f^* - f_{slb} \leq f(x_{i+2,0}) - f_{slb} \leq B^2(f(x_{i,0}) - f_{slb})\). Combining this inequality with that above yields \(f(x_{i,K_i-1}) - f* \leq B(f(x_{i,0}) - f_{slb}) - f^* + f_{slb}\), which rearranges to yield:

\[
\frac{f(x_{i,K_i-1}) - f_{slb}}{f(x_{i,0}) - f_{slb}} \leq B ,
\]

and which contradicts the definition of \(K_i\). Therefore \(K_i \leq J_i\). 

\[\square\]

**Proposition 2.3.13.** Under the same setting, notation, and conditions (i) and (ii) of Proposition 2.3.12, let \(i\) be the index of an outer iteration. If \(j \geq J_i\) and \(x_{i,j+1}\) exists, then:

\[
\frac{f(x_{i,0}) - f_{slb}}{f^* - f_{slb}} \leq \frac{1}{B - v} .
\]

Furthermore, if also \(j \geq \max\{J_i, I_i\}\), then

\[
\frac{f(y_{i,j}) - f^*}{f^* - f_{slb}} \leq \varepsilon' .
\]

**Proof:** Since \(j \geq J_i\) it follows from condition (i) that

\[
f(x_{i,j}) - f^* \leq v(f(x_{i,0}) - f_{slb}) ,
\]

and also since \(x_{i,j+1}\) exists then \(K_i \geq j + 1\), whereby:

\[
\frac{f(x_{i,j}) - f_{slb}}{f(x_{i,0}) - f_{slb}} \geq B .
\]
It then follows from these two inequalities that

\[
\frac{f(x_{i,0}) - f_{slb}}{f^* - f_{slb}} = \frac{1}{\frac{f(x_{i,j}) - f_{slb}}{f(x_{i,0}) - f_{slb}} - \frac{f^* - f_{slb}}{f(x_{i,0}) - f_{slb}}} \leq \frac{1}{B - v} .
\]

(2.20)

If also \( j \geq I_i \), then we have from condition (ii) that

\[
f(y_{i,j}) - f^* \leq v\varepsilon'(f(x_{i,0}) - f_{slb}) \leq \frac{v\varepsilon'}{B - v}(f^* - f_{slb}) \leq (f^* - f_{slb})\varepsilon' ,
\]

where the first inequality is from condition (ii), the second inequality uses (2.20), and the third inequality uses \( B \geq 2v \).

\[\square\]

**Proposition 2.3.14.** Under the same setting, notation, and conditions (i) and (ii) of Proposition 2.3.12, let \( \hat{N} \) count the total number of inner iterations prior to and including the first iteration for which \( y_{i,j} \) is an \( \varepsilon' \)-relative solution (2.2). Then

either (i) \( \hat{N} \leq \sum_{i=1}^{p+1} J_i + I_{p+1} \), or (ii) \( \hat{N} \leq \sum_{i=1}^{p+1} J_i + I_p + I_{p+1} \) and \( K_p \geq J_p + 1 \).

(2.21)

**Proof:** First consider the case when \( p = 0 \). Then \( K_1 = +\infty \) and therefore with \( i = 1 \) we have \( x_{i,j+1} \) exists for \( j = \max\{J_1, I_1\} \), whereby from Proposition 2.3.13 it holds that \( y_{i,j} \) satisfies (2.2). In this case \( \hat{N} \leq j = \max\{J_1, I_1\} \leq J_1 + I_1 = \sum_{i=1}^{p+1} J_i + I_{p+1} \) and therefore (i) of (2.21) is satisfied.

Next consider the case where \( p \geq 1 \) and \( K_p \geq \max\{J_p, I_p\} + 1 \). Let \( i \) be the index of an outer iterate. If \( i \leq p - 1 \) it follows from Proposition 2.3.12 that \( K_i \leq J_i \). For \( i = p \) it holds for this case that \( K_p \geq \max\{J_p, I_p\} + 1 \), and it follows from Proposition 2.3.13 that \( x_{p,j+1} \) exists for \( j = \max\{J_p, I_p\} \) and therefore \( y_{p,j} \) satisfies (2.2). In this case \( \hat{N} \leq \sum_{i=1}^{p-1} K_i + \max\{J_p, I_p\} \leq \sum_{i=1}^{p-1} J_i + \max\{J_p, I_p\} \leq \sum_{i=1}^{p} J_i + I_p \) and \( K_p \geq J_p + 1 \) whereby (ii) of (2.21) is satisfied.

Next consider the case where \( p \geq 1 \) and \( K_p \leq \max\{J_p, I_p\} \) and also \( K_p \leq J_p \). Let \( i \) be the index of an outer iterate. If \( i \leq p - 1 \) it follows from Proposition 2.3.12 that \( K_i \leq J_i \). Since \( K_{p+1} = +\infty \) it follows that \( x_{p+1,j+1} \) exists for \( j = \max\{J_{p+1}, I_{p+1}\} \),
whereby from Proposition 2.3.13 we have \( y_{p+1,j} \) satisfies (2.2). And since \( K_p \leq J_p \) in this case, it follows that \( \hat{N} \leq \sum_{i=1}^{p-1} K_i + J_p + \max\{J_{p+1}, I_{p+1}\} \leq \sum_{i=1}^{p+1} J_i + I_{p+1} \), and therefore (i) of (2.21) is satisfied.

The last case is where \( p \geq 1 \) and \( K_p \leq \max\{J_p, I_p\} \) and also \( K_p \geq J_p + 1 \). Then just as in the third case above, we arrive at \( \hat{N} \leq \sum_{i=1}^{p-1} K_i + \max\{J_p, I_p\} + \max\{J_{p+1}, I_{p+1}\} \leq \sum_{i=1}^{p+1} J_i + I_p + I_{p+1} \), and thus (ii) of (2.21) is satisfied, thereby proving (2.21).

\[ \square \]

**Proof of Theorem 2.3.3:** Algorithm 4 satisfies the setting of Proposition 2.3.12, and it follows from Propositions 2.3.10 and 2.3.11 that Algorithm 4 satisfies conditions (i) and (ii) of Proposition 2.3.12 by letting \( v = 2t \), \( J_i = T \), and \( I_i = U \) for all outer iterations \( i \). Therefore the conclusions of Propositions 2.3.12, 2.3.13, and 2.3.14 all hold true. Let \( N \) denote the total number of iterates of Algorithm 4 computed prior to and including the first iterate \( y_{i,j} \) that is an \( \epsilon' \)-relative solution (2.2). Since two iterates are computed at each iteration, we have \( N = 2\hat{N} \) (where \( \hat{N} \) is defined in Proposition 2.3.14) and it follows from Proposition 2.3.14 that \( N = 2\hat{N} \leq 2 \sum_{i=1}^{p+1} J_i + 2I_p + 2I_{p+1} \), since the right-side of this inequality dominates both bounds (i) and (ii) of (2.21). Substituting in the values of \( T \) and \( U \) and the bound on \( p \) from Proposition 2.3.12 we obtain:

\[ N \leq 2(p + 1)T + 4U \leq 2 \left[ 1 + \frac{\ln\left(1 + \frac{t\phi - f^*}{f^* - f_{\text{stab}}}\right)}{\ln(1/B)} \right] \left[ \frac{G\sqrt{2AD}}{t} - 1 \right] + 4 \left[ \frac{G\sqrt{2AD}}{\epsilon't} - 1 \right] \]

\[ \leq G\sqrt{AD} \left( 22.63 + 32.65 \ln \left(1 + \frac{t\phi - f^*}{f^* - f_{\text{stab}}}\right) + 45.26 \left(\frac{1}{\epsilon'}\right) \right), \]

where the third inequality follows from substituting in the values \( B = \frac{1}{2} \) and \( t = \frac{1}{8} \), which then rounds up to the desired bound in the theorem.

\[ \square \]
2.4 Computational Guarantees when \( f(\cdot) \) is Smooth

In this section we study the computational complexity of solving (4.1) in the case when \( f(\cdot) \) is convex and differentiable on an open set containing \( Q \). We assume that \( \nabla f(\cdot) \) is Lipschitz on \( Q \) as defined in (2.8).

Let us first consider directly applying the Accelerated Gradient Method (Algorithm 2) to solve (4.1), and let us apply Theorem 2.2.2. Let \( \varepsilon' > 0 \) denote the relative accuracy, and note again that an \( \varepsilon' \)-relative solution of (4.1) corresponds to an absolute \( \varepsilon \)-solution for \( \varepsilon := \varepsilon' \cdot (f^* - f_{\text{slb}}) \). Let \( x^0 \in Q \) be the initial point. It then follows from Theorem 2.2.2 using \( S_\delta = \{ x \in Q : f(x) \leq f^* \} = \text{Opt} \) that if

\[
N \geq \frac{\sqrt{2\sqrt{L}} \text{Dist}(x^0, \text{Opt})}{\sqrt{\varepsilon'} \sqrt{f^* - f_{\text{slb}}} - 1},
\]

then

\[
\frac{f(x^N) - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon'.
\]

Herein we will derive a new computational guarantee for a version of the Accelerated Gradient Method that can improve on (2.22) in many cases. Our new version of the Accelerated Gradient Method periodically restarts the method with an appropriate rule for deciding when to do the restarts, and is presented in Algorithm 5. At the \( i \)th outer iteration of Algorithm 5 the algorithm starts the Accelerated Gradient Method at the point \( x_{i,0} \) for optimizing \( f(\cdot) \) on \( Q \). The outer iteration \( i \) runs until the ratio test in Step (2a.) fails, at which point the current point \( x_{i,j} \) becomes the starting point of the next outer iteration, namely \( x_{i+1,0} \leftarrow x_{i,j} \). The counter \( K_i \) records the number of inner iterations computed in outer iteration \( i \). Similar to the notation in Algorithm 4, the notation “\( x_{i,j} \leftarrow \text{AGM}(f(\cdot), x_{i,0}, j) \)” in Algorithm 5 denotes assigning to \( x_{i,j} \) the \( j \)th iterate of the Accelerated Gradient Method applied to the optimization problem (4.1) with objective function \( f(\cdot) \) using the initial point \( x_{i,0} \in Q \).
Algorithm 5 Accelerated Gradient Method with Simple Restarting

Initialize. Initialize with $x^0 \in Q$.
Define $B := 0.5$
Set $x_{1,0} \leftarrow x^0$, $i \leftarrow 1$.

At outer iteration $i$:
1. Initialize inner iteration. $K_i \leftarrow +\infty$, $j \leftarrow 0$
2. Run inner iterations. At inner iteration $j$:
   (2a.) If $\frac{f(x_{i,j}) - f_{\text{slb}}}{f(x_{i,0}) - f_{\text{slb}}} \geq B$, then
   \begin{align*}
x_{i,j+1} &\leftarrow \text{AGM}(f(\cdot), x_{i,0}, j + 1), \\
j &\leftarrow j + 1, \text{ and Goto (2a.)}.
   \end{align*}
   Else $K_i \leftarrow j$, $x_{i+1,0} \leftarrow x_{i,j}$, $i \leftarrow i + 1$, and Goto step 1.

We have the following computational guarantee associated with Algorithm 5.

Theorem 2.4.1. (Complexity Bound for Accelerated Gradient Method with Simple Restarting) Within a total number of computed iterates that does not exceed

$$G\sqrt{L} \left(10\sqrt{f(x^0) - f_{\text{slb}}} + 12 \left[\frac{\sqrt{f^* - f_{\text{slb}}}}{\varepsilon'}\right]\right),$$

the Accelerated Gradient Method with Simple Restarting (Algorithm 5) will compute an iterate $x_{i,j}$ for which

$$\frac{f(x_{i,j}) - f^*}{f^* - f_{\text{slb}}} \leq \varepsilon'.$$

The computational guarantee in Theorem 2.4.1 can itself be bounded by:

$$G\sqrt{L} \left(10\sqrt{f^* - f_{\text{slb}}} + 10\sqrt{\frac{3}{2}\text{Dist}(x^0, \text{Opt})} + 12 \left[\frac{\sqrt{f^* - f_{\text{slb}}}}{\varepsilon'}\right]\right),$$

(2.23)
which follows from the chain of inequalities:

\[
\sqrt{f(x^0) - f_{slb}} = \sqrt{(f^* - f_{slb}) + (f(x^0) - f^*)}
\]

\[
\leq \sqrt{(f^* - f_{slb}) + \frac{1}{2} \text{Dist}(x^0, \text{Opt})^2} \leq \sqrt{(f^* - f_{slb}) + \frac{1}{2} \text{Dist}(x^0, \text{Opt})}.
\]

Comparing (2.23) with the standard bound for the Accelerated Gradient Method given in (2.22), we see that the factor involving $1/\sqrt{\varepsilon'}$ in (2.23) is independent of \text{Dist}(x^0, \text{Opt}), unlike the standard bound (2.22).

Towards the proof of Theorem 2.4.1, for notational convenience we will work with two constants $B$ and $v$ that must be chosen to satisfy

\[
B > 0, \quad v > 0, \quad B - B^2 \geq v, \quad \text{and} \quad B \geq 2v,
\]

and whose specific values are set to $B = 0.5$ in Algorithm 4, and $v = 0.25$.

**Proposition 2.4.1.** Let $i$ be the index of an outer iteration of Algorithm 5. Define $J_i := \left[ G \sqrt{\frac{2L(f(x_i,0) - f_{slb})}{v}} - 1 \right]$. If $k \geq J_i$ and $x_{i,k}$ exists, then it holds that:

\[
f(x_{i,k}) - f^* \leq v(f(x_{i,0}) - f_{slb}).
\]

**Proof:** It follows from Theorem 2.2.2 applied to the function $f(\cdot)$ and using $\delta = f^*$ that for any $k \geq J_i$ we have:

\[
f(x_{i,k}) - f^* \leq \frac{2L}{(J_i + 1)^2} \text{Dist}(x_{i,0}, \text{Opt})^2 \leq \frac{2L}{(J_i + 1)^2} G^2 (f(x_{i,0}) - f_{slb})^2 \leq v (f(x_{i,0}) - f_{slb}),
\]

where the second inequality uses the definition of $G$, and the last inequality uses the value of $J_i$. \qed

**Proposition 2.4.2.** Let $i$ be the index of an outer iteration of Algorithm 5. Define
If \( k \geq I_i \) and \( x_{i,k} \) exists, then it holds that:

\[
 f(x_{i,k}) - f^* \leq \varepsilon'(f(x_{i,0}) - f_{\text{slb}}).
\]

**Proof:** The proof follows using identical logic as in Proposition 2.4.1.

**Proof of Theorem 2.4.1:** Even though Algorithm 5 does not simultaneously run two versions of the Accelerated Gradient Method, we can still view Algorithm 5 as an instance of the general algorithm setting of Proposition 2.3.12 by simply defining \( y_{i,j} := x_{i,j} \) for all \( i,j \). It follows from Propositions 2.4.1 and 2.4.2 that Algorithm 5 satisfies conditions (i) and (ii) of Proposition 2.3.12, and therefore Propositions 2.3.12, 2.3.13, and 2.3.14 hold for Algorithm 5. Substituting in the values of \( J_i \) and using the fact that \( f(x_{i,0}) - f_{\text{slb}} \leq B^{i-1}(f(x_{1,0}) - f_{\text{slb}}) \) for all iteration counters \( i \), we obtain:

\[
\sum_{i=1}^{p+1} J_i \leq \sum_{i=1}^{p+1} G \sqrt{\frac{2L(f(x_{i,0}) - f_{\text{slb}})}{v}} \leq \left( \sum_{i=0}^{p} B^\frac{i}{2} \right) G \sqrt{\frac{2L(f(x_{1,0}) - f_{\text{slb}})}{v}} \leq \left( \sum_{i=0}^{\infty} B^\frac{i}{2} \right) G \sqrt{\frac{2L(f(x_0) - f_{\text{slb}})}{v}} = \frac{G}{1 - \sqrt{B}} \sqrt{\frac{2L(f(x_0) - f_{\text{slb}})}{v}}.
\]

Next observe that \( K_{p+1} = \infty \geq J_p \), whereby it follows from Proposition 2.3.13 with \( i = p + 1 \) that \( f(x_{p+1,0}) - f_{\text{slb}} \leq \frac{1}{B-v}(f^* - f_{\text{slb}}) \), and therefore it holds that:

\[
I_{p+1} \leq G \sqrt{\frac{2L(f(x_{p+1,0}) - f_{\text{slb}})}{v\varepsilon'}} \leq G \sqrt{\frac{2L(f^* - f_{\text{slb}})}{(B-v)v\varepsilon'}}.
\]

Also, if \( K_p \geq J_p + 1 \), then similarly applying Proposition 2.3.13 with \( i = p \) using the logic above implies that \( I_p \leq G \sqrt{\frac{2L(f^* - f_{\text{slb}})}{(B-v)v\varepsilon'}} \).

56
Let $N$ denote the total number of iterates of Algorithm 5 computed prior to and including the first iterate $x_{i,j}$ that is an $\varepsilon'$-relative solution (2.2). Then $N = \hat{N}$ where $\hat{N}$ is defined in Proposition 2.3.14. In either case (i) or (ii) of (2.21), it follows from Proposition 2.3.14 that:

$$
N = \hat{N} \leq \sum_{i=1}^{p+1} J_i + I_{p+1} + G \sqrt{\frac{2L(f^* - f_{slb})}{(B - v)\varepsilon'}}
$$

$$
\leq \frac{G}{1 - \sqrt{B}} \frac{2L(f(x^0) - f_{slb})}{v} + 2G \sqrt{\frac{2L(f^* - f_{slb})}{(B - v)\varepsilon'}}
$$

$$
\leq G\sqrt{L} \left(9.66 \sqrt{f(x^0) - f_{slb}} + \frac{11.32}{\sqrt{\varepsilon'}}\right),
$$

where the third inequality follows from substituting in the values $B = \frac{1}{2}$ and $v = \frac{1}{4}$, which then rounds up to the bound stated in the theorem.

It turns out that we can further improve the computational guarantee of Theorem 2.4.1 by further modifying the Accelerated Gradient Method with Simple Restarting (Algorithm 5), if we know and can easily work with an adjoint representation of $f(\cdot)$ to do “extra smoothing.” Let us see how this can be done. We will assume that $f(\cdot)$ has the representation:

$$
f(x) = \max_{\lambda \in P} \{\lambda^T Ax - d(\lambda)\}, \tag{2.26}
$$

where $P$ is a convex set and $d(\cdot)$ is a strongly convex function on $P$ with strong convexity parameter $\sigma$ and for which $\min_{\lambda \in P} d(\lambda) \geq 0$. (See [76] for properties of strongly convex functions.) It then follows that $f(\cdot)$ is a globally smooth convex function with Lipschitz constant at most $L := \|A\|^2/\sigma$, see Nesterov [78]. We presume further that $A$, $d(\cdot)$, and $P$ are given and that the optimization problem in (2.26) is simple to solve. That being the case, for a given $x \in Q$, if $\tilde{\lambda}$ solves the optimization problem (2.26), then it holds that $f(x) = \tilde{\lambda}^T Ax - d(\tilde{\lambda})$ and $\nabla f(x) = A^T \tilde{\lambda}$.

In a similar spirit as the smoothing technique employed in Section 2.3.3, we will consider parametrically working with a modification $f_\mu(\cdot)$ of $f(\cdot)$ that is more smooth...
than $f(\cdot)$ by increasing the weight on the strongly convex function $d(\cdot)$ in (2.26).

For any $\mu \geq 0$ define the function $f_\mu(\cdot)$ by:

$$f_\mu(x) = \max_{\lambda \in P}\{\lambda^T Ax - (1 + \mu)d(\lambda)\}.$$  \hspace{1cm} (2.27)

If $P$ is bounded, then $\bar{D} := \max_{\lambda \in P}\{d(\lambda)\}$ is finite, and the above smoothing technique has the following two properties:

(i) $f_\mu(\cdot)$ is not far from $f(\cdot)$,

$$f(x) - \bar{D}\mu \leq f_\mu(x) \leq f(x) \text{ for all } x \in Q, \text{ and}$$  \hspace{1cm} (2.28)

(ii) $f_\mu(\cdot)$ has Lipschitz continuous gradient on $Q$ with Lipschitz constant $L_\mu$ satisfying

$$L_\mu \leq L/(1 + \mu).$$  \hspace{1cm} (2.29)

This setting is very similar to the properties we have for smoothing of a non-smooth function $f(\cdot)$ in Section 2.3.3, and the only difference is that the Lipschitz constant $L_\mu$ here is bounded above by $L/(1 + \mu)$ instead of by $\bar{A}/\mu$ as was the case in (2.17).

Let $\varepsilon' > 0$ be given. As before, we aspire to compute an $\varepsilon'$-relative solution of (4.1) as defined in (2.2). We will use and analyze the Parametric Smoothing/Rescaling Method (Algorithm 4) but with $f_\mu(\cdot)$ defined by (2.27) and hence satisfying (2.28) and (2.29). We have the following computational guarantee associated with Algorithm 4 applied to the case when $f(\cdot)$ is smooth and $f_\mu(\cdot)$ is given by (2.27).

**Theorem 2.4.2. (Complexity Bound for Parametric Smoothing/Restarting Method (Algorithm 4) for Smooth Optimization)** Suppose that $f_\mu(\cdot)$ is given by (2.27) and hence satisfies (2.28) and (2.29). Within a total number of computed
iterates that does not exceed
\[
G \sqrt{LD} \left(22.7 + 32.7 \ln \left(1 + \frac{f(x^0) - f^*}{f^* - f_{slb}}\right) + 32 \frac{f^* - f_{slb}}{\varepsilon'}\right),
\]

Algorithm 4 will compute an iterate \( y_{i,j} \) for which
\[
\frac{f(y_{i,j}) - f^*}{f^* - f_{slb}} \leq \varepsilon'.
\]

The dependence in Theorem 2.4.2 on the quality of the initial point is logarithmic in the optimality gap \( f(x^0) - f^* \), while it is the square root of the optimality gap in Theorem 2.4.1. We will prove Theorem 2.4.2 by first proving two propositions. For notational convenience we will work with two constants \( B \) and \( t \), whose specific values are \( B = \frac{1}{2} \) and \( t = \frac{1}{8} \).

**Proposition 2.4.3.** Let \( i \) be the index of an outer iteration of Algorithm 4. Define \( T := \left\lfloor G \sqrt{\frac{2L\bar{D}}{t}} - 1 \right\rfloor \). If \( k \geq T \) and \( x_{i,k} \) exists, then:
\[
f(x_{i,k}) - f^* \leq 2t(f(x_{i,0}) - f_{slb}).
\]

**Proof:** The proof follows by applying Proposition 2.3.9 with \( \mu = \mu'_1 = \frac{t(f(x_{i,0}) - f_{slb})}{\bar{D}}, \beta = \frac{LD}{t^2}, Y = T, \) and \( \hat{x}^0 = x_{i,0} \). It then follows that
\[
f(x_{i,k}) - f^* \leq \frac{L}{\beta} \left( f(\hat{x}^0) - f_{slb} \right)^2 + \mu \bar{D} \leq \frac{L}{\mu \beta} \left( f(x_{i,0}) - f_{slb} \right)^2 + \mu \bar{D} = 2t(f(x_{i,0}) - f_{slb}),
\]
where the second inequality uses \( L_\mu \leq L/(1 + \mu) \leq L/\mu \) from (2.29) and the final equality derives from substituting in the values of \( \mu \) and \( \beta \).

**Proposition 2.4.4.** Let \( i \) be the index of an outer iteration of Algorithm 4. Define \( I_i := \left\lfloor G \sqrt{2L(f(x_{i,0}) - f_{slb})} - 1 \right\rfloor \). If \( k \geq I_i \) and \( y_{i,k} \) exists, then:
\[
f(y_{i,k}) - f^* \leq 2t\varepsilon'(f(x_{i,0}) - f_{slb}).
\]

59
Proof: The proof follows by applying Proposition 2.3.9 with \(\mu = \mu_i^2 = \frac{t\varepsilon'}{D} (f(x_i^0) - f_{slb})\), \(\beta = \frac{L(f(x_i^0) - f_{slb})}{t\varepsilon'}\), \(Y = I_i\), and \(\tilde{x}^0 = x_i^0\). It then follows that

\[
f(y_{i,k}) - f^* \leq \frac{L\mu}{\beta} (f(\tilde{x}^0) - f_{slb})^2 + \mu \bar{D} \leq \frac{L}{\beta} (f(x_i^0) - f_{slb})^2 + \mu \bar{D} = 2t\varepsilon'(f(x_i^0) - f_{slb}),
\]

where the second inequality uses \(L \mu \leq L/(1 + \mu) \leq L\) from (2.29) and the final equality derives from substituting in the values of \(\mu\) and \(\beta\). \(\square\)

Proof of Theorem 2.4.2: Algorithm 4 satisfies the setting of Proposition 2.3.12, and it follows from Propositions 2.4.3 and 2.4.4 that Algorithm 4 satisfies conditions (i) and (ii) of Proposition 2.3.12 by letting \(v = 2t\) and \(J_i = T\) for all outer iterations \(i\). Therefore the conclusions of Propositions 2.3.12, 2.3.13, and 2.3.14 all hold true.

Let \(N\) denote the total number of iterates of Algorithm 4 computed prior to and including the first iterate \(y_{i,j}\) that is an \(\varepsilon'\)-relative solution (2.2). Since two iterates are computed at each iteration, we have \(N = 2\tilde{N}\), where \(\tilde{N}\) is defined in Proposition 2.3.14 and is bounded by either (i) or (ii) of (2.21).

Note that \(K_{p+1} = \infty \geq T = J_i\), whereby it follows from Proposition 2.3.13 that \(f(x_{p+1,0}) - f_{slb} \leq \frac{1}{B-2t} (f^* - f_{slb})\), and therefore

\[
I_{p+1} \leq G\sqrt{\frac{2L(f(x_{p+1,0}) - f_{slb})}{t\varepsilon'}} \leq G\sqrt{\frac{2L(f^* - f_{slb})}{(B-2t)t\varepsilon'}}.
\]

Similarly, if \(K_p \geq T + 1 = J_i + 1\), similar logic demonstrates that \(I_p \leq G\sqrt{\frac{2L(f^* - f_{slb})}{(B-2t)t\varepsilon'}}\). Therefore, in either case (i) or (ii) of (2.21) it holds that:
\[ N = 2 \hat{N} \leq 2 \sum_{i=1}^{p+1} J_i + 2I_{p+1} + 2G \sqrt{\frac{2L(f^* - f_{\text{lab}})}{(B-2t)te'}} \]

\[ \leq 2(p+1) \left[ G \sqrt{\frac{2L}{t^2}} - 1 \right] + 2I_{p+1} + 2G \sqrt{\frac{2L(f^* - f_{\text{lab}})}{(B-2t)te'}} \]

\[ \leq 2 \left( 1 + \frac{\ln\left(1 + \frac{f(x^0) - f^*}{f^* - f_{\text{lab}}} \right)}{\ln(1/B)} \right) G \sqrt{\frac{2L}{t^2}} + 4G \sqrt{\frac{2L(f^* - f_{\text{lab}})}{(B-2t)te'}} \]

\[ \leq G \sqrt{LD} \left( 22.7 + 32.7 \ln \left( 1 + \frac{f(x^0) - f^*}{f^* - f_{\text{lab}}} \right) + 32 \sqrt{\frac{f^* - f_{\text{lab}}}{te'}} \right), \]

where the third inequality follows from substituting in the values \( B = \frac{1}{2} \) and \( t = \frac{1}{8} \), which then rounds up to the desired bound in the theorem. \( \square \)
Chapter 3

Relatively Smooth Convex Optimization by First-Order Methods, and Applications

3.1 Introduction, Definition of “Relative-Smoothness,” and Basic Properties

3.1.1 Traditional Set-up for Smooth First-Order Methods

Our optimization problem of interest is:

\[ P : f^* := \min_{x} f(x) \]  \hspace{1cm} (3.1)

\[ \text{s.t.} \quad x \in Q , \]

where \( Q \subseteq \mathbb{E} \) is a closed convex set in the finite-dimensional vector space \( \mathbb{E} \) with inner product \( \langle \cdot , \cdot \rangle \), and \( f(\cdot) : Q \to \mathbb{R} \) is a differentiable convex function.

There are by now very many first-order methods for tackling the optimization problem (4.1), see for example [76], [105], [87]; virtually all such methods are designed
to solve (4.1) when the gradient of \( f(\cdot) \) satisfies a uniform Lipschitz condition on \( Q \), namely there exists a constant \( L_f < \infty \) for which:

\[
\|\nabla f(x) - \nabla f(y)\|_* \leq L_f \|x - y\| \quad \text{for all } x, y \in Q ,
\]

(3.2)

where \( \| \cdot \| \) is a given norm on \( \mathbb{R} \) and \( \| \cdot \|_* \) denotes the usual dual norm. For example, consider the standard gradient descent scheme, which presumes the norm in (4.2) is Euclidean, and uses the following update:

\[
x^{i+1} \leftarrow \arg \min_{x \in Q} \left\{ f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + \frac{L_f}{2} \|x - x^i\|_2^2 \right\} .
\]

(3.3)

One can prove for the standard gradient descent scheme that after \( k \) iterations it holds for any \( x \in Q \) that:

\[
f(x^k) - f(x) \leq \frac{L_f \|x - x^0\|_2^2}{2k} ,
\]

(3.4)

which is an \( O(1/k) \) sublinear rate of convergence [76], [87]. Furthermore, if \( f(\cdot) \) is also uniformly \( \mu_f \)-strongly convex for some \( \mu_f > 0 \), namely:

\[
f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_f}{2} \|y - x\|_2^2 \quad \text{for all } x, y \in Q ,
\]

(3.5)

then one can prove linear convergence for the gradient descent scheme, see [76], [87], i.e., for any \( x \in Q \) we have that:

\[
f(x^k) - f(x) \leq \frac{L_f}{2} \left( 1 - \frac{2 \mu_f}{L_f + \mu_f} \right)^k \|x - x^0\|_2^2 .
\]

(3.6)

More general versions of first-order methods are not restricted to the Euclidean \( (\| \cdot \|_2) \) norm, and use a differentiable “prox function” \( h(\cdot) \), which is a 1-strongly convex function on \( Q \), to define a Bregman distance:

\[
D_h(y, x) := h(y) - h(x) - \langle \nabla h(x), y - x \rangle \quad \text{for all } x, y \in Q
\]

(3.7)
which as a result satisfies

\[ D_h(y, x) \geq \frac{1}{2} \| y - x \|^2 . \]

The standard Primal Gradient Scheme (with Bregman distance), see [105], has the following update formula:

\[ x^{i+1} \leftarrow \arg \min_{x \in Q} \left\{ f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + L_f D_h(x, x^i) \right\} . \] (3.8)

Notice in (4.6) by construction that the update requires the capability to solve instances of a subproblem of the general form:

\[ x_{\text{new}} \leftarrow \arg \min_{x \in Q} \left\{ \langle c, x \rangle + h(x) \right\} , \] (3.9)

for suitable iteration-specific values of \( c \); indeed, (4.6) is an instance of the subproblem (4.7) with \( c = \frac{1}{L_f} \nabla f(x^i) - \nabla h(x^i) \) at iteration \( i \). It is especially important to note that the Primal Gradient Scheme is somewhat meaningless whenever we do not have the capability to efficiently solve (4.7), a point which we will return to later on. In a typical design and implementation of a first-order method for solving (4.1), one attempts to specify the norm \( \| \cdot \| \) and the strongly convex prox function \( h(\cdot) \) in consideration of the shape of the feasible domain \( Q \) while also ensuring that the subproblem (4.7) is efficiently solvable.

Regarding computational guarantees, one can prove for the Primal Gradient Scheme that after \( k \) iterations it holds for any \( x \in Q \) that:

\[ f(x^k) - f(x) \leq \frac{L_f D_h(x, x^0)}{k} , \] (3.10)

which is an exact generalization of (4.5), see [105], [71].

We emphasize that standard first-order methods as stated above for solving (4.1) require that \( f(\cdot) \) be uniformly smooth on \( Q \), that is, that there is a finite value of the Lipschitz constant \( L_f \) as defined in (4.2), in order to ensure associated computational guarantees. However, there are many differentiable convex functions in practice that
do not satisfy a uniform smoothness condition. Consider $f(x) := -\ln \det(HXH^T)$ with $X := \text{Diag}(x)$ in $D$-optimal design on the feasible set $Q = \{x \in \mathbb{R}^n : \langle e, x \rangle = 1, x \geq 0\}$, or $f(x) = |x|^3$ or $f(x) = x^4$ on the feasible set $Q = \mathbb{R}$, or $f(x) = -\ln(x) + x^2$ on $Q = \mathbb{R}_{++}$. Of course, if the algorithm iterates have monotone decreasing objective function values (which is provably the case for most smooth first-order methods), it then is sufficient just to ensure that $f(\cdot)$ is smooth on some level set of $f(\cdot)$. Nevertheless, even in this case the constant $L_f$ may be huge. For instance, let $f(x) = -\ln(x) + x^2$ on $Q = \mathbb{R}_{++}$, and consider the level set $\{x : f(x) \leq 10\}$. Then one still has $L_f \approx \exp^{20}$ on this level set, which is not reasonable for practical use.

Notice that unlike quadratic functions, the second-order terms of the functions in the above examples vary dramatically on $Q$ – and especially as $x \to \partial Q$ (or as $x$ goes to infinity in $Q$). It therefore becomes unreasonable to use a uniform bound of the form $L_f$ to upper-bound second-order information.

Motivated by the above drawbacks in standard first-order methods, we develop a notion of “relative smoothness” and relative strong convexity, relative to a given “reference function” $h(\cdot)$ and which does not require the specification of any particular norm – and indeed $h(\cdot)$ need not be either strictly or strongly convex. Armed with relative smoothness and relative strong convexity, we demonstrate the capability to solve a more general class of differentiable convex optimization problems (without uniform Lipschitz continuous gradients), and we also demonstrate linear convergence results for both a Primal Gradient Scheme and a Dual Averaging Scheme when the function is both relatively smooth and relatively strongly convex.

There is a certain overlap of ideas and results herein with the paper [8] by Bolte, Bauschke, and Teboulle. For starters, the relative smoothness condition definition in the present chapter in Definition 3.1.1 is equivalent to the (LC) condition in [8] except that [8] also requires the reference function $h(\cdot)$ to be essentially smooth and strictly convex, which we do not need in this chapter. The main developments in [8] are based on generalizing a key descent lemma and applying this generalization to tackle (additive) composite optimization problems using the primal gradient scheme (called the
NoLips Algorithm in [8]) with associated complexity analysis involving a symmetry measure of the Bregman distance $D_h(\cdot, \cdot)$. These results are then illustrated in the application of composite optimization to Poisson inverse problems. While the NoLips Algorithm in [8] is structurally the same as Algorithm 6 herein, they are both instantiations of the standard primal gradient scheme; however, as will be seen in Section 4.4 here, we do not need any symmetry measure in constructing step-sizes or in the complexity analysis. The paper [118] by Zhou, Liang, and Shen also tackles composite optimization using the standard primal gradient scheme which therein is called PGA-\textit{B}, with a focus on demonstrating equivalence of proximal gradient and proximal point methods more broadly. Here we develop measures of relative smoothness and also relative strong convexity, which can improve the computational guarantees of the primal gradient scheme, see Theorem 3.3.1. We further present computational guarantees for the dual averaging scheme [73] in Theorem 3.3.2. In Section 3.2 we show that many differentiable convex functions are relatively smooth with respect to a correspondingly fairly-simple reference function $h(\cdot)$ that is easy to construct and for which algorithmic computations can be efficiently be performed. In Section 3.4 we apply our approach to develop a new first-order method for the $D$-optimal design problem, with associated computational complexity analysis. Throughout the current chapter, we compare and clarify similarities and differences between our work and [8] in the context of the specific contributions as they arise.

### 3.1.2 Relative Smoothness and Relative Strong Convexity

Let $h(\cdot)$ be any given differentiable convex function (it need not be strongly nor even strictly convex) defined on $Q$. We will henceforth refer to $h(\cdot)$ as the “reference function.” We define “relative smoothness” and “relative strong convexity” of $f(\cdot)$ relative to $h(\cdot)$ using the Bregman distance (3.7) associated with $h(\cdot)$ as follows.

**Definition 3.1.1.** $f(\cdot)$ is $L$-smooth relative to $h(\cdot)$ on $Q$ if for any $x, y \in \text{int } Q$, there

67
is a scalar $L$ for which

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + LD_h(y, x) .$$  \hspace{1cm} (3.11)

**Definition 3.1.2.** $f(\cdot)$ is $\mu$-strongly convex relative to $h(\cdot)$ on $Q$ if for any $x, y \in \text{int } Q$, there is a scalar $\mu \geq 0$ for which

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \mu D_h(y, x) .$$  \hspace{1cm} (3.12)

(Here and elsewhere $\text{int } Q$ denotes the interior of $Q$. In cases where $Q$ has no interior, one can instead use the relative interior of $Q$.) Note that relative smoothness and relative strong convexity of $f(\cdot)$ are defined relative to the reference function $h(\cdot)$ directly; no norm is involved in the definitions, so that smoothness/strong convexity does not depend on any norm. Furthermore, $h(\cdot)$ is not presumed to have any special properties by itself such as strict or (traditional) strong convexity; rather the key structural properties involve how $f(\cdot)$ behaves relative to $h(\cdot)$. The definition of relative smoothness above is equivalent to the (LC) condition in [8], but [8] requires the reference function to be essentially smooth and strictly convex, which we do not need.

The following proposition presents equivalent definitions of relative smoothness and relative strong convexity. In the case when both $f(\cdot)$ and $h(\cdot)$ are twice differentiable, parts (a-iii) and (b-iii) of the proposition demonstrate that the above definitions are equivalent to

$$\mu \nabla^2 h(x) \preceq \nabla^2 f(x) \preceq L \nabla^2 h(x) \quad \text{for all } x \in \text{int } Q ,$$

which is an intuitively simple condition on the Hessian matrices of the two functions.

**Proposition 3.1.1.** The following conditions are equivalent:

(a-i) $f(\cdot)$ is $L$-smooth relative to $h(\cdot)$,
(a-ii) \( Lh(\cdot) - f(\cdot) \) is a convex function on \( Q \),

(a-iii) Under twice-differentiability \( \nabla^2 f(x) \preceq L \nabla^2 h(x) \) for any \( x \in \text{int} \ Q \),

(a-iv) \( \langle \nabla f(x) - \nabla f(y), x - y \rangle \leq L \langle \nabla h(x) - \nabla h(y), x - y \rangle \) for all \( x, y \in \text{int} \ Q \).

The following conditions are equivalent:

(b-i) \( f(\cdot) \) is \( \mu \)-strongly convex relative to \( h(\cdot) \),

(b-ii) \( f(\cdot) - \mu h(\cdot) \) is a convex function on \( Q \),

(b-iii) Under twice-differentiability \( \nabla^2 f(x) \succeq \mu \nabla^2 h(x) \) for any \( x \in \text{int} \ Q \),

(b-iv) \( \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \mu \langle \nabla h(x) - \nabla h(y), x - y \rangle \) for all \( x, y \in \text{int} \ Q \).

The first part of Proposition 3.1.1 is almost equivalent to Proposition 1 of [8].

Proof: For \( x \in Q \) define \( \phi(x) := Lh(x) - f(x) \). Using (3.11) and (3.7) it follows that (a-i) holds if and only if \( \phi(x) \geq \phi(y) + \langle \nabla \phi(y), x - y \rangle \) for all \( x, y \in Q \), which is equivalent to the convexity of \( \phi(\cdot) = Lh(\cdot) - f(\cdot) \) from Theorem 2.1.2 of [76], thus showing that (a-i) \( \Leftrightarrow \) (a-ii). It follows from Theorem 2.1.3 of [76] applied to \( \phi(\cdot) \) that \( \phi(\cdot) \) is convex if and only if \( \langle \nabla \phi(x) - \nabla \phi(y), x - y \rangle \geq 0 \) for all \( x, y \in Q \), which shows that (a-ii) \( \Leftrightarrow \) (a-iv). If \( f(\cdot) \) and \( h(\cdot) \) are twice differentiable, then it follows from Theorem 2.1.4 of [76] that (a-ii) \( \Leftrightarrow \) (a-iii).

Similar proofs can be applied for part (b). \( \square \)

For notational convenience, let us denote by \( f(\cdot) \preceq h(\cdot) \) that \( h(\cdot) - f(\cdot) \) is a convex function, whereby this also means \( f(\cdot) \) is 1-smooth with respect to \( h(\cdot) \) from Proposition 3.1.1. Similarly \( f(\cdot) \succeq h(\cdot) \) means \( f(\cdot) - h(\cdot) \) is a convex function and so \( f(\cdot) \) is 1-strongly convex with respect to \( h(\cdot) \). (In the case when both \( f(\cdot) \) and \( h(\cdot) \) are twice differentiable, the relation “\( \cdot \succeq \cdot \)” on two functions is consistent with the Löwner partial order on the Hessians of these two functions from Proposition 3.1.1.) Then the condition that \( f(\cdot) \) is \( L \)-smooth with respect to \( h(\cdot) \) is equivalent to \( f(\cdot) \preceq Lh(\cdot) \); similarly the condition that \( f(\cdot) \) is \( \mu \)-strongly convex with respect to \( h(\cdot) \) is equivalent
to $f(\cdot) \geq \mu h(\cdot)$. In addition, relative-smoothness and relative strong convexity are each transitive, so that $f(\cdot) \leq g(\cdot)$ and $g(\cdot) \leq h(\cdot)$ implies that $f(\cdot) \leq h(\cdot)$.

We can also work with sums and linear transformations of relatively smooth and/or relatively strongly convex functions, as the next proposition states.

**Proposition 3.1.2.**

1. If $f_1(\cdot) \leq L_1 h(\cdot)$ and $f_2(\cdot) \leq L_2 h_2(\cdot)$, then for all $\alpha, \beta \geq 0$ it holds that $f(\cdot) := \alpha f_1(\cdot) + \beta f_2(\cdot) \leq h(\cdot) := \alpha L_1 h_1(\cdot) + \beta L_2 h_2(\cdot)$.

2. If $f_1(\cdot) \geq \mu_1 h_1(\cdot)$ and $f_2(\cdot) \geq \mu_2 h_2(\cdot)$, then for all $\alpha, \beta \geq 0$ it holds that $f(\cdot) := \alpha f_1(\cdot) + \beta f_2(\cdot) \geq h(\cdot) := \alpha \mu_1 h_1(\cdot) + \beta \mu_2 h_2(\cdot)$.

3. If $f(\cdot) \leq h(\cdot)$, and $A$ is a linear transformation of appropriate dimension, then $\phi_f(x) := f(Ax) \leq \phi_h(x) := h(Ax)$.

4. If $f(\cdot) \geq h(\cdot)$, and $A$ is a linear transformation of appropriate dimension, then $\phi_f(x) := f(Ax) \geq \phi_h(x) := h(Ax)$.

**Proof:** The proofs of the first two arguments follow directly from the definitions of relative smoothness and relative strong convexity in Definitions 3.1.1 and 4.3.2. The proofs of the last two arguments follow from the equivalent definition (a-iv) and (b-iv) in Proposition 3.1.1. □

### 3.1.3 Constructive Algorithmic Set-up

Let us now discuss criteria for choosing the reference function $h(\cdot)$ in the context of computational schemes for solving the optimization problem (4.1). To be concrete, consider a simple Primal Gradient Scheme as shown in Algorithm 6. Note that this scheme is essentially as described in the update formula (4.6), except that the uniform smoothness constant $L_f$ is replaced by the relative smoothness parameter $L$ of $f(\cdot)$ with respect to the reference function $h(\cdot)$ as defined in Definition 3.1.1, and the only
formal requirement for $h(\cdot)$ is that the pair $(f(\cdot), h(\cdot))$ must satisfy the conditions of Definition 3.1.1.

Algorithm 6 Primal Gradient Scheme with reference function $h(\cdot)$

**Initialize.** Initialize with $x^0 \in Q$. Let $L, h(\cdot)$ satisfying Definition 3.1.1 be given.

At iteration $i$:

**Perform Updates.** Compute $\nabla f(x^i)$,

$$x^{i+1} \leftarrow \arg \min_{x \in Q} \{ f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + LD_h(x, x^i) \}.$$

In order to efficiently execute the update step in Algorithm 6 we also require of $h(\cdot)$ that the subproblem (4.7) is efficiently solvable for any given $c$. In summary, to solve the optimization problem (4.1) using Algorithm 6, we need to specify a reference function $h(\cdot)$ that has the following two properties:

(i) $f(\cdot)$ is $L$-smooth relative to $h(\cdot)$ on $Q$, and

(ii) the subproblem (4.7) always has a solution, and the solution is efficiently computable.

In Section 3.2 we will see how this can be done for several useful classes of problems that are not otherwise solvable by traditional first-order methods that require uniform Lipschitz continuity of the gradient. In Section 4.4 we analyze the computational guarantees associated with the Primal Gradient Scheme (Algorithm 6) as well as a Dual Averaging Scheme. In Section 3.4, we apply the computational guarantees of Section 4.4 to the D-optimal design problem.

**Notation.** For a vector $x$, $X = \text{Diag}(x)$ denotes the diagonal matrix with the coefficients of $x$ along the diagonal. For a symmetric matrix $A$, $\text{diag}(A)$ denotes the vector of the diagonal coefficients of $A$, and $\text{Mdiag}(A)$ denotes the diagonal matrix whose diagonal coefficients correspond to the diagonal coefficients of $A$. Unless otherwise specified, the norm of a matrix is the operator norm using $\ell_2$ norms. The $\ell_p$ norm of a vector $x$ is denoted by $\|x\|_p$. For symmetric matrices, "\(\succeq\)" denotes the Löwner partial order. In a mild double use of notation, $f(\cdot) \succeq h(\cdot)$ denotes
\( f(\cdot) - h(\cdot) \) is a convex function, and the appropriate meaning of "\( \geq \)" will be obvious in context. Let \( e \) denote the vector of 1's whose dimension is dictated by context. Let \( \Delta_n := \{ x \in \mathbb{R}^n : \langle e, x \rangle = 1, \ x \geq 0 \} \) denote the standard unit simplex in \( \mathbb{R}^n \). Given two matrices \( A \) and \( B \) of the same order, let \( A \circ B \) denote the Hadamard (i.e., component-wise) product of \( A \) and \( B \), see for example Anstreicher [4]. Let exp denote the base of the natural logarithm.

### 3.2 Examples of Relatively Smooth Optimization Problems

Here we show several classes of optimization problems (4.1) for which one can easily construct a reference function \( h(\cdot) \) with the two properties mentioned above, namely (i) \( f(\cdot) \) is \( L \)-smooth relative to \( h(\cdot) \) for an easily determined value \( L \), and (ii) the subproblem (4.7) is efficiently solvable.

#### 3.2.1 Optimization over \( \mathbb{R}^n \) with \( \|\nabla^2 f(x)\| \) growing as a polynomial in \( \|x\|_2 \)

Suppose that \( f(\cdot) \) is a twice-differentiable convex function on \( Q := \mathbb{R}^n \) and let \( \|\nabla^2 f(x)\| \) denote the operator norm of \( \nabla^2 f(x) \) with respect to the \( \ell_2 \)-norm on \( \mathbb{R}^n \). Suppose that \( \|\nabla^2 f(x)\| \leq p_r(\|x\|_2) \), where \( p_r(\alpha) = \sum_{i=0}^{r} a_i \alpha^i \) is an \( r \)-degree polynomial of \( \alpha \). Let

\[
    h(x) := \frac{1}{r+2} \|x\|_2^{r+2} + \frac{1}{2} \|x\|_2^2 .
\]

Then the following proposition states that \( f(\cdot) \) is \( L \)-smooth relative to \( h(\cdot) \) for an easily computable value \( L \). This implies that no matter how fast the Hessian of \( f(\cdot) \) grows as \( \|x\|_2 \to \infty \), \( f(\cdot) \) can still be smooth relative to the simple reference function \( h(\cdot) \), even though \( \nabla f(\cdot) \) need not exhibit uniform Lipschitz continuity.
Proposition 3.2.1. Suppose \( f(\cdot) \) is twice differentiable and satisfies \( \|\nabla^2 f(x)\| \leq p_r(\|x\|_2) \) where \( p_r(\alpha) \) is an \( r \)-degree polynomial of \( \alpha \). Let \( L \) be such that \( p_r(\alpha) \leq L(1 + \alpha^r) \) for \( \alpha \geq 0 \). Then \( f(\cdot) \) is \( L \)-smooth relative to \( h(x) = \frac{1}{r+2}\|x\|_2^{r+2} + \frac{1}{2}\|x\|_2^2 \).

Proof: It follows from elementary rules of differentiation that

\[
\nabla^2 h(x) = (1 + \|x\|_2^2)I + (r+1)\|x\|_2^{r-2}xx^T \geq (1 + \|x\|_2^2)I \geq \frac{1}{L}p_r(\|x\|_2)I \geq \frac{1}{L}\nabla^2 f(x),
\]

and so \( f(\cdot) \) is \( L \)-smooth relative to \( h(\cdot) \) by part (iii) of Proposition 3.1.1. \( \square \)

Utilizing the additivity property in Proposition 4.3.2 together with Proposition 3.2.1, one concludes that virtually every twice-differentiable convex function on \( \mathbb{R}^n \) is \( L \)-smooth relative to some simple polynomial function of \( \|x\|_2 \).

Remark 3.2.1. Suppose \( p_r(\alpha) = \sum_{i=0}^r a_i \alpha^i \). In Proposition 3.2.1, one simple way to set \( L \) is to use \( L = \sum_{i=0}^r |a_i| \). Then

\[
p_r(\alpha) \leq \begin{cases} 
\sum_{i=0}^r |a_i| & \text{for } 0 \leq \alpha \leq 1 \\
\sum_{i=0}^r |a_i|\alpha^r & \text{for } \alpha \geq 1 
\end{cases} \tag{3.14}
\]

whereby \( p_r(\alpha) \leq \max\{L, L\alpha^r\} \leq L(1 + \alpha^r) \) for \( \alpha \geq 0 \).

Solving the subproblem (4.7). Let us see how we can solve the subproblem (4.7) for this class of optimization problems. The subproblem (4.7) can be written as

\[
\min_{x \in \mathbb{R}^n} \langle c, x \rangle + \frac{1}{r+2}\|x\|_2^{r+2} + \frac{1}{2}\|x\|_2^2, \tag{3.15}
\]

and the first-order optimality conditions are simply:

\[
c + (1 + \|x\|_2^r)x = 0,
\]

whereby \( x = -\theta c \) for some \( \theta \geq 0 \), and it remains to simply determine the value of the nonnegative scalar \( \theta \). If \( c = 0 \), then \( x = 0 \) satisfies the optimality conditions. For
$c \neq 0$, notice from above that $\theta$ must satisfy:

$$1 - \theta - \|c\|^2 \cdot \theta^{r+1} = 0,$$

which is a univariate polynomial in $\theta$ with a unique positive root. For $r = 1, 2, 3$, this root can be computed in closed form. Otherwise, the root can be computed (up to machine precision) using any scalar root-finding method.

**Remark 3.2.2.** We can incorporate in problem (4.23) a simple set constraint $x \in Q$ provided that we can easily compute the Euclidean projection on $Q$. In the case when $h(\cdot)$ is a convex function of $\|x\|^2_2$, the subproblem (4.7) can be converted to a 1-dimensional convex optimization problem, see Appendix B.1 for details.

**A more specific example.** Let $f(x) := \frac{1}{4}\|Ax - b\|^4_2 + \frac{1}{2}\|Cx - d\|^2_2$. Then $\nabla^2 f(x) = 3A^T D^2(x)A + C^T C$, where $D(x) = \text{Diag}(Ax - b)$. Let us show that $f(x)$ is $L$-smooth relative to

$$h(x) := \frac{1}{4}\|x\|^4_2 + \frac{1}{2}\|x\|^2_2$$

on $Q = \mathbb{R}^n$ for $L = 3\|A\|^4_4 + 6\|A\|^3_3 ||b||_2 + 3\|A\|^2_2 ||b||^2_2 + ||C||^2_2$. To see this, notice first that:

$$\|\nabla^2 f(x)\| \leq 3\|A\|^2 (||b||_2 + \|A\||x||_2^2)^2 + ||C||^2 = (3\|A\|^2 ||b||^2_2 + ||C||^2) + 6\|A\|^3 ||b||_2 ||x||_2 + 3\|A\|^4 ||x||^2_2,$$

which is 2-degree polynomial in $||x||_2$ with coefficients $a_0 = 3\|A||^2 ||b||^2_2 + ||C||^2$, $a_1 = 6\|A||^3 ||b||_2$, and $a_2 = 3\|A||^4$. Therefore following Remark 3.2.1 it suffices to set

$$L = \sum_{i=0}^2 a_i = 3\|A||^4 + 6\|A||^3 ||b||_2 + 3\|A||^2 ||b||^2_2 + ||C||^2.$$

**An example with Non-Lipschitz $\mu$-strong convexity.** Let $f(x) := \frac{1}{4}||Ex||^4_2 +$
\[ \frac{1}{4}\|Ax - b\|^4_4 + \frac{1}{2}\|Cx - d\|^2_2, \] and let \( \sigma_E \) and \( \sigma_C \) denote the smallest singular values of \( E \) and \( C \), respectively, and let us suppose that \( \sigma_E > 0 \) and \( \sigma_C > 0 \). Then \( \nabla^2 f(x) = \|Ex\|_2^2ETE + 2ETExx^TE + 3ATD^2(x)A + CT^TC \), where \( D(x) = \text{Diag}(Ax - b) \). Let us show that \( f(x) \) is \( L \)-smooth and \( \mu \)-strongly convex relative to

\[ h(x) := \frac{1}{4}\|x\|^4_2 + \frac{1}{2}\|x\|^2_2 \]

on \( Q = \mathbb{R}^n \) for \( L = 3\|E\|^4 + 3\|A\|^4 + 6\|A\|^3\|b\|_2 + 3\|A\|^2\|b\|^2_2 + \|C\|^2 \) and \( \mu = \min\{\sigma_E^2, \sigma_C^2\} \). Similar to what we have above,

\[
\|\nabla^2 f(x)\| \leq \|E\|^4\|x\|^2_2 + 2\|E\|^4\|x\|^2_2 + 3\|A\|^2(\|b\|_2 + \|A\||x\|_2)^2 + \|C\|^2 \]

\[ = (3\|A\|^2\|b\|^2_2 + \|C\|^2) + 6\|A\|^3\|b\|_2\|x\|_2 + (3\|E\|^4 + 3\|A\|^4)\|x\|^2_2, \]

which is 2-degree polynomial in \( \|x\|_2 \) with coefficients \( a_0 = 3\|A\|^2\|b\|^2_2 + \|C\|^2 \), \( a_1 = 6\|A\|^3\|b\|_2 \), and \( a_2 = 3\|E\|^4 + 3\|A\|^4 \). Therefore following Remark 3.2.1 it suffices to set

\[ L = \sum_{i=0}^{2} a_i = 3\|E\|^4 + 3\|A\|^4 + 6\|A\|^3\|b\|_2 + 3\|A\|^2\|b\|^2_2 + \|C\|^2. \]

On the other hand,

\[
\nabla^2 f(x) \succeq \|Ex\|_2^2ETE + C^TC \succeq \sigma_E^2\|x\|^2_2I \succeq \sigma_C^2\|x\|^2_2I \]

\[ \succeq \mu \left(1 + 3\|x\|^2_2\right)I \succeq \mu \left((1 + \|x\|^2_2)I + 2xx^T\right) = \mu \nabla^2 h(x) \]

(where the last matrix inequality follows since \( \|x\|^2_2I \succeq xx^T \)), and thus \( f(x) \) is \( \mu \)-strongly convex relative to \( h(x) \).

**Remark 3.2.3.** In place of the simple reference function \( h(\cdot) \) in (3.13) one can instead consider a “re-centered” version of the form:

\[ h(x) = h_{\epsilon}(x) := \frac{1}{r+2}\|x - x^\epsilon\|^2_2 + \frac{1}{2}\|x - x^\epsilon\|^2_2, \]

75
where the “center” value \( x^c \) is suitably chosen to align \( f(\cdot) \) with \( h(\cdot) \) and possibly attain better values of \( L \) and \( \mu \). Note that introducing the given center value \( x^c \) does not increase the difficulty of solving the subproblem (4.7). We illustrate this idea with a simple univariate example. Suppose that our objective function is \( f(x) = x^4 - 4x^3 + 7x^2 - 5x + 3 \). From the results in Section 3.2.1 we know we can use the reference function \( h_1(x) := \frac{1}{4}x^4 + \frac{1}{2}x^2 \). We can also translate \( x \) by the center point \( x^c := 1 \) and use the reference function \( h_2(x) := \frac{1}{4}(x - 1)^4 + \frac{1}{2}(x - 1)^2 \). Straightforward calculation yields values of \( L = L_1 = 9 + \sqrt{73} \approx 17.5440 \) for \( h_1(\cdot) \) and \( L = L_2 = 4 \) for \( h_2(\cdot) \), whereby \( h_2(\cdot) \) yields a better value of \( L \) than \( h_1(\cdot) \) for this example.

### 3.2.2 D-Optimal Design Problem

Given a matrix \( H \in \mathbb{R}^{m \times n} \) of rank \( m \) where \( n \geq m + 1 \), the D-optimal design problem is:

\[
D: \quad f^* = \min_x f(x) := -\ln \det (HXHT)
\]

\[
\text{s.t.} \quad (e, x) = 1
\]

\[
x \geq 0 ,
\]

where recall \( X := \text{Diag}(x) \). In statistics, the D-optimal design problem corresponds to maximizing the determinant of the Fisher information matrix \( \mathbb{E}(hh^T) \), see [55], [6]. And in computational geometry, D-optimal design arises as a Lagrangian dual problem of the minimum volume covering ellipsoid (MVCE) problem, which dates back at least 60 years to [49], see Todd [103] for a modern treatment. Indeed, (3.16) is useful in a variety of different application areas, for example, computational statistics [29] and data mining [56]. In terms of algorithms for solving (3.16), Khachiyan and Todd [54] proposed a theory-oriented scheme based on interior-point methods, see also Zhang [117] as well as [100] for more practical treatments using interior-point methods. Khachiyan [53] later proposed and analyzed a first-order method (equivalent to the Frank-Wolfe method) to solve (3.16), which led to other works along this line including Yildirim [112] and Ahipasaoglu, Sun, and Todd [1]. The complexity analysis
in these papers is very specialized for the $D$-optimal design problem. In contrast, we will show how the Primal Gradient Scheme (Algorithm 6) can be applied to the $D$-optimal design problem; furthermore, in Section 3.4 we will apply the complexity analysis of Section 4.4 for the Primal Gradient Scheme to the set-up of $D$-optimal design, along with a comparison of our convergence guarantees with the guarantees from prior literature.

Notice that (3.16) is an instance of (4.1) with $Q = \Delta_n := \{x \in \mathbb{R}^n : \langle e, x \rangle = 1, \; x \geq 0\}$. Although strictly speaking, $f(\cdot)$ in (3.16) is not defined everywhere on the relative boundary of $Q$ and hence does not have gradients or Hessians everywhere on the relative boundary of $Q$, this will not be of concern. For $f(\cdot)$ in (3.16) let us choose the reference function $h(\cdot)$ to be the logarithmic barrier function, namely

$$h(x) := -\sum_{j=1}^{n} \ln(x_j),$$

which is defined on the positive orthant $\mathbb{R}_{++}^n$. The following proposition states that $f(\cdot)$ is 1-smooth relative to $h(\cdot)$.

**Proposition 3.2.2.** Suppose $f(x) = -\ln \det (HXHT)$, where $X = \text{Diag}(x)$. Then $f(\cdot)$ is 1-smooth relative to $h(x) = -\sum_{j=1}^{n} \ln(x_j)$ on $\mathbb{R}_{++}^n$.

**Proof:** The gradient of $f(\cdot)$ is $\nabla f(x) = \text{diag}(-C)$ and the Hessian of $f(\cdot)$ is $\nabla^2 f(x) = C \circ C$, where $C := HT(HXHT)^{-1}H$. Let $U =HX^{\frac{1}{2}}$; then $U^T(UU^T)^{-1}U \preceq I$ since the left side of this matrix inequality is a projection operator, whereby $X^{\frac{1}{2}}HT(HXHT)^{-1}HX^{\frac{1}{2}} \preceq I$. Multiplying this matrix inequality on the left and right by $X^{-\frac{1}{2}}$ then shows that $C \preceq X^{-1}$. Therefore,

$$\nabla^2 f(x) = C \circ C \preceq C \circ X^{-1} \preceq X^{-1} \circ X^{-1} = X^{-2} = \nabla^2 h(x),$$

(3.17)

where the first and the second matrix inequality above each follows from the fact that $C \preceq X^{-1}$ and the Hadamard product of two symmetric positive semidefinite matrices is also a symmetric positive semidefinite matrix. The result then follows.
Solving the subproblem (4.7). Let us see how we can solve the subproblem (4.7) for $Q$ and $h(\cdot)$ given above. The subproblem (4.7) can be written as

$$\min_{x \in \Delta_n} \langle c, x \rangle - \sum_{j=1}^{n} \ln(x_j),$$

and the first-order optimality conditions are simply:

$$x > 0, \quad \langle e, x \rangle = 1, \quad \text{and } c - X^{-1}e = -\theta e$$

for some scalar multiplier $\theta$. Given $\theta$, it then follows that $x_j = 1/(c_j + \theta)$ for $j = 1, \ldots, n$, and it remains to simply determine the value of the scalar $\theta$. Now notice that $\theta$ must satisfy:

$$d(\theta) := \sum_{j=1}^{n} \frac{1}{c_j + \theta} - 1 = 0$$

(3.18)

for some $\theta$ in the interval $\mathcal{F} := (-\min_j\{c_j\}, \infty)$. Notice that $d(\cdot)$ is strictly decreasing on $\mathcal{F}$, and $d(\theta) \to +\infty$ as $\theta \searrow \min_j\{c_j\}$ and $d(\theta) \to -1$ as $\theta \to \infty$, whereby (3.18) has a unique solution in $\mathcal{F}$. Furthermore, as suggested by results in Ye [111] or [34], one can use Newton’s method (or any other suitable scalar solution-finding method) to efficiently compute the solution of (3.18) (up to machine precision) on the interval $\mathcal{F}$.

3.2.3 Generalized Volumetric Function Optimization

For a given integer parameter $p > 0$, let us also study optimization on the simplex of the following generalization of the volumetric barrier function:

$$\min_x f_p(x) = \ln \det (HX^{-p}H^T)$$

s.t. $\langle e, x \rangle = 1$

$$x \geq 0,$$ 

(3.19)
where the integer $p$ is the parameter of the volumetric function $f_p(\cdot)$, and $H \in \mathbb{R}^{m \times n}$ is a rank-$m$ matrix where $n \geq m + 1$. Here the feasible region is $Q = \Delta_n$. Note that $f_p(\cdot)$ is a convex function when $p \geq 0$ (and $f_p(\cdot)$ is a concave function when $p = -1$).

Similar to the $D$-optimal design problem, $f_p(\cdot)$ is not defined everywhere on the boundary of $\mathbb{R}_n^+$, but this will not be a concern. The reference function $h(\cdot)$ we choose is the logarithmic barrier function, namely

$$h(x) := -\sum_{j=1}^n \ln(x_j),$$

which is defined on $\mathbb{R}_+^n$. The following proposition states that $f_p(\cdot)$ is $p(p+1)$-smooth relative to $h(\cdot)$.

**Proposition 3.2.3.** $f_p(\cdot)$ is $p(p+1)$-smooth relative to $h(x) = -\sum_{j=1}^n \ln(x_j)$ on $\mathbb{R}_+^n$.

**Proof:** By elementary calculus, the gradient of $f_p(\cdot)$ is

$$\nabla f_p(x) = -p \cdot \text{diag} \left( X^{-1/2-p/2} CX^{-1/2-p/2} \right),$$

and the Hessian of $f_p(\cdot)$ is

$$\nabla^2 f_p(x) = p(p+1) M \text{diag} \left( X^{-1-p/2} CX^{-1-p/2} \right) - p^2 X^{-1-p/2} (C \circ C) X^{-1-p/2},$$

where $C := H^T (H X^{-p} H^T)^{-1} H$, and $M \text{diag}(M)$ denotes the diagonal matrix whose entries are the diagonal components of the matrix $M$. Let $U = H X^{-p/2}$; then $U^T U U^T \succeq I$ since the left side of this matrix inequality is a projection operator. Therefore each diagonal component of $U^T U U^T \succeq I$ does not exceed 1, whereby we
have \( \mathbf{M}_{\text{diag}} (U^T (UU^T)^{-1} U) \preceq I \). Therefore,

\[
\nabla^2 f_p(x) \preceq p(p+1) \mathbf{M}_{\text{diag}} \left( X^{-1-p/2} C X^{-1-p/2} \right)
\]

\[
= p(p+1) X^{-1} \mathbf{M}_{\text{diag}} \left( U^T (UU^T)^{-1} U \right) X^{-1}
\]

\[
\preceq p(p+1) X^{-2}
\]

\[
= p(p+1) \nabla^2 h(x)
\]

where the first inequality follows from the fact that the Hadamard product of two symmetric positive semidefinite matrices is also a symmetric positive semidefinite matrix and \( C \) is a positive semidefinite matrix, and the first equation follows since \( X \) is itself a diagonal matrix. The result then follows by property (iii) of Proposition 3.1.1.

\[ \square \]

**Solving the subproblem** (4.7). Using \( h(x) = -\sum_{j=1}^n \ln(x_j) \), the subproblem (4.7) here is identical to that for the \( D \)-optimal design problem, since the reference function \( h(\cdot) \) and the feasible domain \( Q \) are the same. Therefore the methodology discussed in Section 3.2.2 applies here as well.

**Remark.** By setting \( H = A^T \) and using Proposition 4.3.2, it can also be shown that \( \hat{f}(x) := \ln \det \left( A^T \text{Diag} (Ax - b)^{-p} A \right) \) is \( p(p+1) \)-smooth relative to \( h(x) := -\sum_i \ln(A_i x - b_i) \). When \( p = 2 \) this is the volumetric barrier function on the set \( Q = \{ x \in \mathbb{R}^n : Ax \geq b \} \), see [106], [5].

### 3.2.4 Optimization over \( Q \subset (0,u)^n \) with \( \| \nabla^2 f(x) \| \) growing as a polynomial in \( \sum_{i=1}^n \frac{1}{x_i} \)

Suppose that \( f(\cdot) \) is a twice-differentiable convex function on \( Q \subset (0,u)^n \) and that

\[ \| \nabla^2 f(x) \| \leq q_s \left( \sum_{i=1}^n \frac{1}{x_i} \right) \]

where \( q_s(\alpha) = \sum_{i=0}^s a_i \alpha^i \) is an \( s \)-degree polynomial in \( \alpha \).
(Recall $\|\nabla^2 f(x)\|$ denotes the operator norm of $\nabla^2 f(x)$ with respect to the $\ell_2$-norm on $\mathbb{R}^n$.) Let

$$h(x) := \frac{u^3}{2(s+1)} \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s+1}.$$ 

Then the following proposition states that $f(\cdot)$ is $L$-smooth relative to $h(\cdot)$ for an easily computable value $L$. This implies that no matter how fast $\nabla f(x)$ grows as $x$ approaches the open boundary of the region $(0,u)^n$, $f(\cdot)$ is smooth relative to the simple reference function $h(\cdot)$, even though $\nabla f(\cdot)$ need not exhibit uniform Lipschitz continuity on $Q$.

**Proposition 3.2.4.** Suppose $f(\cdot)$ is twice differentiable on $Q$ and satisfies $\|\nabla^2 f(x)\| \leq q_s \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)$ where $q_s(\alpha)$ is an $s$-degree polynomial in $\alpha$. Let $L$ be such that $q_s(\alpha) \leq L\alpha^s$ for all $\alpha \geq \frac{n}{u}$. Then $f(\cdot)$ is $L$-smooth relative to $h(x) = \frac{u^3}{2(s+1)} \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s+1}$.

**Proof:** Let $X := \text{Diag}(x)$, and it follows from elementary rules of differentiation that

$$\nabla^2 h(x) = u^3 \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s} X^{-3} + \frac{u^3 s}{2} \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s-1} X^{-2} e e^T X^{-2} .$$

(3.20)

Therefore

$$\nabla^2 h(x) \succeq u^3 \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s} X^{-3} \succeq \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s} I \succeq \frac{1}{L} q_s \left( \sum_{i=1}^{n} \frac{1}{x_i} \right) (3.21)$$

where the second matrix inequality uses $u \geq x_i$ and the third matrix inequality is due to $\sum_{i=1}^{n} \frac{1}{x_i} \geq \sum_{i=1}^{n} \frac{1}{u} = \frac{n}{u}$. Therefore $f(\cdot)$ is $L$-smooth relative to $h(\cdot)$ by part (iii) of Proposition 3.1.1.

**Remark 3.2.4.** Suppose $q_s(\alpha) = \sum_{i=0}^{s} a_i \alpha^i$. In Proposition 3.2.4, one simple way to set $L$ is to use $L = \sum_{i=0}^{s} |a_i| \left( \frac{u}{n} \right)^{i-s}$. This implies for $\alpha \geq \frac{n}{u}$ that

$$q_s(\alpha) \leq \sum_{i=0}^{s} |a_i| \alpha^i \leq \left( \sum_{i=0}^{s} |a_i| \left( \frac{u}{n} \right)^{i-s} \right) \alpha^s = L\alpha^s .$$

(3.22)
Solving the subproblem (4.7). Let us see how we can solve the subproblem (4.7) for this class of optimization problems. After rescaling $c$ by $u^3/2$, the subproblem (4.7) can be equivalently written as

$$
\min_{x \in (0,u]^n} \langle c, x \rangle + \frac{1}{s+1} \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^{s+1}.
$$

Let $\theta = \left( \sum_{i=1}^{n} \frac{1}{x_i} \right)^s$, then the optimality conditions for (3.23) can be written as:

$$
x_i = \begin{cases} 
    u & \text{if } c_i \leq \frac{\theta}{u^2} \\
    \sqrt[2]{\frac{\theta}{c_i}} & \text{for } c_i > \frac{\theta}{u^2},
\end{cases}
$$

for $i = 1, \ldots, n$. For a given $\theta > 0$, define $x_i(\theta)$ using the above rule (3.24), and it remains to simply determine the value of the positive scalar $\theta$ in the interval $\mathcal{F} := \left[ \left( \frac{n}{u} \right)^s, \infty \right)$ that satisfies

$$
d(\theta) := \theta - \left( \sum_{i=1}^{n} \frac{1}{x_i(\theta)} \right)^s = 0.
$$

Notice that $d(\cdot)$ is strictly increasing on $\mathcal{F}$, and $d \left( \left( \frac{n}{u} \right)^s \right) \leq 0$ (since $x_i(\theta) \leq u$ for any $\theta$) and $d(\theta) \to \infty$ as $\theta \to \infty$. Therefore (3.25) has a unique solution in $\mathcal{F}$, which can be solved with high accuracy using any suitable root-finding method, for example binary search combined with 1-dimensional Newton’s method.

Remark 3.2.5. In a sense, there are basically two ways that a twice-differentiable convex function can fail to have a uniformly Lipschitz gradient: (i) when the Hessian grows without limit as $\|x\| \to \infty$, and/or (ii) when the Hessian grows without limit as $x \to x^0 \in \partial Q$. Section 3.2.1 has provided a mechanism for constructing a reference function $h(\cdot)$ for case (i) when the growth is polynomial, and Section 3.2.4 has provided such a mechanism for case (ii) when the growth is polynomial. By utilizing the additivity and linear transformation properties of relative smoothness in Proposition 82.
4.3.2, it should be possible to construct suitable reference functions for many convex functions of interest.

3.3 Computational Analysis for the Primal Gradient Scheme and the Dual Averaging Scheme

In this section we present computational guarantees for two algorithms: the Primal Gradient Scheme (Algorithm 6) as well as a Dual Averaging Scheme (Algorithm 7).

3.3.1 Analysis of Primal Gradient Scheme (Algorithm 6)

Our main result for the Primal Gradient Scheme is the following sublinear and linear convergence bounds.

**Theorem 3.3.1.** Consider the Primal Gradient Scheme (Algorithm 6). If \( f(\cdot) \) is \( L \)-smooth and \( \mu \)-strongly convex relative to \( h(\cdot) \) for some \( L > 0 \) and \( \mu \geq 0 \), then for all \( k \geq 1 \) and \( x \in Q \), sequence \( \{f(x^k)\} \) is monotonically decreasing, and the following inequality holds:

\[
    f(x^k) - f(x) \leq \frac{\mu D_h(x, x^0)}{\left(1 + \frac{\mu}{L-\mu}\right)^k - 1} \leq \frac{L - \mu}{k} D_h(x, x^0),
\]

where, in the case when \( \mu = 0 \), the middle expression is defined in the limit as \( \mu \to 0^+ \).

The first inequality in (4.4.3) shows linear convergence when \( \mu > 0 \); indeed, in this case it holds that

\[
    \frac{\mu D_h(x, x^0)}{\left(1 + \frac{\mu}{L-\mu}\right)^k - 1} \leq L \left(1 - \frac{\mu}{L}\right)^k D_h(x, x^0).
\]
(This inequality holds trivially for \( k = 1 \), and induction on \( k \) establishes the result for \( k \geq 2 \).) Furthermore, when \( k \) is large the \(-1\) term in the denominator of the left-hand side can be ignored which yields the asymptotic bound \( \mu \left(1 - \frac{k}{k}\right)^k D_h(x, x^0) \).

The second inequality in (4.4.3) shows an \( O(1/k) \) sublinear convergence rate. In particular, the convergence rate in (4.4.3) is \( \frac{1}{k} D_h(x, x^0) \) when \( \mu = 0 \).

Note that Algorithm 6 herein and the NoLips algorithm in [8] as well as algorithm PGA-B in [118] are structurally identical (they are all instantiations of the primal gradient methodology). However, the step-size rule in [8] as well as the complexity analysis in [8] depends on a symmetry measure of \( D_h(\cdot, \cdot) \), namely \( \alpha := \min_{x,y \neq x} D_h(x, y)/D_h(y, x) \), whereas there is no such dependence here. The instantiation of Algorithm 6 in [8] uses a smaller “step-size” of \( (1 + \alpha)/2L \) as opposed to \( 1/L \) in the update computation in Algorithm 6 (since it must always hold that \( \alpha \leq 1 \)), and [8] proves a computational guarantee of \( f(x^k) - f(x) \leq \frac{2L}{(1 + \alpha)k} D_h(x, x^0) \).

The bound in Theorem 3.3.1 is better than this symmetry-based bound, but only by a multiplicative constant factor \( (1 + \alpha)/2 \) when \( \mu = 0 \); it is of course far better (linear convergence rather than sublinear convergence) when \( \mu > 0 \).

The proof of the bound in Theorem 3.3.1 relies on the following standard Three-Point Property:

**Lemma 3.3.1. (Three-Point Property of Tseng [105])** Let \( \phi(x) \) be a convex function, and let \( D_h(\cdot, \cdot) \) be the Bregman distance for \( h(\cdot) \). For a given vector \( z \), let

\[
\mathbf{z^+} := \arg \min_{x \in Q} \{ \phi(x) + D_h(x, z) \} .
\]

Then

\[
\phi(x) + D_h(x, z) \geq \phi(z^+) + D_h(z^+, z) + D_h(x, z^+) \quad \text{for all } x \in Q . \quad \Box
\]
Proof of Theorem 3.3.1: Define a parameter sequence

\[
C_k := \frac{1}{\sum_{i=1}^{k} \left( \frac{L}{L-\mu} \right)^i} = \frac{\mu}{L \left( 1 + \frac{\mu}{L-\mu} \right)^k - 1},
\]

where the second equality "(\cdot)" follows from elementary geometric series’ analysis, and holds only when \(\mu > 0\). In particular, \(C_k = \frac{1}{k}\) if \(\mu = 0\). For any \(x \in Q\) and \(i \geq 1\) we have:

\[
f(x^i) \leq f(x^{i-1}) + \langle \nabla f(x^{i-1}), x^i - x^{i-1} \rangle + LD_h(x^i, x^{i-1})
\]

\[
\leq f(x^{i-1}) + \langle \nabla f(x^{i-1}), x - x^{i-1} \rangle + LD_h(x, x^{i-1}) - LD_h(x, x^i) \tag{3.28}
\]

\[
\leq f(x) + (L - \mu)D_h(x, x^{i-1}) - LD_h(x, x^i),
\]

where the first inequality follows from the definition of \(L\)-smoothness relative to \(h(\cdot)\), the second inequality is due to the Three-Point Property with \(\phi(x) = \frac{1}{L} \langle \nabla f(x^{i-1}), x - x^{i-1} \rangle\) and \(z = x^{i-1}, z^+ = x^i\), and the last inequality uses the \(\mu\)-strong convexity of \(f(\cdot)\) relative to \(h(\cdot)\), which implies \(\langle \nabla f(x^{i-1}), x - x^{i-1} \rangle \leq f(x) - f(x^{i-1}) - \mu D_h(x, x^{i-1})\).

Substituting \(x = x^{i-1}\) in (3.28) shows in particular that \(f(x^i) \leq f(x^{i-1})\) which proves monotonicity of the sequence \(\{f(x^i)\}\).

It then follows using induction and (3.28) that

\[
\sum_{i=1}^{k} \left( \frac{L}{L-\mu} \right)^i f(x^i) \leq \sum_{i=1}^{k} \left( \frac{L}{L-\mu} \right)^i f(x) + LD_h(x, x^0) - \left( \frac{L}{L-\mu} \right)^k LD_h(x, x^k). \tag{3.29}
\]

Using the monotonicity of \(f(x^i)\) and the nonnegativity of \(D_h(x, x^k)\), this implies that

\[
\left( \sum_{i=1}^{k} \left( \frac{L}{L-\mu} \right)^i \right) (f(x^k) - f(x)) \leq LD_h(x, x^0) - \left( \frac{L}{L-\mu} \right)^k LD_h(x, x^k) \leq LD_h(x, x^0). \tag{3.30}
\]

85
By substituting in the equality

$$\sum_{i=1}^{k} \left( \frac{L}{L-\mu} \right)^i = \frac{1}{C_k}$$

in (3.30) and rearranging, we obtain

$$f(x^k) - f(x) \leq C_kLDh(x, x^0) = \frac{\mu D_h(x, x^0)}{(1 + \frac{\mu}{L-\mu})^k - 1}. \quad (3.31)$$

The proof of the second inequality in (4.4.3) follows by noting that \((1 + \frac{\mu}{L-\mu})^k \geq 1 + \frac{k\mu}{L-\mu} \). \hfill \square

### 3.3.2 Dual Averaging Scheme and Analysis

Another algorithm for solving our optimization problem (4.1) is the Dual Averaging Scheme [73], which we present here in Algorithm 7. Somewhat akin to the Primal Gradient Scheme, the update step in the Dual Averaging Scheme also requires the solution of a subproblem exactly of the form (4.7). Notice that we need the coefficient \(\mu\) of strong convexity in order to implement Algorithm 7, in contrast to the Primal Gradient Scheme (Algorithm 6). One can always conservatively set \(\mu \leftarrow 0\) in Algorithm 7 if no reasonable lower bound on best value of \(\mu\) is known.

**Algorithm 7** Dual Averaging Scheme with reference function \(h(\cdot)\)

**Initialize.** Let \(L, \mu\) and \(h(\cdot)\) satisfying Definitions 3.1.1 and 4.3.2 be given.

Let \(x^0\) be the "\(h(\cdot)\)-center" of \(Q\), namely \(x^0 \leftarrow \arg\min_{x \in Q} \{h(x)\}\), satisfying \(h(x^0) = 0\).

At iteration \(k\):

**Perform Updates.** Compute \(f(x^k), \nabla f(x^k), a_{k+1} = \frac{1}{L-\mu} \left( \frac{L}{L-\mu} \right)^k\), and

$$x^{k+1} \leftarrow \arg\min_{x \in Q} \left\{ h(x) + \sum_{i=0}^{k} a_{i+1} (f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + \mu D_h(x, x^i)) \right\}.$$
Averaging Scheme.

**Theorem 3.3.2.** Consider the Dual Averaging Scheme (Algorithm 7). If \( f(\cdot) \) is \( L \)-smooth and \( \mu \)-strongly convex relative to \( h(\cdot) \) with \( L > \mu \), then for all \( k \geq 1 \) and \( x \in Q \), the following inequality holds:

\[
\min_{i=1,...,k} \{ f(x^i) \} - f(x) \leq \frac{\mu h(x)}{(1 + \frac{\mu}{L-\mu})^k - 1} \leq \frac{L - \mu}{k} h(x) ,
\]

(3.32)

where in the case \( \mu = 0 \), the middle expression is defined as the limits as \( \mu \to 0^+ \).

Similar to the result in Theorem 3.3.1, the first inequality in (3.32) shows linear convergence when \( \mu > 0 \), since

\[
\frac{\mu h(x)}{(1 + \frac{\mu}{L-\mu})^k - 1} \leq L \left( 1 - \frac{\mu}{L} \right)^k h(x) ;
\]

(3.33)

this follows using identical logic as in (3.27).

**Proof of Theorem 3.3.2:** Define \( \psi_k(x) := h(x) + \sum_{i=0}^{k-1} \left( f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + \mu D_h(x, x^i) \right) \) for \( k \geq 0 \) and \( \psi^*_k := \min_{x \in Q} \psi_k(x) \), whereby \( x^k = \arg \min_{x \in Q} \psi_k(x) \) and \( \psi_k(x^k) = \psi^*_k \).

It follows from the definition of relative strongly convexity (Definition 4.3.2) that for any \( x \in Q \):

\[
\psi^*_k \leq h(x) + A_k f(x) ,
\]

(3.34)

where

\[
A_k := \sum_{i=0}^{k-1} a_{i+1} \left[ \frac{1}{\mu} \left( 1 + \frac{\mu}{L-\mu} \right)^k - 1 \right]
\]

for all \( k \geq 0 \), and where the second equality "\((\cdot)\)" above follows from elementary geometric series’ analysis and holds only when \( \mu > 0 \); note that \( A_k = \frac{k}{L} \) when \( \mu = 0 \).

The function \( \psi_k(\cdot) \) is a sum of a linear function and the reference function \( h(\cdot) \) multiplied by the coefficient \( 1 + \mu A_k \). Therefore \((1 + \mu A_k)h(\cdot) \) and \( \psi_k(\cdot) \) define the
same Bregman distance, whereby for any $x \in Q$ it holds that:

$$(1 + \mu A_k)D_h(x, x^k) = D_{\psi_k}(x, x^k) = \psi_k(x) - \psi_k(x^k) - \langle \nabla \psi_k(x^k), x - x^k \rangle \leq \psi_k(x) - \psi_k^*, \tag{3.35}$$

where the last inequality utilizes $\psi_k(x^k) = \psi_k^*$ as well as the first order optimality condition of $x^k = \arg \min_{x \in Q} \psi_k(x)$. Therefore:

$$\psi_{k+1}^* = \psi_{k+1}(x^{k+1})$$

$$= \psi_k(x^{k+1}) + a_{k+1} \left( f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \mu D_h(x^{k+1}, x^k) \right)$$

$$\geq \psi_k^* + a_{k+1} \left( f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \left( \mu + \frac{1}{a_{k+1}} (1 + \mu A_k) \right) D_h(x^{k+1}, x^k) \right),$$

where the last inequality uses (3.35) with $x = x^{k+1}$. Taking into account that $\mu + \frac{1}{a_{k+1}} (1 + \mu A_k) = \frac{1 + \mu A_{k+1}}{a_{k+1}} = \frac{1}{a_{k+1}} \left( \frac{L}{L - \mu} \right)^{k+1} = L$, and using the relative smoothness of $f(\cdot)$ (Definition 3.1.1), we obtain:

$$\psi_{k+1}^* \geq \psi_k^* + a_{k+1} f(x^{k+1}).$$

It then follows by induction that:

$$\sum_{i=0}^{k-1} a_{i+1} f(x^{i+1}) \leq \psi_k^* \leq h(x) + A_k f(x), \tag{3.36}$$

where the second inequality is from (3.34). The proof is completed by rearranging (3.36) and taking the minimum over $i$. \qed

88
3.3.3 On Optimization Problems with a Composite Function

Sometimes we are interested in solving the *composite* optimization problem [80]:

\[
P: \quad f^* := \text{minimum}_x \ f(x) + P(x)
\]

(3.37)

s.t. \( x \in Q \),

under the same assumptions on \( f(\cdot) \) and \( Q \) as in (4.1), but now the objective function includes another function \( P(\cdot) \) that is assumed to be convex but not necessarily differentiable, and for which the following subproblem is efficiently solvable:

\[
x_{\text{new}} \leftarrow \arg \min_{x \in Q} \{ (c, x) + P(x) + h(x) \},
\]

(3.38)

for any given \( c \). Under this assumption it is straightforward to show that Algorithm 6 naturally extends to cover the case of the composite optimization problem (3.37) (see [8] and [118]) and that the computational guarantee in Theorem 3.3.1 extends to composite optimization as well. (Indeed, when \( \mu = 0 \) this extension is implied in principle from [118].) It turns out that one can actually view composite optimization as working with the objective function \( \tilde{f}(\cdot) \) that is 1-smooth relative to the reference function \( \tilde{h}(\cdot) := Lh(\cdot) + P(\cdot) \). However, the definition of the reference function \( h(\cdot) \) has been premised on \( h(\cdot) \) being differentiable on \( Q \), which might not hold for \( \tilde{h}(\cdot) \) as just defined. This can all be taken care of by a suitable modification of the theory, see Appendix B.2 for details.

3.3.4 Questions: Accelerated Methods, Conjugate Duality, Choosing the Reference Function

We have shown here in Section 4.4 that the computational guarantees of two standard first-order methods for smooth optimization – the Primal Gradient Scheme and the Dual Averaging Scheme – extend in precise ways to the case when \( f(\cdot) \) is \( L \)-smooth
relative to the reference function \( h(\cdot) \). The proof techniques used here suggest that very many other first-order algorithms for smooth optimization should extend similarly with analogous computational guarantees. However, we have not been able to extend any accelerated methods, i.e., methods that attain an \( O(1/k^2) \) convergence guarantee such as [78], [76], [105], to the relatively smooth case. One avenue for further research is to answer the question whether one can develop computational guarantees for an accelerated method in the case when \( f(\cdot) \) is \( L \)-smooth relative to the reference function \( h(\cdot) \)?

Another question that arises concerns conjugate (duality) theory for the setting of relatively smooth convex functions. One simple result in conjugate duality theory is that when \( f(\cdot) \) is \( L \)-smooth (relative to \( h(\cdot) := \frac{1}{2} \| \cdot \|^2 \)) the conjugate function \( f^*(\cdot) \) is \( 1/L \)-strongly convex (relative to \( h^*(\cdot) := \frac{1}{2} \| \cdot \|^2 \)), see [114]. Is there a way to develop a more general conjugate duality theory that yields an analogous result when \( f(\cdot) \) is \( L \)-smooth relative to a general convex function \( h(\cdot) \)?

A third question is how can we choose the reference function \( h(\cdot) \) in order to lower the value of the bounds in Theorems 3.3.1 and 3.3.2? Several ways to think about this question are discussed in Appendix B.3.

### 3.4 D-Optimal Design Revisited: Computational Guarantees using the Primal Gradient or Dual Averaging Scheme

Let us now apply the computational guarantees for the Primal Gradient Scheme (Theorem 3.3.1) and the Dual Averaging Scheme (Theorem 3.3.2) to the \( D \)-optimal design optimization problem (3.16) discussed in Section 3.2.2. Recall from the exposition in Section 3.2.2 that \( Q = \Delta_n \) and \( f(x) = -\ln \det(HXH^T) \) is 1-smooth relative to the
logarithmic barrier function

\[ h(x) = -\sum_{j=1}^{n} \ln(x_j), \tag{3.39} \]

and that the subproblem (4.7) is efficiently solvable. The following theorem presents a computational guarantee for using the Primal Gradient Scheme to approximately solve the D-optimal design optimization problem (3.16).

**Theorem 3.4.1.** Consider using the Primal Gradient Scheme (Algorithm 6) with the reference function (3.39) to solve the D-optimal design problem (3.16) using the initial point \( x^0 = \frac{1}{n} e \), and suppose that \( \varepsilon \leq f(x^0) - f^* \). If

\[ k \geq \frac{2n \ln \left( \frac{2(f(x^0) - f^*)}{\varepsilon} \right)}{\varepsilon}, \]

then \( f(x^k) - f^* \leq \varepsilon \).

**Proof:** Let \( \delta = \frac{\varepsilon}{2(f(x^0) - f^*)} \). Then \( \delta \leq \frac{1}{2} \) since \( \varepsilon \leq f(x^0) - f^* \). Let \( \hat{x} := (1 - \delta)x^* + \delta x^0 \). It follows from the convexity of \( f(\cdot) \) that

\[ f(\hat{x}) \leq (1 - \delta)f^* + \delta f(x^0), \]

whereby

\[ f(\hat{x}) - f^* \leq \delta(f(x^0) - f^*). \tag{3.40} \]

Meanwhile,

\[ D_h(\hat{x}, x^0) = h(\hat{x}) - h(x^0) - \langle \nabla h(x^0), \hat{x} - x^0 \rangle = h(\hat{x}) - h(x^0) \leq -n \ln \left( \frac{\delta}{n} \right) + n \ln \left( \frac{1}{n} \right) = n \ln(1/\delta), \tag{3.41} \]

where the second equality uses \( \nabla h(x^0) = -n \cdot e \) which then implies \( \langle \nabla h(x^0), \hat{x} - x^0 \rangle = 0 \), and the inequality follows since \( \hat{x} \geq (\delta/n)e \). Therefore, for \( k \) satisfying the
inequality in the statement of the theorem, we have:

\[
\begin{align*}
f(x^k) - f^* & = f(x^k) - f(\bar{x}) + f(\bar{x}) - f^* \\
& \leq \frac{D_h(\bar{x}, x^0)}{k} + \delta(f(x^0) - f^*) \\
& \leq \frac{n \ln(1/\delta)}{k} + \frac{\varepsilon}{2} \\
& \leq \varepsilon,
\end{align*}
\]  

(3.42)

where the first inequality follows from Theorem 3.3.1 using \( x = \bar{x} \), as well as (3.40), the second inequality is from (4.22) and the definition of \( \delta \), and the third inequality follows since \( k \geq [2n \ln(1/\delta)]/\varepsilon \).

\[ \square \]

**Remark 3.4.1.** For the Dual Averaging Scheme (Algorithm 7), one obtains the identical bound as in Theorem 3.4.1. This is proved by following virtually the same logic as above, except we use Theorem 3.3.2 which bounds the smallest optimality gap using \( h(x) - h(x^0) \) instead of \( D_h(x, x^0) \). However, it follows from (4.22) that these two quantities are the same in this case. Also, in the case of the Dual Averaging Scheme, the relevant final quantity of interest is \( \min_{i=1,\ldots,k} f(x^i) - f^* \) instead of \( f(x^k) - f^* \).

It is instructive to compare the computational guarantees in Theorem 3.4.1/Remark 3.4.1 to those of the Frank-Wolfe method applied to \( D \)-optimal design (first analyzed by Khachiyan [53] and re-evaluated in [1] based in part on work by Yildirim [112]). Table 3.1 shows such a comparison, where absolute constants have been suppressed in order to highlight the dependencies on particular quantities of interest. The second column of Table 3.1 compares the iteration bound of the methods using the starting point \( x^0 = (1/n)e \), where we emphasize that \( \varepsilon \) is the target optimality gap for the \( D \)-optimal design problem. While it follows from observations in [53] that \( f(x^0) - f^* \leq m \ln(n/m) \) for \( x^0 = (1/n)e \), we do not show this in Table 3.1, as we wish to highlight where the dependence on the initial iterate arises. Examining the
first column of Table 3.1, note that the number of iterations of the Primal Gradient Scheme (or Dual Averaging Scheme) can be less than that of the Frank-Wolfe method, especially when ε is not too small and when \( n \ll m^2 \). However, as the second column of Table 3.1 shows, the Frank-Wolfe method requires only \( mn \) operations per iteration in the worst – i.e., dense matrix – case, as it does a rank-1 update of a matrix inverse in the computation of \( \nabla f(x^k) \), whereas the Primal Gradient Scheme (or Dual Averaging Scheme) requires \( m^2 n \) operations per iteration in the dense case (it must re-compute a matrix inverse in order to work with \( \nabla f(x^k) \)). Therefore the total bound on operations of the Frank-Wolfe method (shown in the last column of Table 3.1) is superior.

The bound for the Frank-Wolfe method applied to the \( D \)-optimal design problem is based on analysis that is uniquely designed for evaluating the \( D \)-optimal design problem, and is not part of the general theory for the Frank-Wolfe method (that we are aware of). Even though the Primal Gradient Scheme and the Dual Averaging Scheme have inferior computational guarantees to the Frank-Wolfe method applied to the \( D \)-optimal design problem, they are the first (that we are aware of) first-order methods for which one has a general theory (Theorems 3.3.1 and 3.3.2) that can be meaningfully applied to yield computational guarantees for the \( D \)-optimal design problem. We hope that this analysis will spur further interest in developing improved algorithms for \( D \)-optimal design and its dual problem – the minimum volume enclosing ellipsoid problem.
Frank-Wolfe Method

Primal Gradient Scheme
or Dual Averaging Scheme

Table 3.1: Comparison of the order of computational guarantees for the Frank-Wolfe Method [53], [1] and the Primal Gradient and Dual Averaging Schemes (Theorem 3.4.1 and Remark 3.4.1) for D-optimal design. All constants have been suppressed in order to highlight the dependencies on particular quantities of interest. It also follows from [53] that $f(x^0) - f^* \leq m \ln(n/m)$ for $x^0 = (1/n)e$, which can be inserted in the above bounds as well.
Chapter 4

Relatively Continuous Convex Optimization by First-Order Methods, and Applications

4.1 Introduction

The usual approach to developing and analyzing first-order methods for non-differentiable convex optimization (which we review shortly) assumes that the objective function is uniformly Lipschitz continuous, in both deterministic and stochastic settings. However, in many settings the objective function $f$ is not uniformly Lipschitz continuous. For example, consider the Support Vector Machine problem (SVM) for binary classification in machine learning, whose optimization formulation is:

$$\text{SVM: } \min_x f(x) := \frac{1}{n} \sum_{i=1}^{n} \max \{0, 1 - y_i x^T w_i\} + \frac{1}{2} \|x\|^2,$$

where $w_i$ is the input feature vector of sample $i$ and $y_i \in \{-1, 1\}$ is the label of sample $i$. Notice that $f(\cdot)$ is not differentiable due to the presence of hinge loss terms in the summation, and $f(\cdot)$ is also not Lipschitz continuous due to the presence of
the $\ell_2$-norm regularization term; thus we cannot directly utilize typical subgradient or gradient schemes and their associated computational guarantees for SVM.

Another example is the problem of computing a point $x \in \mathbb{R}^m$ in the intersection of $n$ ellipsoids, which can be tackled via the optimization problem

$$\text{IEP: } f^* = \min_x f(x) := \max_{0 \leq i \leq n} \{ \frac{1}{2} x^T A_i x + b_i^T x + c_i \} ,$$

where the $i^{th}$ ellipsoid is $Q_i = \{ x \in \mathbb{R}^m : \frac{1}{2} x^T A_i x + b_i x + c_i \leq 0 \}$ and $A_i \in \mathbb{R}^{m \times m}$ is a symmetric positive semi-definite matrix, $i = 1, \ldots, n$. Observe that the objective function $f(\cdot)$ is both non-differentiable and non-Lipschitz, and so it falls outside of the scope of standard classes of optimization problems for which first-order methods are guaranteed to work. Nevertheless, using the machinery developed in this chapter, we will show in Section 4.5.3 how to solve both of these problems using deterministic or stochastic Mirror Descent.

In this chapter we develop a general theory and algorithmic constructs that overcome the drawbacks in the usual analyses of first-order methods that are grounded on restricted notions of uniform Lipschitz continuity. Here we develop a notion of “relative continuity” with respect to a given convex “reference function” $h$, a notion which does not require the specification of any particular norm – indeed $h$ need not be strongly (or even strictly) convex. Armed with “relative continuity”, we demonstrate the capability to solve a more general class of non-differentiable convex optimization problems (without uniform Lipschitz continuity) in both deterministic and stochastic settings.

This chapter is a companion for non-differentiable convex optimization to the previous chapter for differentiable convex optimization. In [6], with a very similar philosophy, we developed the notion of relative smoothness and relative strong convexity with respect to a given convex reference function. In that chapter we demonstrated the capability to solve a more general class of differentiable convex optimization problems (without uniform Lipschitz continuous gradients), and we also
demonstrated linear convergence results for a Primal Gradient Scheme when the objective function $f(\cdot)$ is both smooth and strongly convex.

There are some concurrent works on smooth optimization sharing a similar spirit to [61]. Bauschke, Bolte, and Teboulle [8] presents a similar definition of relative smoothness as in [61] and analyzes the convergence of Mirror Descent Algorithm, although their algorithm and convergence complexity depend on a symmetry measure of the Bregman distance. Zhou et al. [118] discusses a unified proof of Mirror Descent and the Proximal Point Algorithm under a similar assumption of relative smoothness. Nguyen [107] develops similar ideas on analyzing Mirror Descent in a Banach space. A more detailed discussion comparing these related works is also presented in [61]. More recently, Hanzely and Richtarik [45] develop stochastic algorithms for the relatively smooth optimization setting.

In Section 2 we review the traditional set-up for Mirror Descent in both the deterministic and stochastic settings. In Section 3 we introduce our notion of "relative continuity" in both the deterministic and stochastic settings, together with some relevant properties. In Section 4 we prove computational guarantees associated with the Mirror Descent and Stochastic Mirror Descent algorithms under relative continuity. In Section 5 we show constructively how our ideas apply to a large class of non-differentiable and non-Lipschitz convex optimization problems that are not otherwise solvable by traditional first-order methods. Also in Section 5 we analyze computational guarantees associated with Mirror Descent and Stochastic Mirror Descent for the Intersection of Ellipsoids Problem (IEP) and also the Support Vector Machine (SVM) problem.

**Notation.** $\| \cdot \|$ denotes a given norm in $\mathbb{R}^n$ and $\| \cdot \|_*$ denotes the usual dual norm on the dual space. $\| x \|_2 := \sqrt{x^T x}$ denotes the Euclidean (inner product) norm, where $x^T$ means the transpose of the vector $x$, and $B_2(c, r) := \{ x \in \mathbb{R}^n : \| x - c \|_2 \leq r \}$. $\| A \|_2$ denotes the $\ell_2$ (spectral) norm of a matrix $A$. The inner product $\langle \cdot, \cdot \rangle$ specifically denotes the dot inner product in the underlying vector space. For a conditional random variable $s(x)$ given $x$, $\mathbb{E}[s(x)|x]$ denotes the conditional expectation of $s(x)$.
given $x$.

### 4.2 Traditional Mirror Descent

The optimization problem of interest is:

$$ P : \quad \min_x f(x) $$

where $Q \subseteq \mathbb{R}^n$ is a closed convex set and $f : Q \rightarrow \mathbb{R}$ is a convex function that is not necessarily differentiable. There are very many deterministic and stochastic first-order methods for tackling (4.1), see for example [17], [76], [69] and the references therein. Virtually all such methods are designed to solve (4.1) when the objective function $f$ satisfies a uniform Lipschitz continuity condition on $Q$, which in the deterministic setting is (essentially) equivalent to the condition that there exists a constant $M_f < \infty$ for which:

$$ \|g(x)\|_* \leq M_f \quad \text{for all } x \in Q \text{ and } g(x) \in \partial f(x), $$

(4.2)

where $\partial f(x)$ is the subdifferential of $f(\cdot)$ at $x$ (i.e., the collection of subgradients of $f(\cdot)$ at $x$), $\| \cdot \|$ is a given norm on $\mathbb{R}^n$, and $\| \cdot \|_*$ denotes the usual dual norm.

Here we use “$g(x)$” to denote an assignment of a subgradient (or an oracle call thereof) at $x$, and so $g(x)$ is not a function nor is it a point-to-set map.

Another useful functional notion is strong convexity: $f(\cdot)$ is (uniformly) $\mu_f$-strongly convex for some $\mu_f > 0$ if

$$ f(y) \geq f(x) + \langle g(x), y - x \rangle + \frac{\mu_f}{2} \|y - x\|^2 \quad \text{for all } x, y \in Q \text{ and } g(x) \in \partial f(x). $$

(4.3)

Nedić and Lee [69] obtain improved convergence guarantees for the stochastic mirror descent algorithm under strong convexity, for example.
4.2.1 Deterministic Setting

Let us now recall the Mirror Descent Algorithm (see [71] and [9]), which is also referred to as the prox subgradient method when interpreted in the space of primal variables. Mirror Descent employs a differentiable convex “prox function” $h$ to define a Bregman distance:

$$D_h(y, x) := h(y) - h(x) - \langle \nabla h(x), y - x \rangle \quad \text{for all } x, y \in Q.$$  \hspace{1cm} (4.4)

The Bregman distance is used in the computation of the Mirror Descent update:

$$x^{i+1} \leftarrow \arg \min_{x \in Q} \left\{ f(x^i) + \langle g(x^i), x - x^i \rangle + \frac{1}{t_i} D_h(x, x^i) \right\},$$

where $\{t_i\}$ is the sequence of step-sizes for the scheme. A formal statement of the Mirror Descent Algorithm is presented in Algorithm 8. The traditional set-up requires that $h(\cdot)$ is 1-strongly convex with respect to the given norm $\| \cdot \|$, and in this set-up one can prove that after $k$ iterations it holds for any $x \in Q$ that:

$$\min_{0 \leq i \leq k} f(x^i) - f(x) \leq \frac{\frac{1}{2} M_f^2 \sum_{i=0}^{k} t_i^2 + D_h(x, x^0)}{\sum_{i=0}^{k} t_i},$$  \hspace{1cm} (4.5)

which leads to an $O(1/\sqrt{k})$ sublinear rate of convergence using an appropriately chosen step-size sequence $\{t_i\}$, see [9].

**Algorithm 8** Deterministic Mirror Descent Algorithm with Bregman distance $D_h(\cdot, \cdot)$

*Initialize.* Initialize with $x^0 \in Q$. Let $h(\cdot)$ be a given convex function.

At iteration $i$:

**Perform Updates.** Compute $g(x^i) \in \partial f(x^i)$, determine step-size $t_i$, and compute update:

$$x^{i+1} \leftarrow \arg \min_{x \in Q} \left\{ f(x^i) + \langle g(x^i), x - x^i \rangle + \frac{1}{t_i} D_h(x, x^i) \right\}. \hspace{1cm} (4.6)$$

Notice in (4.6) by construction that the update requires the capability to solve
instances of a "linearized subproblem" (which we denote by "LS") of the general form:

\[
\text{LS} : \quad x_{\text{new}} \leftarrow \arg\min_{x \in Q} \{ \langle c, x \rangle + h(x) \},
\]  

(4.7)

for suitable iteration-specific values of \( c \). Indeed, (4.6) is an instance of (4.7) with \( c = t_ig(x^i) - \nabla h(x^i) \) at iteration \( i \). It is especially important to note that the Mirror Descent update (4.6) is somewhat meaningless absent the capability to efficiently solve (4.7), a fact which we return to later. In the usual design and implementation of Mirror Descent for solving (4.1), one attempts to specify the norm \( \| \cdot \| \) and the 1-strongly convex prox function \( h \) in consideration of properties of the feasible domain \( Q \) while also ensuring that the LS subproblem (4.7) is efficiently solvable.

Notice that the Mirror Descent Algorithm (Algorithm 8) itself does not require the traditional set-up that \( h(\cdot) \) be 1-strongly convex for some particular norm; rather this requirement is part of the traditional analysis. As we will see, we can instead analyze Mirror Descent by considering the intrinsic ways that \( f(\cdot) \) and \( D_h(\cdot, \cdot) \) are related functionally, in a manner that is constructive in terms of actual algorithm design and implementation. Furthermore, this is in the same spirit as was done in the predecessor chapter.

4.2.2 Stochastic Setting

For some convex functions, computing an exact subgradient at \( x \in Q \) may be expensive or even intractable, but sampling a random stochastic estimate of a subgradient at \( x \), which we denote by \( \tilde{g}(x) \), may be easy. We say that \( \tilde{g}(x) \) is an unbiased stochastic subgradient if \( \mathbb{E}[\tilde{g}(x)|x] \in \partial f(x) \). The usefulness of a stochastic subgradient methodology is easily seen in the context of machine and statistical learning problems. A prototypical learning problem is to compute an approximate solution of the following
empirical loss minimization problem:

\[
\min_x f(x) := \frac{1}{n} \sum_{j=1}^{n} f_j(x)
\]

\(\text{s.t. } x \in Q\),

where \(f_j\) is a non-differentiable convex loss function associated with sample \(j\), for \(j = 1, \ldots, n\) data samples. When \(n \gg 0\), the standard subgradient method needs to evaluate \(n\) subgradients in order to compute a subgradient of \(f(\cdot)\), which can be prohibitively expensive. A typical alternative is to compute a stochastic subgradient. Letting \(x^i\) denote the \(i^{th}\) iterate, at iteration \(i\) a single sample index \(\tilde{j}\) is drawn uniformly and independently on \(\{1, \ldots, n\}\), and then a subgradient \(\tilde{g} \in \partial f_{\tilde{j}}(x^i)\) is computed that is used to define \(\tilde{g}(x^i) := \tilde{g}\). This stochastic subgradient is then used in place of a subgradient at iteration \(i\). Notice that by construction \(\tilde{g}(x^i)\) is a conditional random variable given \(x^i\), and \(\tilde{g}(x^i)\) is an unbiased stochastic subgradient, namely \(E[\tilde{g}(x^i)|x^i] \in \partial f(x^i)\).

A stochastic version of Mirror Descent is presented in Algorithm 9. The structure of Stochastic Mirror Descent is identical to that of Mirror Descent, the only difference being that the stochastic estimate of a subgradient \(\tilde{g}(x^i)\) replaces the exact subgradient \(g(x^i)\) in Algorithm 9.

\begin{algorithm}
\caption{Stochastic Mirror Descent Algorithm with Bregman distance \(D_h(\cdot, \cdot)\)}
\begin{algorithmic}
\State \textbf{Initialize.} Initialize with \(x^0 \in Q\). Let \(h(\cdot)\) be a given convex differentiable function.
\State At iteration \(i\): \textbf{Perform Updates.} Compute a stochastic subgradient \(\tilde{g}(x^i)\), determine step-size \(t_i\), and compute update:
\[
x^{i+1} \leftarrow \arg \min_{x \in Q} \{f(x^i) + \langle \tilde{g}(x^i), x - x^i \rangle + \frac{1}{t_i} D_h(x, x^i)\}.
\]
\end{algorithmic}
\end{algorithm}

A standard condition that is required in the traditional convergence analysis for Stochastic Mirror Descent (as well as other stochastic first-order methods) is that
there exists $G_f > 0$ for which:

$$
\mathbb{E}[\|g(x)\|^2] \leq G_f^2, \text{ for any } x \in Q .
$$

(4.9)

For notational convenience, we say that $f(\cdot)$ is $G_f$-stochastically continuous if (4.9) holds. In [69], Nedić and Lee developed convergence results for Stochastic Mirror Descent (Algorithm 2). Under the conditions that (i) $f(\cdot)$ is $G_f$-stochastically continuous, (ii) $h(\cdot)$ is a differentiable and $\mu_h$-strongly convex function on $Q$, and (iii) $Q$ is a closed bounded set, Nedić and Lee ([69] equation (27)) show the following convergence result using step-sizes $t_i = \frac{\mu_h D_{\text{max}}}{G_f(i+1)}$ :

$$
\mathbb{E}[f(\bar{x}^k)] - f^* \leq \frac{3G_f \sqrt{D_{\text{max}}}}{2\sqrt{\mu_h(k+1)}},
$$

(4.10)

where $\bar{x}^k := \frac{1}{\sum_{i=0}^k t_i} \sum_{i=0}^k t_i x^i$ and $D_{\text{max}} := \max_{x,y \in Q} D_h(x,y)$.

Furthermore, if also (a) $f(\cdot)$ is $\mu_f$-strongly convex, and (b) $h(\cdot)$ is $L_h$-smooth, Nedić and Lee ([69] Theorem 1) show that with step-sizes $t_i = \frac{2L_h}{\mu_f(i+1)}$ it holds that:

$$
\mathbb{E}[f(\bar{x}^k)] - f^* \leq \frac{2G_f^2 L_h}{\mu_f(k+1)\mu_h},
$$

(4.11)

where $\bar{x}^k := \frac{2}{(i+1)(i+2)} \sum_{i=0}^k (i+1)x^i$.

### 4.3 Relative Continuity

In this section we introduce our definition of relative continuity of a function $f(\cdot)$ – actually two different definitions – one for the deterministic and another for the stochastic setting. The starting point is a “reference function” $h(\cdot)$ which is a given differentiable convex function on $Q$ that is used to construct the usual Bregman distance $D_h(\cdot, \cdot)$ (4.4), and that is used as part of the Mirror Descent update (4.6). However, we point out for emphasis that unlike the traditional set-up there are no
assumptions on $h(\cdot)$ (such as strong or strict convexity).

### 4.3.1 Deterministic Setting

Consider the objective function $f(\cdot)$ of (4.1). We define “relative continuity” of $f(\cdot)$ relative to the reference function $h(\cdot)$ using the Bregman distance $D_h(\cdot, \cdot)$ of $h(\cdot)$ as follows.

**Definition 4.3.1.** $f(\cdot)$ is $M$-relative continuous with respect to the reference function $h(\cdot)$ on $Q$ if for any $x, y \in Q$, $x \neq y$, and $g(x) \in \partial f(x)$, it holds that

$$
\|g(x)\|_* \leq \frac{M\sqrt{2D_h(y, x)}}{\|y - x\|}.
$$

(In the particular case when $h(x) = \frac{1}{2}\|x\|_2^2$, the Bregman distance is $D_h(y, x) = \frac{1}{2}\|y - x\|_2^2$, and the relative continuity condition (4.12) becomes $\|g(x)\|_2 \leq M$, which corresponds to the standard definition of Lipschitz continuity (4.2) for the $\ell_2$-norm.)

We can rewrite (4.12) as

$$
\|g(x)\|^2 \leq M^2 \frac{D_h(y, x)}{\frac{1}{2}\|y - x\|^2},
$$

which states that the square of the norm of any subgradient is bounded by the ratio of the Bregman distance $D_h(y, x)$ to $\frac{1}{2}\|y - x\|^2$.

The following proposition presents the “key property” of an $M$-relative continuous function that is used in the proofs of results to follow.

**Proposition 4.3.1. (Key property of $M$-relative continuity)** If $f(\cdot)$ is $M$-relative continuous with respect to the reference function $h(\cdot)$, then for any $t > 0$ it holds for all $x, y \in Q$ and $g(x) \in \partial f(x)$ that:

$$
\frac{1}{t}D_h(y, x) + \langle g(x), y - x \rangle + \frac{1}{2}tM^2 \geq 0.
$$

103
Proof: If \( f(\cdot) \) is \( M \)-relative continuous with respect to \( h(\cdot) \), then for any \( t > 0 \) it follows that

\[
-(g(x), y - x) \leq \|g(x)\| \|y - x\| \leq M\sqrt{2D_h(y, x)} \leq \frac{1}{2}tM^2 + \frac{D_h(y, x)}{t},
\]

where the last inequality is an application of the arithmetic-geometric mean inequality. The proof follows by rearranging terms.

The "key property" (4.14) is what is used in the proofs of results herein, so we could define \( M \)-relative continuity using (4.14) instead of (4.12). Furthermore, (4.14) is independent of any norm structure, and so is attractive for its generality. However, we use the definition (4.12) because it leads to easy verification of \( M \)-relative continuity in practical instances as is shown in Section 4.5.

The following proposition presents some scaling and additivity properties of relative continuity.

Proposition 4.3.2. Additivity of Relative Continuity

1. If \( f(\cdot) \) is \( M \)-relative continuous with respect to \( h(\cdot) \), then for any \( \alpha > 0 \), \( f(\cdot) \) is \( \frac{M}{\alpha} \)-relative continuous with respect to \( \alpha^2 h(\cdot) \).

2. If \( f(\cdot) \) is \( M \)-relative continuous with respect to \( h(\cdot) \), then for any \( \alpha > 0 \), \( \alpha f(\cdot) \) is \( M \)-relative continuous with respect to \( \alpha^2 h(\cdot) \).

3. If \( f_j \) is \( M \)-relative continuous with respect to \( h_j \) for \( j = 1, \ldots, n \), then \( \sum_{j=1}^{n} f_j \) is \( \sqrt{n}M \)-relative continuous with respect to \( \sum_{j=1}^{n} h_j \).

4. If \( f_j \) is \( M_j \)-relative continuous with respect to \( h_j \) for \( j = 1, \ldots, n \), then for \( \alpha_j > 0 \) and \( M > 0 \) it holds that \( \sum_{j=1}^{n} \alpha_j f_j \) is \( \sqrt{n}M \)-relative continuous with respect to \( \sum_{j=1}^{n} \frac{\alpha_j^2}{\beta_j^2} h_j \) with \( \beta_j := \frac{M}{M_j} \).

Proof: Let \( x, y \in Q, x \neq y \), and \( g(x) \in \partial f(x) \).
1. It holds that

\[ \|g(x)\|_* \leq \frac{M \sqrt{2D_h(y, x)}}{\|y - x\|} = \frac{M \alpha \sqrt{2D_{\alpha^2 h}(y, x)}}{\|y - x\|}, \]

which establishes the result.

2. Notice that \( g(x) \) is a subgradient of \( f(x) \) if and only if \( \alpha g(x) \) is a subgradient of \( \alpha f(x) \), whereby

\[ \|\alpha g(x)\|_* = \alpha \|g(x)\|_* \leq \frac{\alpha M \sqrt{2D_h(y, x)}}{\|y - x\|} = \frac{M \sqrt{2D_{\alpha^2 h}(y, x)}}{\|y - x\|}, \]

which establishes the result.

3. Any subgradient of \( \sum_{j=1}^{n} f_j \) at \( x \) can be written as \( \sum_{j=1}^{n} g_j(x) \) where \( g_j(x) \in \partial f_j(x) \) for \( j = 1, \ldots, n \) (see Theorem B.21 of [11]). From the triangle inequality and the relative continuity of \( f_j \) we have:

\[
\left\| \sum_{j=1}^{n} g_j(x) \right\|_* \leq \sum_{j=1}^{n} \|g_j(x)\|_* \leq \frac{M \left( \sum_{j=1}^{n} \sqrt{2D_{h_j}(y, x)} \right)}{\|y - x\|} \leq \frac{\sqrt{n} M \left( \sqrt{2 \sum_{j=1}^{n} D_{h_j}(y, x)} \right)}{\|y - x\|} = \frac{\sqrt{n} M \sqrt{2D_{h_1 + \cdots + h_n}(y, x)}}{\|y - x\|},
\]

where the third inequality is an application of the \( \ell_1/\ell_2 \)-norm inequality applied to the \( n \)-tuple \( (\sqrt{2D_{h_1}(y, x)}, \ldots, \sqrt{2D_{h_n}(y, x)}) \).

4. It follows from part (2.) that \( \alpha_j f_j \) is \( M_j \)-continuous relative to \( \alpha_j^2 h_j \). Thus \( \alpha_j f_j \) is also \( M \)-continuous relative to \( \frac{\alpha_j^2}{\beta_j^2} h_j \) from part (1.), whereby the proof is finished by utilizing part (3.). \( \Box \)

We also make use of the notion of “relative strong convexity” which was introduced in [61], and is used here in some of the convergence guarantee analyses.

**Definition 4.3.2.** \( f(\cdot) \) is \( \mu \)-strongly convex relative to \( h(\cdot) \) on \( Q \) if there is a scalar
\[ f(y) \geq f(x) + \langle g(x), y - x \rangle + \mu D_h(y, x) \quad \text{for any } x, y \in \text{int } Q \text{ and any } g(x) \in \partial f(\cdot) \text{ it holds that} \]

In [61] it was shown that the notion of relative strong convexity embodied in Definition 4.3.2 is the natural way to define strong convexity in the context of mirror descent and similar algorithms, and leads to linear convergence of mirror descent in the smooth setting. In the non-smooth setting, we will show in Theorem 4.4.2 that relative strong convexity improves the convergence of mirror descent from \( O(1/\sqrt{k}) \) to \( O(1/k) \).

### 4.3.2 Stochastic Setting

For \( x \in Q \), let \( \tilde{g}(x) \) denote a random stochastic estimate of a subgradient of \( f(\cdot) \) at \( x \). Extending the definition of relative continuity from the deterministic setting, we define stochastic relative continuity as follows.

**Definition 4.3.3.** \( f(\cdot) \) is \( G \)-stochastically-relative continuous with respect to the reference function \( h(\cdot) \) on \( Q \) for some \( G > 0 \) if \( f(\cdot) \) together with the oracle to compute a stochastic subgradient satisfies:

1. Unbiasedness property: \( \mathbb{E}[\tilde{g}(x)|x] \in \partial f(x) \), and
2. Boundedness property: \( \mathbb{E}[\|\tilde{g}(x)\|_2^2|x] \leq G^2 \frac{D_h(y, x)}{\frac{1}{2}||y - x||^2} \) for all \( x, y \in Q \) and \( x \neq y \).

(In the particular case when \( h(x) = \frac{1}{2}\|x\|_2^2 \), the Bregman distance is \( D_h(y, x) = \frac{1}{2}\|y - x\|_2^2 \), whereby the stochastically-relative continuity boundedness property becomes \( \mathbb{E}[\|\tilde{g}(x)\|_2^2|x] \leq G^2 \) for all \( x \in Q \), which corresponds to the standard condition (4.9) for the \( \ell_2 \)-norm.)

For \( x \in Q \), define

\[ \tilde{M}(x) := \|\tilde{g}(x)\|_\ast \max_{y \in Q, y \neq x} \frac{\|y - x\|}{\sqrt{2D_h(y, x)}}. \]

106
Notice for a given $x$ that $\max_{y \in Q, y \neq x} \frac{\|y - x\|}{\sqrt{2D_h(y, x)}}$ is a deterministic quantity, and therefore $\tilde{M}(x)$ is a conditional random variable (given $x$) that is defined on the same probability space as $\tilde{g}(x)$. Clearly, if $f(\cdot)$ is $G$-stochastically-relative continuous, we have by the boundedness property that for any $x \in Q$

$$E[\tilde{M}(x)^2|x] = E[\|\tilde{g}(x)\|^2|x] \max_{y \in Q, y \neq x} \frac{\|y - x\|^2}{2D_h(y, x)} \leq G^2. \quad (4.17)$$

Exactly as in the deterministic setting, we have:

**Proposition 4.3.3.** If $f(\cdot)$ is $G$-stochastically-relative continuous with respect to the reference function $h(\cdot)$ on $Q$, then for any $t > 0$ it holds for all $x, y \in Q$ and any stochastic subgradient estimate $\tilde{g}(x)$ that:

$$\frac{1}{t}D_h(y, x) + \langle \tilde{g}(x), y - x \rangle + \frac{1}{2}t\tilde{M}^2(x) \geq 0.$$  

**Proof:** For any $t > 0$, we have

$$-\langle \tilde{g}(x), y - x \rangle \leq \|\tilde{g}(x)\|\|y - x\| \leq \tilde{M}(x)\sqrt{2D_h(y, x)} \leq \frac{1}{2}t\tilde{M}(x)^2 + \frac{D_h(y, x)}{t},$$

and the proof is finished by rearranging terms. \qed

### 4.4 Computational Analysis for Stochastic Mirror Descent and (Deterministic) Mirror Descent

In this section we present computational guarantees for Stochastic Mirror Descent (Algorithm 9) for minimizing a convex function $f(\cdot)$ that is $G$-stochastically-relative continuous with respect to a given reference function $h(\cdot)$. We also present computational guarantees for (deterministic) Mirror Descent (Algorithm 8) when $f(\cdot)$ is $M$-relative continuous with respect to a reference function $h(\cdot)$, which follows as a special case of the stochastic setting.
We begin by recalling the standard Three-Point Property for optimization using Bregman distances:

**Lemma 4.4.1. (Three-Point Property (Tseng [105]))** Let $\phi(x)$ be a convex function, and let $D_h(\cdot, \cdot)$ be the Bregman distance for $h(\cdot)$. For a given vector $z$, let

$$z^+ := \arg\min_{x \in Q} \{\phi(x) + D_h(x, z)\}.$$ 

Then

$$\phi(x) + D_h(x, z) \geq \phi(z^+) + D_h(z^+, z) + D_h(x, z^+) \quad \text{for all } x \in Q. \quad \square$$

Let us denote the (primitive) random variable at the $i^{\text{th}}$ iteration of the Stochastic Mirror Descent Algorithm (Algorithm 9) by $\gamma_i$, i.e., $\gamma_i$ is the random variable that determines the (stochastic) subgradient $\tilde{g}(x^i)$ at iterate $x^i$ in the Stochastic Mirror Descent Algorithm. Then $x^{i+1}$ is computed according to the update of the Stochastic Mirror Descent Algorithm, whereby $x^{i+1}$ is a random variable which depends on all previous values $\gamma_0, \ldots, \gamma_i$ and we denote this string of random variables by

$$\xi_i := \{\gamma_0, \ldots, \gamma_i\}.$$

The following theorem states convergence guarantees for the Stochastic Mirror Descent Algorithm in terms of expectation.

**Theorem 4.4.1. (Convergence Bound for Stochastic Mirror Descent Algorithm)** Consider the Stochastic Mirror Descent Algorithm (Algorithm 9) with given step-size sequence $\{t_i\}$. If $f(\cdot)$ is $G$-stochastically-relative continuous with respect to $h(\cdot)$ for some $G > 0$, then the following inequality holds for all $k \geq 1$ and $x \in Q$:

$$\mathbb{E}_{\xi_{k-1}} \left[f(\bar{x}^k)\right] - f(x) \leq \frac{1}{2} G^2 \frac{\sum_{i=0}^k t_i^2 + D_h(x, x^0)}{\sum_{i=0}^k t_i},$$

where $\bar{x}^k := \frac{1}{\sum_{i=0}^k t_i} \sum_{i=0}^k t_i x^i$. 

108
Proof: First notice that

\[ f(x^i) + \langle g(x^i), x - x^i \rangle \]

\[ = f(x^i) + \langle \mathbb{E}_{\xi_i}[\tilde{g}(x^i)|x^i], x - x^i \rangle \]

\[ = f(x^i) + \mathbb{E}_{\xi_i}[(\tilde{g}(x^i), x - x^i)|x^i] \]

\[ \geq f(x^i) + \mathbb{E}_{\xi_i}[(\tilde{g}(x^i), x^{i+1} - x^i) + \frac{1}{t_i} D_h(x^{i+1}, x^i) + \frac{1}{t_i} D_h(x, x^{i+1}) - \frac{1}{t_i} D_h(x, x^i)|x^i] \]

\[ \geq f(x^i) + \mathbb{E}_{\xi_i}[-\frac{1}{2} \tilde{M}(x^i)^2 t_i + \frac{1}{t_i} D_h(x, x^{i+1}) - \frac{1}{t_i} D_h(x, x^i)|x^i] \]

\[ \geq f(x^i) - \frac{1}{2} G^2 t_i + \frac{1}{t_i} \mathbb{E}_{\xi_i}[D_h(x, x^{i+1})|x^i] - \frac{1}{t_i} D_h(x, x^i) , \] \hspace{1cm} (4.19)

where the first equality uses the unbiasedness of \( \tilde{g}(x) \), the second equality is because of linearity, the first inequality is from the Three-Point Property with \( \phi(x) = t_i (\tilde{g}(x^i), x - x^i) \), the second inequality uses Proposition 4.3.3, and the last inequality uses (4.17).

Since also \( f(x) \geq f(x^i) + \langle g(x^i), x - x^i \rangle \) from the definition of a subgradient, we have from (4.19):

\[ f(x) \geq f(x^i) + \langle g(x^i), x - x^i \rangle \geq f(x^i) - \frac{1}{2} G^2 t_i + \frac{1}{t_i} \mathbb{E}_{\xi_i}[D_h(x, x^{i+1})|x^i] - \frac{1}{t_i} D_h(x, x^i) . \]

Taking expectation with respect to \( \xi_i \) on both sides of the above inequality yields:

\[ f(x) \geq \mathbb{E}_{\xi_i-1}[f(x^i)] - \frac{1}{2} G^2 t_i + \frac{1}{t_i} \mathbb{E}_{\xi_i}[D_h(x, x^{i+1})] - \frac{1}{t_i} \mathbb{E}_{\xi_i-1}[D_h(x, x^i)] \] \hspace{1cm} (4.20)

Now rearrange and multiply through by \( t_i \) to yield:

\[ t_i \mathbb{E}_{\xi_i-1}[f(x^i) - f(x)] \leq \frac{1}{2} G^2 t_i^2 + \mathbb{E}_{\xi_i-1}[D_h(x, x^i)] - \mathbb{E}_{\xi_i}[D_h(x, x^{i+1})]. \]

Summing up the above inequality over \( i \) and noting that \( D_h(x, x^{k+1}) \geq 0 \) we arrive
at:

\[
\frac{1}{2}G^2 \sum_{i=0}^{k} t_i^2 + D_h(x,x^0) \geq \sum_{i=0}^{k} t_i \mathbb{E}_{\xi_{i-1}} [f(x^i) - f(x)] = \mathbb{E}_{\xi_{k-1}} \sum_{i=0}^{k} t_i [f(x^i) - f(x)] \]
\[
\geq \left( \sum_{i=0}^{k} t_i \right) \mathbb{E}_{\xi_{k-1}} [f(x^k) - f(x)] ,
\]

where the last inequality uses the convexity of \( f(\cdot) \). Dividing by \( \sum_{i=0}^{k} t_i \) completes the proof.

**Remark 4.4.1.** As a direct consequence of (4.21), we obtain the following result which is similar to the deterministic setting (4.5):

\[
\mathbb{E}_{\xi_{k-1}} \left[ \min_{0 \leq i \leq k} f(x^i) \right] - f(x) \leq \frac{1}{2}G^2 \sum_{i=0}^{k} t_i^2 + D_h(x,x^0) \frac{1}{\sum_{i=0}^{k} t_i} .
\]

Theorem 4.4.1 implies the following high-probability result using a simple Markov bound.

**Corollary 4.4.1.** Let \( x^* \) be an optimal solution of (4.1). Under the conditions of Theorem 4.4.1, for any \( \delta > 0 \) it holds that:

\[
\mathbb{P} \left[ f(x^k) - f^* \geq \delta \right] \leq \frac{1}{2}G^2 \sum_{i=0}^{k} t_i^2 + D_h(x^*,x^0) \frac{1}{\delta \sum_{i=0}^{k} t_i} .
\]

**Proof:** Using the Markov inequality, we have:

\[
\mathbb{P} \left[ f(x^k) - f^* \geq \delta \right] \leq \frac{\mathbb{E} \left[ f(x^k) - f^* \right]}{\delta} \leq \frac{1}{2}G^2 \sum_{i=0}^{k} t_i^2 + D_h(x^*,x^0) \frac{1}{\delta \sum_{i=0}^{k} t_i} .
\]

Similar to the case of traditional analysis of stochastic mirror descent, the Stochastic Mirror Descent Algorithm (Algorithm 9) leads to an \( O\left( \frac{1}{\varepsilon^2} \right) \) convergence guarantee (in expectation) by using an appropriate step-size sequence \( \{t_i\} \) as the next corollary shows.
Corollary 4.4.2. Under the conditions of Theorem 4.4.1, for a given $\varepsilon > 0$ suppose that the step-sizes are set to:

$$t_i := \frac{\varepsilon}{G^2}$$

for all $i$. Then within

$$k := \left\lfloor \frac{2G^2 D_h(x^*, x^0)}{\varepsilon^2} \right\rfloor - 1$$

iterations of the Stochastic Mirror Descent Algorithm it holds that:

$$\mathbb{E} [f(x^K)] - f^* \leq \varepsilon,$$

where $x^*$ is any optimal solution of (4.1).

Proof: Substituting the values of $t_i$ in (4.18) yields the result directly. \quad \square

Remark 4.4.2. Similar to the standard stochastic gradient descent scheme, the step-size rule in Corollary 4.4.2 leads to the optimal rate of convergence provided by the bound in Theorem 4.4.1.

Remark 4.4.3. Let us now compare these results to related results of Nedic and Lee [69]. In order to attain an $\varepsilon$-optimality gap, [69] proved a bound of $\left\lfloor \frac{9G^2 D_{\max}}{4\mu_h \varepsilon^2} \right\rfloor$ iterations, which follows by rearranging (4.10). In addition to not requiring Lipschitz continuity of $f(\cdot)$, our bound does not require that $h(\cdot)$ be strongly convex. We also do not require boundedness of the feasible region; and in most settings $D_h(x^*, x^0) \ll D_{\max}$ even when $D_{\max} < +\infty$. Furthermore, even in the setting of (4.10), it holds that:

$$G^2 = \mathbb{E} [\tilde{g}(x)^2|x] \cdot \max_{y \in Q, y \neq z} \frac{||y - x||^2}{2D_h(y, x)} \leq \mathbb{E} [||\tilde{g}(x)||^2_2|x] \cdot \frac{1}{\mu_h} \leq \frac{G^2}{\mu_h},$$

where the first inequality utilizes the strong convexity (in the standard sense) of $h(\cdot)$, and the second inequality is due to the assumption that $f(\cdot)$ is $G_f$-stochasticly continuous (in the standard sense). Thus we see that the bound in Corollary 4.4.2 improves on the bound in [69].
In the case when \( f(\cdot) \) is also \( \mu_{f} \)-strongly convex relative to \( h(\cdot) \) (see Definition 4.3.2), we obtain an \( O\left(\frac{1}{k}\right) \) convergence guarantee in expectation, which is also similar to the traditional case of stochastic gradient descent. This is shown in the next result.

**Theorem 4.4.2. (Convergence Bound for Stochastic Mirror Descent Algorithm under Strong Convexity relative to \( h(\cdot) \))** Consider the Stochastic Mirror Descent Algorithm (Algorithm 9). If \( f(\cdot) \) is \( G \)-stochastically-relative continuous with respect to \( h(\cdot) \) for some \( G > 0 \) and \( f(\cdot) \) is \( \mu \)-strongly convex relative to \( h(\cdot) \) for some \( \mu > 0 \), and if the step-sizes are chosen as \( t_{i} = \frac{2}{\mu(i+1)} \), then the following inequality holds for all \( k \geq 1 \):

\[
\mathbb{E}_{\xi_{k-1}} [f(\hat{x}^{k})] - f^{*} \leq \frac{2G^{2}}{\mu(k+1)},
\]

where \( \hat{x}^{k} := \frac{2}{k(k+1)} \sum_{i=0}^{k} i \cdot x^{i} \).

**Proof:** For any \( x \in Q \) it follows from the definition of \( \mu \)-strong convexity (4.15) that

\[
f(x) \geq f(x^{i}) + \langle g(x^{i}), x - x^{i} \rangle + \mu D_{h}(x, x^{i}) .
\]

Combining the above inequality with (4.19) yields

\[
f(x) \geq f(x^{i}) - \frac{1}{2}G^{2}t_{i} + \frac{1}{i} \mathbb{E}_{\gamma_{i}}[D_{h}(x, x^{i+1})|x^{i}] - (\frac{1}{i} - \mu)D_{h}(x, x^{i}) .
\]

Substituting \( t_{i} = \frac{2}{\mu(i+1)} \) and multiplying by \( i \) in the above inequality yields:

\[
i (f(x^{i}) - f(x)) \leq \frac{G^{2} i}{\mu(i+1)} + \frac{\mu}{2} (i(i - 1)D_{h}(x, x^{i}) - i(i + 1)\mathbb{E}_{\gamma_{i}}[D_{h}(x, x^{i+1})|x^{i}])
\]

\[
\leq \frac{G^{2}}{\mu} + \frac{\mu}{2} (i(i - 1)D_{h}(x, x^{i}) - i(i + 1)\mathbb{E}_{\gamma_{i}}[D_{h}(x, x^{i+1})|x^{i}]) .
\]

Taking expectation over \( \xi_{i-1} \) and summing up the above inequality over \( i \) then yields

\[
\left( \sum_{i=1}^{k} i \right) \mathbb{E}_{\xi_{k-1}} [f(\hat{x}^{k}) - f(x)] \leq \sum_{i=1}^{k} i \left( \mathbb{E}_{\xi_{i-1}} [f(x^{i})] - f(x) \right) \leq \frac{kG^{2}}{\mu} - k(k+1) \left( \frac{\mu}{2} \right) \mathbb{E}_{\xi_{k}}[D_{h}(x, x^{k+1})] \leq \frac{kG^{2}}{\mu} ,
\]
where the first inequality uses the convexity of $f(\cdot)$ and the observation that $E_{\xi_{i-1}} f(x^i) = E_{\xi_{k-1}} f(x^i)$ for $i \leq k$. Taking $x = x^*$ where $x^*$ is an optimal solution of (4.1), the proof is completed by noticing $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$.

\section*{Remark 4.4.4.} It may not be easy to find cases when the objective function is both $G$-stochastically-relative continuous and is $\mu$-relatively strongly convex relative to $h$. However, as long as these properties are satisfied along the path of iterates or around the minimum, one can achieve the faster convergence of Theorem 4.4.2.

\section*{Remark 4.4.5.} Let us also compare the computational guarantee of Theorem 4.4.2 to the results in Nedić and Lee [69]. In order to attain an $\varepsilon$-optimality gap, [69] proved the bound (4.11). First notice that we do not require either that $f(\cdot)$ is uniformly Lipschitz continuous or that $h(\cdot)$ is strongly convex in the traditional sense, or that $h$ is uniformly smooth. However, even if these requirements hold, it follows from Remark 4.4.3 that $G^2 \leq \frac{C_j}{\mu_h}$, and it also holds that:

\[ D_f(x, y) \geq \frac{\mu_f}{2} \|x - y\|^2 \geq \frac{\mu_f}{L_h} D_h(x, y), \]

where the first inequality utilizes that $f(\cdot)$ is $\mu_f$ strongly convex and the second inequality $h(\cdot)$ is $L_h$ smooth in the standard sense. Thus $f(\cdot)$ is at least $\mu = \frac{\mu_f}{L_h}$-strongly convex relative to $h(\cdot)$ (this follows by applying Proposition 1.1 in [61]). Therefore, even under the stronger requirements of [69], Theorem 4.4.2 improves on the corresponding result in [69].

We end this section with a discussion of the deterministic setting, namely the (Deterministic) Mirror Descent Algorithm (Algorithm 8). Suppose that there is no stochasticity in the computation of subgradients. We can cast this as an instance of the Stochastic Mirror Descent Algorithm (Algorithm 9) wherein $\tilde{g}(x) = g(x) \in \partial f(x)$ for all $x \in Q$. In this case relative stochastic continuity (Definition 4.3.3) is equivalent to relative continuity (Definition 4.3.1) with the same constant. Thus deterministic Mirror Descent is a special case of Stochastic Mirror Descent, and we have the following computational guarantees as special cases of the stochastic case.

\section*{Theorem 4.4.3. (Convergence Bound for Deterministic Mirror Descent Algorithm)} Consider the (Deterministic) Mirror Descent Algorithm (Algorithm 8). If $f(\cdot)$ is
$M$-relative continuous with respect to $h(\cdot)$ for some $M > 0$, then for all $k \geq 1$ and $x \in Q$ the following inequality holds:

$$f(\bar{x}^k) - f(x) \leq \frac{\frac{1}{2}M^2 \sum_{i=0}^{k} t_i^2 + D_h(x, x^0)}{\sum_{i=0}^{k} t_i},$$

where $\bar{x}^k := \frac{1}{\sum_{i=0}^{k} t_i} \sum_{i=0}^{k} t_i x^i$.

\[\square\]

**Corollary 4.4.3.** Under the conditions of Theorem 4.4.3, for a given $\varepsilon > 0$ suppose that the step-sizes are set to:

$$t_i := \frac{\varepsilon}{M^2}$$

for all $i$. Then within

$$k := \left\lceil \frac{2M^2 D_h(x^*, x^0)}{\varepsilon^2} \right\rceil - 1$$

iterations of Deterministic Mirror Descent it holds that:

$$f(\bar{x}^k) - f^* \leq \varepsilon,$$

where $\bar{x}^k := \frac{1}{\sum_{i=0}^{k} t_i} \sum_{i=0}^{k} t_i x^i$, and $x^*$ is any optimal solution of (4.1).

\[\square\]

**Theorem 4.4.4. (Convergence Bounds for Deterministic Mirror Descent with Strong Relative Convexity)** Consider the Deterministic Mirror Descent Algorithm (Algorithm 8). If $f(\cdot)$ is $M$-relative continuous with respect to $h(\cdot)$ for some $M > 0$ and $f(\cdot)$ is $\mu$-strongly convex relative to $h(\cdot)$ for some $\mu > 0$, and if the step-sizes are chosen as $t_i = \frac{2}{\mu(i+1)}$, then the following inequality holds for all $k \geq 1$:

$$f(\bar{x}^k) - f^* \leq \frac{2M^2}{\mu(k+1)},$$

where $\bar{x}^k := \frac{2}{k(k+1)} \sum_{i=0}^{k} i \cdot x^i$.

\[\square\]
4.5 Specifying a Reference Function $h(\cdot)$ with Relative Continuity for Mirror Descent

Let us discuss using either deterministic or stochastic Mirror Descent (Algorithm 8 or Algorithm 9) for solving the optimization problem (4.1) with objective function $f(\cdot)$ that is $M$-relative continuous or $G$-stochastically-relative continuous (respectively) with respect to the reference function $h(\cdot)$. In order to efficiently execute the update step in Algorithm 8 and/or Algorithm 9 we need $h(\cdot)$ to be such that the linearization subproblem LS (4.7) is efficiently solvable for any given $c$. Therefore, in order execute Mirror Descent for solving (4.1) using Algorithm 8 or Algorithm 9, we need to specify a differentiable convex reference function $h(\cdot)$ that has the following two properties:

(i) $f(\cdot)$ is $M$-relative continuous (or $G$-stochastically-relative continuous) with respect to $h(\cdot)$ on $Q$ for $M$ (or $G$) that is easy to determine, and

(ii) the linearization subproblem LS (4.7) has a solution, and the solution is efficiently computable.

We now discuss quite broadly how to construct such a reference function $h(\cdot)$ with these two properties when $\|g(x)\|^2$ is bounded by a polynomial in $\|x\|^2$.

4.5.1 Deterministic Setting

Suppose that $\|g(x)\|^2 \leq p_r(\|x\|^2)$ for all $x \in Q$ and all $g(x) \in \partial f(x)$, where $p_r(\alpha) = \sum_{i=0}^{r} a_i \alpha^i$ is an $r$-degree polynomial of $\alpha$ whose coefficients $\{a_i\}$ are nonnegative. Let

$$ h(x) := \sum_{i=0}^{r} \frac{a_i}{i+2} \|x\|_{l_2}^{i+2}. $$

Then the following proposition states that $f(\cdot)$ is 1-relative continuous with respect to $h(\cdot)$. This implies that no matter how fast the subgradient of $f(\cdot)$ grows polynomially as $\|x\|_2 \to \infty$, $f(\cdot)$ is relatively continuous with respect to the simple reference function $h(\cdot)$, even though $f$ does not exhibit uniform Lipschitz continuity.
Proposition 4.5.1. \( f(\cdot) \) is 1-continuous relative to \( h(x) = \sum_{i=0}^{r} a_i \|x\|_{i+2}^i \).

Proof: Let \( h_i(x) = \frac{1}{i+2} \|x\|_{i+2}^i \), then \( h(x) = \sum_{i=0}^{r} a_i h_i(x) \), and by the definition of Bregman distance, we have

\[
D_{h_i}(y, x) = \frac{1}{i+2} \|y\|_{i+2}^i - \frac{1}{i+2} \|x\|_{i+2}^i - \langle \|x\|_{i+2}^i, y - x \rangle
\]

\[
= \frac{1}{i+2} (\|y\|_{i+2}^i + (i + 1)\|x\|_{i+2}^i - (i + 2)\|x\|_{i+2}^i \langle x, y \rangle).
\]

Notice that

\[
\|y\|_{i+2}^i + (i + 1)\|x\|_{i+2}^i - (i + 2)\|x\|_{i+2}^i \langle x, y \rangle
\]

\[
= (\|y\|_{i+2}^i + \frac{1}{2}\|x\|_{i+2}^i - \frac{i+2}{2}\|x\|_{i+2}^i \|y\|_{i+2}^i) + \frac{i+2}{2}\|x\|_{i+2}^i (\|x\|_{i+2}^i + \|y\|_{i+2}^i - 2 \langle x, y \rangle)
\]

\[
\geq 0 + \frac{i+2}{2}\|x\|_{i+2}^i (\|x\|_{i+2}^i + \|y\|_{i+2}^i - 2 \langle x, y \rangle)
\]

\[
= \frac{i+2}{2}\|x\|_{i+2}^i \|y - x\|_{i+2}^i,
\]

where the inequality above is an application of arithmetic-geometric mean inequality \( a^\lambda b^{1-\lambda} \leq \lambda a + (1 - \lambda)b \) with \( a = \|x\|_{i+2}^i, b = \|y\|_{i+2}^i \), and \( \lambda = \frac{i}{i+2} \). Thus we have

\[
D_{h}(y, x) = \sum_{i=0}^{r} a_i D_{h_i}(y, x) \geq \frac{1}{2} \|y - x\|_{i+2}^i \left( \sum_{i=0}^{r} a_i \|x\|_{i+2}^i \right).
\]

Therefore

\[
\|g(x)\|_{i+2}^i \leq p_i(\|x\|_{i+2}^i) = \sum_{i=0}^{r} a_i \|x\|_{i+2}^i \leq \frac{D_{h}(y, x)}{\frac{1}{2} \|y - x\|_{i+2}^i},
\]

which shows that \( f(\cdot) \) is 1-relative continuous with respect to \( h(\cdot) \).

Solving the linearization subproblem (4.7). Let us see how we can solve the linearization subproblem (4.7) for this class of optimization problems. The linearization subproblem (4.7) can be written as

\[
\text{LS} : \min_{x \in \mathbb{R}^n} \langle c, x \rangle + \sum_{i=0}^{r} \frac{a_i}{i+2} \|x\|_{i+2}^i.
\]
and the first-order optimality condition is simply:

$$c + \left( \sum_{i=0}^{r} a_i \|x\|_2^i \right) x = 0,$$

whereby $x = -\theta c$ for some scalar $\theta \geq 0$, and it remains to simply determine the value of the nonnegative scalar $\theta$. In the case when $c = 0$ we have $x = 0$ satisfies (4.24), so let us examine the case when $c \neq 0$, in which case from (4.24) $\theta$ must satisfy:

$$\sum_{i=0}^{r} a_i \|c\|_2^{i+1} - 1 = 0,$$

which implies that $\theta$ is the unique positive root of a univariate polynomial monotone in $\theta \geq 0$. For $r \in \{0, 1, 2, 3\}$ this root can be computed in closed form. Otherwise the root can be computed efficiently (up to machine precision) using any suitable root-finding method.

**Remark 4.5.1.** We can incorporate a simple set constraint $x \in Q$ in problem (4.23) provided that we can easily compute the Euclidean projection on $Q$. In this case, the linearization subproblem (4.7) can be converted to a 1-dimensional convex optimization problem, see Appendix A.1 of [61] for details.

### 4.5.2 Stochastic Setting

In the stochastic setting, the stochastic subgradient $\hat{g}(x)$ is a conditional random variable for a given $x$. Suppose that $E[\|\hat{g}(x)\|_2^2|x] \leq p_r(\|x\|_2)$ for all $x \in Q$, where $p_r(\alpha) = \sum_{i=0}^{r} a_i \alpha^i$ is an $r$-degree polynomial whose coefficients $\{a_i\}$ are nonnegative. Let

$$h(x) := \sum_{i=0}^{r} \frac{a_i}{i+2} \|x\|_2^{i+2},$$

and similar to the deterministic case we have:

**Proposition 4.5.2.** $f(\cdot)$ is 1-stochastically continuous relative to $h(x) = \sum_{i=0}^{r} \frac{a_i}{i+2} \|x\|_2^{i+2}$.

**Proof:** For any $x, y \in Q$ with $x \neq y$ we have:

$$E[\|\hat{g}(x)\|_2^2|x] \leq p_r(\|x\|_2) = \sum_{i=0}^{r} a_i \|x\|_2^i \leq \frac{2D_h(y, x)}{\|y - x\|_2^2},$$

(4.25)
where the last inequality follows from (4.22), and thus \( f(\cdot) \) is 1-stochastically continuous relative to \( h(\cdot) \).

\[ \square \]

**Solving the linearization subproblem** (4.7). The linear optimization subproblem is identical in structure to that in the deterministic case and so can be solved as discussed at the end of Section 4.5.1.

### 4.5.3 Relative Continuity for instances of SVM and IEP

Here we examine in detail the two motivating examples stated in the Introduction, namely the Support Vector Machine (SVM) problem, and the Intersection of Ellipsoids Problem (IEP). We first prove the following lemma, which presents upper bounds on the Bregman distances \( D_h(\cdot, \cdot) \) for \( h(x) = \|x\|^3_2 \) and \( h(x) = \|x\|^4_2 \).

**Lemma 4.5.1.**

1. Let \( h(x) := \|x\|^3_2 \). Then \( D_h(y, x) \leq \frac{1}{3} \|y - x\|^2_2 (\|y\|_2^2 + 2\|x\|_2) \).

2. Let \( h(x) := \|x\|^4_2 \). Then \( D_h(y, x) \leq \frac{1}{4} \|y - x\|^2_2 (\|y + x\|^2_2 + 2\|x\|^2_2) \).

**Proof:**

1.

\[
D_h(y, x) = \frac{1}{3} (\|y\|^3_2 + 2\|x\|^3_2 - 3\|x\|_2(x, y))
\]

\[
\leq \frac{1}{3} (\|y\|^3_2 + 2\|x\|^3_2 - 3\|x\|_2(x, y) - 2\|y\|_2(x, y) + 2\|y\|^2_2\|x\|_2 - \|x\|_2(x, y) + \|y\|_2\|x\|^3_2)
\]

\[
= \frac{1}{3} \|y - x\|^2_2 (\|y\|_2^2 + 2\|x\|_2)
\]

where the first equality follows from simplifying and combining terms, the inequality follows from applying the Cauchy-Schwarz inequality twice, and the final equality is from simplifying and combining terms.
2. 
\[ D_h(y, x) = \frac{1}{4} \left( \Vert y \Vert^2 + 3 \Vert x \Vert^2 - 4 \Vert x \Vert^2 \langle x, y \rangle \right) \]

\[ \leq \frac{1}{4} \left( \Vert y \Vert^2 + 3 \Vert x \Vert^2 - 4 \Vert x \Vert^2 \langle x, y \rangle + 4 \Vert x \Vert^2 \Vert y \Vert^2 - 4 \langle x, y \rangle^2 \right) \]

\[ = \frac{1}{4} \Vert y - x \Vert^2 \left( \Vert y \Vert^2 + 2 \Vert x \Vert^2 \right), \]

where the first equality follows from simplifying and combining terms, the inequality follows from applying the Cauchy-Schwarz inequality once, and the final equality is from simplifying and combining terms.

The **Support Vector Machine (SVM)**. The Support Vector Machine (SVM) is an important supervised learning model for binary classification in machine learning. The SVM optimization problem for binary classification is:

\[
\text{SVM: } \min_x f(x) := \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - y_i x^T w_i\} + \frac{1}{2} \|x\|^2, \quad (4.26)
\]

where \( w_i \) is the input feature vector of sample \( i \) and \( y_i \in \{-1, 1\} \) is the label of sample \( i \). Notice that \( f(\cdot) \) is not differentiable due to the presence of the hinge loss terms in the summation, and \( f(\cdot) \) is also not Lipschitz continuous due to the presence of the \( \ell_2 \)-norm regularization term; thus we cannot directly utilize typical subgradient or gradient schemes in the analysis of \((4.26)\). Researchers have developed various approaches to overcome this limitation. For example, [31] introduced a splitting subgradient-type method, where the basic idea is to split the loss function and the regularization terms. [113] introduced a quasi-Newton method, where they do not need to worry about the unbounded subgradient. Another approach is to a priori constrain \( x \) to lie in an \( \ell_2 \)-ball of radius \( R \) for \( R \) sufficiently large so that the ball contains the optimal solution, and to project onto this ball at each iteration; in this approach \( f(\cdot) \) is Lipschitz continuous in the amended feasible region, see [94]. Indeed, one can show using quadratic optimization optimality conditions that it suffices to set \( R = \min\{ \frac{1}{\lambda} \left( \frac{1}{n} \sum_{i=1}^n \| w_i \|_2 \right), \sqrt{2/\lambda} \} \) (see Appendix B.4) wherein the modulus of Lipschitz continuity in the amended feasible region is at most \( M \leq \frac{1}{n} \sum_{i=1}^n \| w_i \|_2 + \min\{ \sqrt{2\lambda}, \frac{1}{n} \sum_{i=1}^n \| w_i \|_2 \} \). Furthermore, in [57] the authors show that if the initial point lies within a suitably chosen large ball, then Stochastic
Subgradient Descent with a small step-size ensures in expectation that all iterates lie in the large ball, which then ensures that the norms of all subgradients are bounded in expectation.

Let us see how we can directly use the constructs of relative continuity to tackle the SVM problem with a suitably designed version of Stochastic Mirror Descent – without any projection step to a ball. We can rewrite the objective function of (4.26) as

\[ f(x) = \frac{1}{n} \sum_{j=1}^{n} f_j(x), \]

where \( f_j(x) = \max\{0, 1 - y_j x^T w_j\} + \frac{\lambda}{2} \|x\|^2. \) We consider computing a stochastic estimate of the subgradient of \( f(.) \) by using a single sample index \( \tilde{j} \) drawn randomly from \( \{1, \ldots, n\} \), namely \( \tilde{g}(x) \in \partial f_{\tilde{j}}(x) \) where \( \tilde{j} \) is drawn uniformly at random from \( \{1, \ldots, n\} \). Then \( \|\tilde{g}(x)\|^2 \leq (\lambda \|x\|_2 + \|w_j\|_2)^2 \), whereby

\[ \mathbb{E}[\|\tilde{g}(x)\|_2^2] \leq \lambda^2 \|x\|_2^2 + \frac{2\lambda}{n} \left( \sum_{i=1}^{n} \|w_i\|_2 \right) \|x\|_2 + \frac{1}{n} \sum_{i=1}^{n} \|w_i\|_2^2, \]

and notice that the right-hand side is a polynomial in \( \|x\|_2 \) of degree \( r = 2 \). If we choose the reference function \( h(.) \) as

\[ h(x) := \frac{\lambda^2}{4} \|x\|_2^2 + \frac{2\lambda}{3n} \left( \sum_{i=1}^{n} \|w_i\|_2 \right) \|x\|_2^3 + \frac{1}{2n} \left( \sum_{i=1}^{n} \|w_i\|_2^2 \right) \|x\|_2^2, \quad (4.27) \]

it follows from the Proposition 4.5.2 that \( f(.) \) is 1-stochastically continuous relative to \( h(x) \).

**Proposition 4.5.3. (Computational Guarantees for Stochastic Mirror Descent for the SVM problem (4.26).)** Consider applying the Stochastic Mirror Descent algorithm (Algorithm 9) to the Support Vector Machine problem (4.26) using the reference function (4.27). For an absolute optimality tolerance value \( \varepsilon > 0 \), and using the constant step-sizes \( t_i := \varepsilon \), let the algorithm be run for

\[ k := \left[ \|x^* - x^0\|^2 \left( 3\lambda^2 \left( \|x^* + x^0\|_2^3 + 2\|x^0\|_2^2 \right) + \frac{6\lambda}{n} \left( \sum_{i=1}^{n} \|w_i\|_2 \right) \left( \|x^*\|_2 + 2\|x^0\|_2 \right) + \frac{6}{n} \left( \sum_{i=1}^{n} \|w_i\|_2^2 \right) \right]^{-1} \]

iterations, where \( x^* \) is the optimal solution of (4.26). Then it holds that

\[ \mathbb{E} \left[ f(x^k) - f^* \right] \leq \varepsilon, \]
where $x_k^* := \frac{1}{k+1} \sum_{i=0}^k x_i$.

**Proof:** We showed above (using Proposition 4.5.2) that $f(\cdot)$ is 1-stochastically continuous relative to $h(\cdot)$ defined in (4.27). Furthermore, applying Lemma 4.5.1 it follows that

$$D_h(x^*, x^0) \leq \frac{\lambda^2}{4} \|x^* - x^0\|_2 (\|x^* + x^0\|_2^2 + 2\|x^0\|_2^2 + \frac{2\lambda}{3n} \left( \sum_{i=1}^n \|w_i\|_2 \right) (\|x^*\|_2 + 2\|x^0\|_2) + \frac{1}{2n} \left( \sum_{i=1}^n \|w_i\|_2^2 \right) \|x^* - x^0\|_2^2.$$ 

The proof is finished by substituting these values into the computational guarantee of Corollary 4.4.2. 

**Intersection of Ellipsoids Problem (IEP).** Consider the problem of computing a point $x \in \mathbb{R}^m$ in the intersection of $n$ ellipsoids, namely:

$$x \in \mathcal{Q} := \mathcal{Q}_1 \cap \mathcal{Q}_2 \cap \cdots \cap \mathcal{Q}_n ,$$

where $\mathcal{Q}_i = \{ x \in \mathbb{R}^m : \frac{1}{2} x^T A_i x + b_i x + c_i \leq 0 \}$ and $A_i \in \mathbb{R}^{m \times m}$ is a given symmetric positive semi-definite matrix, $i = 1, \ldots, n$. This problem can be cast as a second-order cone optimization problem, and hence can be tackled using interior-point methods. However, interior-point methods are typically only effective when the dimensions $m$ and/or $n$ are of moderate size. On the other hand, another way to tackle the problem is to use a first-order method to solve the unconstrained problem

$$\text{IEP: } f^* = \min_x f(x) := \max_{0 \leq i \leq n} \{ \frac{1}{2} x^T A_i x + b_i^T x + c_i \} ,$$

and notice that $f(x) \leq 0 \iff x \in \mathcal{Q}$, and $\mathcal{Q} \neq \emptyset \iff f^* \leq 0$. However, the objective function $f(\cdot)$ in (4.29) is both non-differentiable and non-Lipschitz, and so it falls outside of the scope of optimization problems for which traditional first-order methods are applicable. Let us see how we can use the machinery of relative continuity to tackle this problem. Let $\sigma := \max_{0 \leq i \leq n} \|A_i\|_2^2$ where $\|A_i\|_2$ is the spectral radius of $A_i$, let $\rho := 2 \max_{0 \leq i \leq n} \|A_i b_i\|_2$ and let $\gamma := \max_{0 \leq i \leq n} \|b_i\|_2^2$. Notice that for any $x$ and $i = 1, \ldots, n$, we have $g_i(x) := A_i x + b_i = \nabla f_i(x)$ where $f_i(x)$ is the $i^{th}$ term in the objective function of (4.29). Since $g(x) \in \partial f(x)$ if and only if $g(x)$ is a convex

---

\footnote{This problem was suggested by Nesterov [72].}
combination of the active gradients $\nabla f_i(x)$ (see Danskin’s Theorem, Proposition B.22 in [11]), it follows for any $g(x) \in \partial f(x)$ that

$$\|g(x)\|_2^2 \leq \max_{0 \leq i \leq n} \|A_i x + b_i\|_2^2 \leq \max_{0 \leq i \leq n} \|A_i\|_2^2 \|x\|_2^2 + 2\|b_i^T A_i\|_2 \|x\|_2 + \|b_i\|_2^2 \leq \sigma \|x\|_2^2 + \rho \|x\|_2 + \gamma.$$ 

Therefore we have $\|g(x)\|_2^2 \leq p_2(\|x\|_2)$, where $p_1(\alpha) = \sigma \alpha^2 + \rho \alpha + \gamma$ is a quadratic function of $\alpha$, which is a polynomial in $\alpha$ of degree $r = 2$. It follows from Proposition 4.5.1 that $f(\cdot)$ is 1-continuous relative to the reference function

$$h(x) := \frac{\sigma}{4} \|x\|_2^4 + \frac{\rho}{3} \|x\|_2^3 + \frac{\gamma}{2} \|x\|_2^2.$$ (4.30)

**Proposition 4.5.4. (Computational Guarantees for Deterministic Mirror Descent for the IEP problem (4.29)).** Consider applying the Deterministic Mirror Descent algorithm (Algorithm 8) to the Ellipsoid Intersection Problem (4.29) using the reference function (4.30), where $\sigma := \max_{0 \leq i \leq n} \|A_i\|_2^2$ and $\|A_i\|_2$ is the spectral radius of $A_i$, $\rho := 2 \max_{0 \leq i \leq n} \|A_i b_i\|_2$ and $\gamma := \max_{0 \leq i \leq n} \|b_i\|_2^2$. For an absolute optimality tolerance value $\varepsilon > 0$, and using the constant step-sizes $t_i := \varepsilon$, let the algorithm be run for

$$k := \left\lfloor \frac{\|x^* - x^0\|_2^2 (3\sigma (\|x^* + x^0\|_2^2 + 2\|x^0\|_2^2) + 4\rho (\|x^*\|_2 + 2\|x^0\|_2) + 6\gamma)}{6\varepsilon^2} \right\rfloor - 1$$ iterations, where $x^*$ is any optimal solution of (4.29). Then it holds that

$$f(\bar{x}^k) - f^* \leq \varepsilon,$$

where $\bar{x}^k := \frac{1}{k+1} \sum_{i=0}^k x_i$.

**Proof:** We showed above (using Proposition 4.5.1) that $f(\cdot)$ is 1-continuous relative to $h(x) = \frac{\sigma}{4} \|x\|_2^4 + \frac{\rho}{3} \|x\|_2^3 + \frac{\gamma}{2} \|x\|_2^2$. Furthermore, applying Lemma 4.5.1 it follows that $D_h(x^*, x^0) \leq \frac{\sigma}{4} \|x^* - x^0\|_2^2 (\|x^* + x^0\|_2^2 + 2\|x^0\|_2^2) + \frac{\rho}{3} \|x^* - x^0\|_2^2 (\|x^*\|_2 + 2\|x^0\|_2) + \frac{\gamma}{2} \|x^* - x^0\|_2^2$. The proof is finished by substituting these values into the computational guarantee of Corollary 4.4.3. \qed
Chapter 5

Randomized Gradient Boosting Machine

5.1 Introduction

Gradient Boosting Machine (GBM) [41] is a powerful supervised learning algorithm that combines multiple weak-learners into an ensemble with excellent predictive performance. It works very well in several prediction tasks arising in spam filtering, online advertising, fraud detection, anomaly detection, computational physics (e.g., the Higgs Boson discovery), etc; and has routinely featured as a top algorithm in Kaggle competitions and the KDDCup [27]. GBM can naturally handle heterogeneous datasets (highly correlated data, missing data, categorical data, etc) and leads to interpretable models by building an additive model [40]. It is also quite easy to use with several publicly available implementations: scikit-learn, gbm, Spark MLLib, LightGBM, XGBoost, TF Boosted Trees, etc [27],[88].

In spite of the usefulness of GBM in practice, there is a considerable gap between its theoretical understanding and its success in practice. The traditional interpretation of GBM is to view it as a form of steepest descent on a certain functional space [41]. While this viewpoint serves as a good starting point, the framework lacks
rigorous computational guarantees, especially, when compared to the growing body of literature in first order convex optimization. There has been some work on deriving convergence rates of GBM—see for example [102][35][13][68], and our discussion in Section 5.1.3. Moreover, there are many heuristics employed by practical implementations of GBM that work well in practice—for example, the constant step-size rule and column sub-sampling mechanism implemented in XGBoost [27], but a formal explanation of these heuristics seems to be lacking in the current literature. This prevents us from systematically addressing important (tuning) parameter choices that may be informed by answers to questions like: how might one choose an optimal step-size, how many weak-learners should one subsample, etc? Addressing these concerns is a goal of this chapter. In this work we build a methodological framework for understanding GBM and Randomized Gradient Boosting Machine (RGBM), introduced herein, by using tools from convex optimization. Our hope is to narrow the gap between the theory and practice of GBM and its randomized variants. Below, we revisit the classical GBM framework and then introduce RGBM.

5.1.1 Gradient Boosting Machine

We consider a supervised learning problem [46], with \( n \) training examples \((x_i, y_i), i = 1, \ldots, n\) such that \( x_i \in \mathbb{R}^p \) is the feature vector of the \( i \)-th example and \( y_i \in \mathbb{R} \) is a label (in a classification problem) or a continuous response (in a regression problem). In the classical version of GBM [41], the prediction corresponding to a feature vector \( x \) is given by an additive model of the form:

\[
f(x) := \sum_{m=1}^{M} \beta_{jm} b(x; \tau_{jm}) ,
\]

where each basis function \( b(x; \tau) \in \mathbb{R} \) (also called a weak-learner) is a simple function of the feature vector indexed by a parameter \( \tau \), and \( \beta_j \) is the coefficient of the \( j \)-th weak-learner. Here, \( \beta_{jm} \) and \( \tau_{jm} \) are chosen in an adaptive fashion to as to improve the data-fidelity (according to a certain rule) as discussed below. Examples
of weak-learners commonly used in practice [46] include wavelet functions, support
vector machines, tree stumps (i.e, decision trees of depth one) and classification and
regression trees (CART) [16], etc. We assume here that the set of weak-learners
is finite with cardinality $K$—in many of the examples alluded to above, $K$ can be
exponentially large, thereby leading to computational challenges.

The goal of GBM is to obtain a good estimate of the function $f$ that approximately
minimizes the empirical loss:

$$
\min_f \sum_{i=1}^{n} \ell(y_i, f(x_i)),
$$

(5.2)

where $\ell(y_i, f(x_i))$ is a measure of the data-fidelity for the $i$th sample for the loss
function $\ell$, which is assumed to be differentiable in the second coordinate. The
original version of GBM [41] (presented in Algorithm 13) can be viewed as applying
a steepest descent algorithm to minimize the loss function (6.2). GBM starts from a
null model $f \equiv 0$ and in each iteration, we compute a weak-learner that best fits the
current pseudo-residual (namely, the negative gradient of the loss function over the
prediction) $r^m$, in terms of the least squares loss as follows:

$$
\hat{j}_m = \arg\min_{j \in [K]} \min_{\sigma} \sum_{i=1}^{n} (r^m_i - \sigma b(x_i; \tau_j))^2,
$$

(5.3)

where $[K]$ is a shorthand for the set $\{1, \ldots, K\}$. (In case of ties in the “argmin”
operation, we choose the one with the smallest index — this convention is used
throughout the chapter.) We then add the $j^\text{th}$ weak-learner into the model by a line
search. As the iterations progress, GBM leads to a sequence of models $\{f^m\}_{m \in [M]}
(indexed by the number of GBM iterations), where each model corresponds to a
certain data-fidelity and complexity/shrinkage [41, 35]—together they control the
out-of-sample performance of the model. The usual intention of GBM is to stop
early—before one is close to a minimum of Problem (6.4)—with the hope that such
a model will lead to good predictive performance [41, 35, 116].
Algorithm 10 Gradient Boosting Machine (GBM) [41]

**Initialization.** Initialize with $f^0(x) = 0$.

For $m = 0, \ldots, M - 1$ do:

**Perform Updates:**

1. Compute pseudo residual $r_m = -\left[ \frac{\partial \ell(y_i, f^m(x_i))}{\partial f^m(x_i)} \right]_{i=1}^n$.
2. Find the best weak-learner: $j_m = \arg \min_{j \in [K]} \min_{\rho} \sum_{i=1}^n (r_i^m - \sigma b(x_i; \tau_j))^2$.
3. Choose the step-size $\rho_m$ by line-search: $\rho_m = \arg \min_{\rho} \sum_{i=1}^n (y_i - f^m(x_i) + \rho b(x_i; \tau_{jm}))$.
4. Update the model $f^{m+1}(x) = f^m(x) + \rho_m b(x; \tau_{jm})$.

**Output.** $f^M(x)$.

Note that since we perform a line-search, rescaling the prediction vector $[b(x_i; \tau_j)]_i$ does not change the output of Algorithm 13. Hence, without loss of generality, we assume that the prediction vector is normalized throughout the chapter.

**Assumption 5.1.1.** The prediction vector of each weak-learner is normalized, namely for every $\tau$

$$\sum_{i=1}^n b(x_i; \tau)^2 = 1.$$  

5.1.2 Randomized Gradient Boosting Machine

In GBM, finding the best weak-learner (step (2) in Algorithm 13) is the most expensive step. For example, when the weak-learners are CART trees of depth $d$, finding the best weak-learner requires one to go over $O(n^{d-1} p^{d-1})$ possible tree splits—this is computationally intractable for medium scale problems, even when $d = 1$.

It seems natural to use a randomization scheme to reduce the cost associated with step (2) in Algorithm 13. To this end, we propose RGBM (see Algorithm 11), where the basic idea is to use a randomized approximation for step (6.4). To be more specific, in each iteration of RGBM, we randomly pick a small subset of weak-learners $J$ by some rule (see below) and then choose the best candidate from within $J$:

$$j_m = \arg \min_{j \in J} \min_{\sigma} \sum_{i=1}^n (r_i^m - \sigma b(x_i; \tau_j))^2. \tag{5.4}$$
If we set $|J|$ to be much smaller than the total number of weak-learners $K$, the cost per iteration in RGBM will be much cheaper than GBM. We note that the implementation of XGBoost utilizes a related heuristic (called column subsampling) [27], which has been seen to be work well in practice. However, to our knowledge, we are not aware of any prior work that formally introduces and studies the RGBM algorithm — a task that we wish to accomplish in this chapter.

Note that the type of selection rule we are advocating in RGBM is different from that employed in the well-known Stochastic Gradient Boosting framework by Friedman [42], in which Friedman introduced a procedure that randomly selects a subset of the training examples to fit a weak-learner at each iteration. In contrast, we randomly choose a subset of weak-learners in RGBM. Indeed, both feature and sample sub-sampling are applied in the context of random forests [58], however, we remind the reader that random forests are quite different from GBM.

Algorithm 11 Randomized Gradient Boosting Machine (RGBM)

**Initialization.** Initialize with $f^0(x) = 0$. For $m = 0, \ldots, M - 1$ do:

**Perform Updates.**

1. Compute pseudo residuals $r_m = - \left[ \frac{\partial y_i f^m(x_i)}{\partial f^m(x_i)} \right]_{i=1}^{n}$.
2. Pick a random subset $J$ of weak-learners by some rule (i.e., one of Type 0 - Type 3)
3. Find the best weak-learner in $J$: $j_m = \arg \min_{j \in J} \min_{\sigma} \sum_{i=1}^{n} (r^m_i - \sigma b(x_i; \tau_j))^2$.
4. Choose the step-size $\rho_m$ by one of the following rules:
   - line-search: $\rho_m = \arg \min_{\rho} \sum_{i=1}^{n} \ell(y_i, f^m(x_i) + \rho b(x_i; \tau_{jm}))$;
   - constant step-size: $\rho_m = \rho (\sum_{i=1}^{n} r^m_i b(x_i; \tau_{jm}))$, where $\rho$ is a constant specified a-priori.
5. Update the model $f^{m+1}(x) = f^m(x) + \rho_m b(x; \tau_{jm})$.

**Output.** $f^M(x)$.

Random Selection Rules for Choosing $J$: We present a set of selection rules to choose $J$:

[Type 0]: we choose $J$ as the whole set of weak-learners. This is a deterministic selection rule.
we choose uniformly at random \( t \) weak-learners from all possible weak-learners without replacement.

**[Type 2]**: given a non-overlapping partition of the weak-learners, we pick one group uniformly at random and let the weak-learners in that group be \( J \).

**[Type 3]**: given a non-overlapping partition of the weak-learners, we pick \( t \) groups uniformly at random and let the collection of weak-learners across these groups be \( J \).

**Remark 5.1.1.** RGBM with Type 0 selection rule recovers GBM.

We present an example to illustrate the different selection rules introduced above.

**Example.** We consider GBM with decision stumps for a binary classification problem. Recall that a decision stump is a decision tree [46] with a depth of one. The parameter \( \tau \) of a decision stump contains two items: which feature to split and what value to split on. More specifically, a weak-learner characterized by \( \tau = (g, s) \) for \( g \in [p] \) and \( s \in \mathbb{R} \) is given by (up to a sign change)

\[
b(x; \tau = (g, s)) = \begin{cases} 
1 & \text{if } x_g \leq s, \\
-1 & \text{if } x_g > s.
\end{cases}
\]  

(5.5)

Notice that for a given feature \( x_g \) and \( n \) training samples, there are at most \( n \) different values for \( s \) (and equality holds when the feature values are all distinct). This leads to \( K = np \) many tree stumps \( \{b(x; \tau)\}_\tau \) indexed by \( \tau \). For the Type 0 selection rule, we set \( J \) to be the collection of all \( np \) tree stumps, in a deterministic fashion. As an example of Type 1 selection rule, \( J \) can be a collection of \( t \) tree stumps selected randomly without replacement from all of \( np \) tree stumps. Let \( I_g \) be a group comprising of all tree stumps that split on feature \( x_g \) — i.e., \( I_g = \{(g, s) \mid s\} \) for every \( g \in [p] \). Then \( \{I_g\}_{g \in [p]} \) defines a partition of all possible tree stumps. Given such a partition, an example of the Type 2 selection rule is: we randomly choose \( g \in [p] \) and set \( J = I_g \). Instead, one can also pick \( t \) (out of \( p \)) features randomly and choose all
nt tree stumps on those \( t \) features as the set \( J \)—this leads to an instance of the Type 3 selection rule. Note that a special case of Type 3 with \( t = 1 \) is the Type 2 selection rule.

![Figure 5-1](image)

**Figure 5-1:** Plots showing the training [top panel] and testing [bottom panel] loss versus number of RGBM iterations and the associated computational cost (measured by “epochs”) for RGBM with different \( t \) values. We consider the a9a dataset (for a classification task) from the LIBSVM library (see text for details). A smaller value of \( t \) corresponds to a smaller cost per iteration. As expected we see overall computational savings for a value of \( t \) that is smaller than the maximum \( t = 123 \), which corresponds to GBM.

For motivation, we illustrate our key idea with a real-data example. Figure 5-1 shows the computational gains of RGBM for solving a binary classification problem with decision stumps. Here we use the Type 3 selection rule (as described above), where each group represents all tree stumps splitting on a single feature, and \( G = 123 \) is the total number of groups. Different lines correspond to different \( t \) values, namely how many groups appear in the random set \( J \) in each iteration. The blue line corresponds to GBM (Algorithm 13) as it uses all the groups. The major computational
cost stems from computing the best weak-learner from a subset of weak-learners. "Epochs" is a counter that keeps track of the number of times we go over all weak-learners—this is a proxy for the total computational cost. The implementation details can be found in Section 5.5. The left column of Figure 5-1 presents the training and testing loss versus number of iterations (See Section 5.5 for details). We can see that when the number of groups $t$ gets smaller, we may get less improvement (in training loss) per iteration, but not by a large margin (for example, the case $t = 24$ has a similar behavior as the case $t = 123$). The right column of Figure 5-1 shows the training/testing loss versus epochs. We can see that with a smaller $t$, the cost per iteration decreases dramatically and overall, one requires fewer passes over all weak-learners to achieve a similar training/testing error.

5.1.3  Related Literature on Convergence Guarantees for GBM

The development of the general convergence guarantees for GBM has seen several milestones in the past decade. After being proposed by [41], Collins et al [28] showed the convergence of GBM, without any rates. Bickel and Ritov [13] proved an exponential convergence rate (more precisely $O(\exp(1/\epsilon^2))$) when the loss function is both smooth and strongly convex. Telgarsky [102] studies the primal-dual structure of GBM. By taking advantage of the dual structure, Telgarsky presented a linear convergence result for GBM with the line search step-size rule. However, the constants in the linear rate are not as transparent as the ones we obtain in this chapter, the only exception being the exponential loss function\footnote{The rate for other loss functions involves a quantity that can be exponentially large in the number of features $p$.}. There are several works for the convergence rate that apply to specific loss functions. Freund and Schapire [37] showed a linear convergence rate for AdaBoost (this can be thought of as GBM with exponential loss and line search rule) under a weak learning assumption. Mukherjee, Rudin and Schapire [68] showed an $O(1/\epsilon)$ rate for AdaBoost, but the constant depends on the dataset and can be exponentially large in the dimension of the problem. We
refer the readers cite telgarsky2012primal for a throughout review on the earlier work on Boosting. For LS-Boost (gradient boosting with a least squares loss function), Freund, Grigas and Mazumder [35] recently show a linear rate of convergence, but the rate is not informative when the number of weak-learners is large. Our analysis here provides a much sharper description of the constant—we achieve this by using a different analysis technique.

### 5.1.4 Contributions

Our contributions in this chapter can be summarized as follows:

1. We propose RGBM, a new randomized version of GBM, which leads to significant computational gains compared to GBM. This is based on what we call a Random-then-Greedy procedure (i.e., we select a random subset of weak-learners and then find the best candidate among them by using a greedy strategy). In particular, this provides a formal justification of heuristics used in popular GBM implementations like XGBoost, and also suggests improvements. Our framework also provides guidelines for a principled choice of step-size rules in RGBM.

2. We derive new computational guarantees for RGBM, based on a coordinate descent interpretation. In particular, this leads to new guarantees for GBM that are superior to existing guarantees for certain loss functions. The constants in our computational guarantees are in terms of a curious geometric quantity that we call Minimal Cosine Angle — this relates to the density of the weak-learners in the prediction space.

3. From an optimization viewpoint, our Random-then-Greedy coordinate descent procedure leads to a novel generalization of coordinate descent-like algorithms. This is done by combining the efficiency of randomized coordinate descent (RCD) and sparsity of greedy coordinate descent (GCD); and promises to be of independent interest.

Notation: For an integer $s$, let $[s]$ denote the set $\{1, 2, \ldots, s\}$. For $a, b \in \mathbb{R}^p$, $\cos(a, b)$
denotes the cosine of the angle between \( a \) and \( b \), namely \( \cos(a, b) = \frac{a \cdot b}{\|a\|_2 \|b\|_2} \). Matrix \( B \) denotes the prediction for all samples over every possible weak-learner, namely \( B_{i,j} = b(x_i; \tau_j) \) for \( i \in [n] \) and \( j \in [K] \). \( B_j \) is the \( j \)th column of \( B \) and \( B_i \) is the \( i \)th row of \( B \). We say \( \{I_g\}_{g \in [G]} \) is a partition of \( [K] \) if \( \bigcup_{g \in [G]} I_g = [K] \) and \( I_g \)'s are disjoint. We often use the notation \( [a_i]_i \) to represent a vector \( a \).

5.2 Random-then-Greedy Coordinate Descent in the Coefficient Space

Let \( [b(x; \tau_j)]_{j \in [K]} \) be a family of all possible weak-learners. Let

\[
f(x) = \sum_{j=1}^{K} \beta_j b(x; \tau_j)
\]

be a weighted sum of all \( K \) weak-learners \( b(x; \tau_j) \), where \( \beta_j \) is the coefficient of the \( j \)th weak-learner (we expect a vast majority of the \( \beta_j \)'s to be zero). We refer to the space of \( \beta \in \mathbb{R}^K \) as the "coefficient space". We can rewrite the minimization problem (6.2) in the coefficient space as:

\[
\min_{\beta} L(\beta) := \sum_{i=1}^{n} \ell \left( y_i, \sum_{j=1}^{K} \beta_j b(x_i; \tau_j) \right).
\]

Here, we assume \( K \) to be finite (but potentially a very large number). We expect that our results can be extended to deal with an infinite number of weak-learners, but we do not pursue this direction in this chapter for simplicity of exposition.

Now, let \( B \) be a \( n \times K \) matrix of the predictions for all feature vectors over every possible weak-learner, namely \( B = [b(x_i; \tau_j)]_{i \in [n], j \in [K]} \), then each column of \( B \) represents the prediction of one weak-learner for the \( n \) samples, and each row of \( B \) represents the prediction of all weak-learners for a single sample. Thus we can rewrite
Algorithm 12 presents the Random-then-Greedy Coordinate Descent (RtGCD) algorithm for solving (5.7). We initialize the algorithm with $\beta = 0$. At the start of the $m$th iteration, the algorithm randomly chooses a subset $J$ of the coordinates using one of the four types of selection rules described in Section 5.1.2. The algorithm then "greedily" chooses $j_m \in J$ by finding a coordinate in $\nabla_j L(\beta^m)$ with the largest magnitude. We then perform a coordinate descent step on the $j_m$ coordinate with either a line-search step-size rule or a constant step-size rule.

**Algorithm 12** Random-then-Greedy Coordinate Descent (RtGCD) in the Coefficient Space

Initialization. Initialize with $\beta^0 = 0$.

For $m = 0, \ldots, M - 1$ do:

Perform Updates.

1. Pick a random subset $J$ of coordinates by some rule (i.e., one of Type 0 - Type 3).
2. Use a greedy rule to find a coordinate in $J$: $j_m = \arg \max_{j \in J} |\nabla_j L(\beta^m)|$.
3. Choose the step-size $\rho_m$ by
   - line-search: $\rho_m = \arg \min_p \sum_{i=1}^n \ell(y_i, B_i \beta^m + pB_i j_m)$;
   - constant step-size: $\rho_m = -\rho \nabla_j L(\beta^m)$ for a given constant $\rho$.
4. Update coefficients: $\beta^{m+1} = \beta^m + \rho_m e_j^m$.

Output. $\sum_{j=1}^K \beta_j^M b(x; \tau_j)$.

The following proposition shows that RGBM (Algorithm 11) is equivalent to RtGCD in the coefficient space (Algorithm 12):

**Proposition 5.2.1.** Suppose Algorithm 11 makes the same choice of the random set $J$ as Algorithm 12 in each iteration, and the step-size rules are chosen to be the same in both algorithms. Then the outputs of Algorithm 11 and Algorithm 12 are the same.

Proof. We will show by induction that $f^m(x)$ in Algorithm 11 is the same as $\sum_{j=1}^K \beta_j^m b(x; \tau_j)$ in Algorithm 12 for $m = 0, 1, \ldots, M$. Then Proposition 5.2.1 holds as a special case for $m = M$. 

133
For \( m = 0 \), we have \( f^0(x) = 0 = \sum_{j=1}^{K} \beta_j^0 b(x; \tau_j) \). Suppose \( f^m(x) = \sum_{j=1}^{K} \beta_j^m b(x; \tau_j) \), then

\[
\nabla_j L(\beta^m) = -\langle B_j, r^m \rangle,
\]

(5.8)

where \( r^m \) is defined in Algorithm 11. In iteration \( m \), the same random subset \( J \) is chosen by both algorithms. Next, Algorithm 11 greedily chooses the weak-learner by

\[
\hat{j}_m = \arg\min_j \min_{\sigma} \sum_{i=1}^{n} (r_i^m - \sigma b(x_i; \tau_j))^2 = \arg\min_j \min_{\sigma} \| r^m - \sigma B_j \|_2^2.
\]

Notice that for any \( j \), it holds that \( \min_{\sigma} \| r^m - \sigma B_j \|_2^2 = \langle B_j, r^m \rangle \), whereby

\[
\hat{j}_m = \arg\min_j \| r^m - \langle B_j, r^m \rangle B_j \|_2^2 = \arg\min_j \left( -\frac{1}{2} \langle B_j, r^m \rangle^2 \right)
\]

\[
= \arg\max_j |\langle B_j, r^m \rangle| = \arg\max_j |\nabla_j L(\beta^m)|,
\]

where the second equality follows from \( \| B_j \|_2^2 = \sum_{i=1}^{n} b(x_i; \tau_j)^2 = 1 \) due to Assumption 5.1.1 and the last equality utilizes (5.8). Therefore, coordinate \( \hat{j}_m \) obtained by Algorithm 11 in the \( m \)th iteration is the same as that obtained by Algorithm 12.

Suppose that both algorithms use a step-size based on the line-search rule, then the step-size in Algorithm 11 is given by

\[
\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} \ell(y_i, f^m(x_i) + \rho b(x_i; \tau_{j_m})) = \arg\min_{\rho} \sum_{i=1}^{n} \ell(y_i, B_i \beta^m + \rho B_i j_m),
\]

where we have (by induction hypothesis) that \( f^m(x_i) = B_i \beta^m \). Thus the step-size \( \rho_m \) is the same as that chosen by Algorithm 12 (with line-search rule). Suppose both algorithms use a constant step-size rule with the same constant \( \rho \), then the step-size in Algorithm 11 is given by:

\[
\rho_m = \rho \left( \sum_{i=1}^{n} r_i^m b(x_i; \tau_{j_m}) \right) = \rho \langle r^m, B_{j_m} \rangle = -\rho \nabla_{j_m} L(\beta^m),
\]

134
which is the same step-size as that in Algorithm 12 (with constant step-size rule). Thus, the step-size \( \rho_m \) at the \( m \)th iteration in Algorithm 11 is the same as that of Algorithm 12 for both step-size rules. Therefore, it holds that

\[
f^{m+1}(x) = f^m(x) + \rho_m b(x; \tau_m) = \sum_{j=1}^K \beta_j^m b(x; \tau_{jm}) + \rho_m b(x; \tau_{jm}) = \sum_{j=1}^K \beta_j^{m+1} b(x; \tau_j),
\]

which furnishes the proof by induction.

Remark 5.2.1. As a special case when \( J \) contains all weak-learners, Algorithm 12 reduces to standard greedy coordinate descent and Proposition 5.2.1 shows GBM (Algorithm 13) is equivalent to greedy coordinate descent in the coefficient space.

5.3 Machinery: Structured Norms and Random Selection Rules

In this section, we introduce four norms and establish how they relate to the four types of selection rules to choose the set \( J \), as described in Section 5.1.2.

5.3.1 Infinity Norm, Slope Norm, Group Norm and a Combined Norm

We introduce the following definitions.

Definition 5.3.1. The “infinity norm” \( \| \cdot \|_{\text{inf}} \) of a vector \( a \in \mathbb{R}^K \) is defined as

\[
\|a\|_{\text{inf}} = \max_{j \in [K]} |a_j|. \quad \text{(Infinity norm)}
\]

Definition 5.3.2. The “slope norm” \( \| \cdot \|_S \) with parameter \( \gamma \in \mathbb{R}^K \) of a vector \( a \in \mathbb{R}^K \) is defined as

\[
\|a\|_S = \sum_{j=1}^K \gamma_j |a_{(j)}|, \quad \text{(Slope norm)}
\]
where the parameter $\gamma$ satisfies $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_K \geq 0$ and $\sum_{j=1}^{K} \gamma_j = 1$, and $|a_{(1)}| \geq |a_{(2)}| \geq \ldots \geq |a_{(K)}|$ are the decreasing absolute values of the coordinates of $a$.

**Definition 5.3.3.** If $\{I_g\}_{g \in [G]}$ is a partition of $[K]$, then the "group norm" of a vector $a \in \mathbb{R}^K$ is defined as

$$
\|a\|_g = \frac{1}{G} \sum_{g=1}^{G} \|a_{I_g}\|_{\infty} .
$$

(Group norm)

**Definition 5.3.4.** If $\{I_g\}_{g \in [G]}$ is a partition of $[K]$, then the "combined norm" of a vector $a \in \mathbb{R}^K$ with parameter $\gamma \in \mathbb{R}^G$ is defined as

$$
\|a\|_c = \sum_{g=1}^{G} \gamma_g \|a_{I_g}\|_{\infty} ,
$$

(Combined norm)

where the parameter $\gamma$ satisfies $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_G \geq 0$ and $\sum_{g=1}^{G} \gamma_g = 1$. Note that $\|a_{I_{(1)}}\|_{\infty} \geq \|a_{I_{(2)}}\|_{\infty} \geq \ldots \geq \|a_{I_{(G)}}\|_{\infty}$ are the sorted values of $\|a_{I_g}\|_{\infty}, g \in [G]$ where $\|a_{I_g}\|_{\infty}$ is the infinity-norm of the sub-vector of $a$ restricted to $I_g$.

**Remark 5.3.1.** Curiously the group and slope norms appear as regularizers in high-dimensional linear models, see [14, 70] for details. In this chapter however, they arise in a very different context.

It can be easily seen that the slope norm is a special instance of the combined norm where each group contains only one entry, and the group norm is another special instance of the combined norm where the parameter $\gamma_g \equiv \frac{1}{G}$ for $g \in [G]$.

With some elementary calculations, we can derive the dual norms of each of the above norms.

\[^2\text{The name stems from the fact that it is a combination of the slope and group norms.}\]
Proposition 5.3.1. (1) The dual norm of the slope norm is

\[ \|b\|_{s^*} = \max_{1 \leq i \leq K} \frac{\sum_{j=1}^{i} |b_{i,j}|}{\sum_{j=1}^{i} \gamma_j}. \]  

(2) The dual norm of the group norm is

\[ \|b\|_{g^*} = G \max_{1 \leq g \leq G} \|b_{i,g}\|_1. \]

(3) The dual norm of the combined norm is

\[ \|b\|_{c^*} = \max_{1 \leq g \leq G} \frac{\sum_{j=1}^{g} \|b_{i,j}\|_1}{\sum_{j=1}^{g} \gamma_j}, \]

where \( \|b_{i,(1)}\|_1 \geq \|b_{i,(2)}\|_1 \geq \cdots \geq \|b_{i,(G)}\|_1 \) are the decreasing values of \( \|b_{i,g}\|_1, g \in [G] \).

Remark 5.3.2. The proof for part (1) of Proposition 5.3.1 can be found in [115]. The proof of part (2) is straightforward, and the proof for part (3) follows from that of (1) and (2).

5.3.2 Random-then-Greedy Procedure

Here we introduce a Random-then-Greedy (RtG) procedure that uses a randomized scheme to deliver an approximate maximum of the absolute entries of a vector \( a \in \mathbb{R}^K \). The expected value of the (random) output available from the RtG procedure with four types of selection rules (cf Section 5.1.2) can be shown to be related to the four norms introduced in Section 5.3.1.

Formally, the RtG procedure is summarized below:

Random-then-Greedy (RtG) procedure
Given \( a \in \mathbb{R}^K \),

1. Randomly pick a subset of coordinates \( J \subseteq [K] \).
2. Output \( \hat{j} = \arg\max_{j \in J} |a_j| \) and \( |a_j| \).

We will next obtain the probability distribution of \( \hat{j} \), and the expectation of \( |a_j| \).

Let \( J \) be chosen by Type 1 selection rule, namely \( J \) is given by a collection of \( t \) coordinates (from \( K \)) without replacement. A simple observation is that the probability of a coordinate \( j \) being chosen depends upon the magnitude of \( a_j \) relative to the other values \( |a_i|, i \neq j \); and not the precise values of the entries in \( a \). Note also that if the value of \( |a_j| \) is higher than others, then the probability of selecting \( j \) increases: this is because (a) all coordinate indices in \( [K] \) are equally likely to appear in \( J \), and (b) coordinates with a larger value of \( |a_j| \) are chosen with higher probability. The following proposition formalizes the above observations and presents the probability of a coordinate being chosen.

**Proposition 5.3.2.** Consider the RtG Procedure for approximately finding the maximal coordinate of \( a \in \mathbb{R}^K \) (in absolute value). Recall that \( (j) \) is the index of the \( j^{th} \) largest coordinate of \( a \) in absolute value\(^3\), namely \(|a_{(1)}| \geq |a_{(2)}| \geq \cdots \geq |a_{(K)}|\). If the subset \( J \) is chosen by the Type 1 selection rule, the probability that \( (j) \) is returned is

\[
P(\hat{j} = (j)) := \gamma^K_t(j) = \frac{\binom{K-j}{t-1}}{\binom{K}{t}}.
\]

**Proof.** There are \( \binom{K}{t} \) different choices for the subset \( J \), and each subset is chosen with equal probability. The event \( \hat{j} = (j) \) happens if and only if \( (j) \in J \) and the remaining \( t-1 \) coordinates are chosen from the \( K-j \) coordinates. There are \( \binom{K-j}{t-1} \) different choices of choosing such a subset \( J \), which furnishes the proof of Proposition 5.3.2.

\[\Box\]

**Remark 5.3.3.** Note that \( \gamma^K_t(j) \) is monotonically decreasing in \( j \) for fixed \( K, t \) (because \( j \to \binom{K-j}{t-1} \) is monotonically decreasing in \( j \)). This corresponds to the intuition

\(^3\)In case of ties, we choose the index with smallest value.
that the RtG procedure returns a coordinate $j$ with a larger magnitude of $a_j$, with higher probability.

For most cases of interest, the dimension $K$ of the input vector is huge. When $K$ is asymptotically large, it is convenient to consider the distribution of the quantile $q = j/K$ (where $0 < q < 1$), instead of $j$. The probability distribution of this quantile evaluated at $j/K$ is given by $K\gamma^K_t(j)$. The following proposition states that $K\gamma^K_t(j)$ asymptotically converges to $t(1-q)^{t-1}$, the probability density function of the Beta distribution with shape parameters $(1,t)$ i.e., Beta$(1,t)$.

**Proposition 5.3.3.** It holds that for $0 < q < 1$,

$$\lim_{j,K \to \infty, j/K = q} K\gamma^K_t(j) = t(1-q)^{t-1}.$$  

**Proof.** By using the expression of $\gamma^K_t(j)$ and canceling out the factorials, it holds that

$$\gamma^K_t(j) = \frac{(K-t)(K-t-1)\cdots(K-j-t+2)}{(K-t)(K-t-1)\cdots(K-j+1)}.$$

Denote $A^K_t(j) = (1 - \frac{t-1}{K-1}) (1 - \frac{t-1}{K-2}) \cdots (1 - \frac{t-1}{K-j+1})$, then it holds that

$$\lim_{j,K \to \infty, j/K = q} \ln A^K_t(j) = \lim_{j,K \to \infty, j/K = q} \sum_{l=1}^{j-1} \ln \left(1 - \frac{t-1}{K-l}\right)$$

$$= \lim_{j,K \to \infty, j/K = q} \sum_{l=1}^{j-1} \frac{t-1}{K-l}$$

$$= \lim_{j,K \to \infty, j/K = q} (t-1) \ln \left(\frac{K-j}{K}\right)$$

$$= (t-1) \ln(1-q),$$
Figure 5-2: Figure shows the profiles of $K\gamma^K_t(j)$ (i.e., the probability distribution of the quantile $q = j/K$ for the RtG procedure, as described in the text) as a function of $q$. We consider three profiles (of $K\gamma^K_t(j)$) for three different values of $K$, and the Beta(1, 10) density function (we fix $t = 10$). We observe that for $K \approx 40$, the profile of $K\gamma^K_t(j)$ and that of the Beta(1,10) distribution are indistinguishable.

We consider three profiles (of $K\gamma^K_t(j)$) for three different values of $K$, and the Beta(1,10) density function (we fix $t = 10$). We observe that for $K \approx 40$, the profile of $K\gamma^K_t(j)$ and that of the Beta(1,10) distribution are indistinguishable.

where the second inequality uses $\ln \left(1 - \frac{j-1}{K-1}\right) \approx -\frac{j-1}{K-1}$ and the third equality is from $\sum_{j=1}^{K-1} \frac{j-1}{K-1} \approx \ln K - \ln(K - j) = \ln \left(\frac{K}{K-j}\right)$, when both $j, K$ are large and $j/K \approx q$. Therefore,

$$\lim_{j,K \to \infty, j/K=q} K\gamma^K_t(j) = \lim_{j,K \to \infty, j/K=q} \exp \left( \ln A_t^{(K,j)} \right) = t(1-q)^{t-1}$$

which completes the proof.

Figure 5-2 compares the probability distribution of the discrete variable $j/K$ and its continuous limit: as soon as $K \approx 40$, the function $K\gamma^K_t(j)$ becomes (almost) identical to the Beta density.

Given a partition $\{I_g\}_{g \in [G]}$ of $[K]$, let us denote for every $g \in [G]$:

$$b_g = \max_{j \in I_g} |a_j| \quad \text{and} \quad k_g = \arg \max_{j \in I_g} |a_j|.$$  (5.11)

For the RtG procedure with Type 2 random selection rule, note that $P(\hat{j} = k_g) = 1/G$ for all $g \in [G]$. Type 3 selection rule is a combination of Type 1 and Type 2 selection rules. One can view the RtG procedure with Type 3 selection rule as a two-step
procedure: (a) compute $b_g$ and $k_g$ as in (5.11); and (b) use a RtG procedure with Type 1 rule on $\{b_g\}_{g \in [G]}$. Using an argument similar to that used in Proposition 5.3.2, we have

$$P(\hat{j} = k(g)) = \gamma_{t}^G(g),$$

(5.12)

where we recall that $|a_{k(1)}| \geq |a_{k(2)}| \geq \ldots \geq |a_{k(G)}|$ and $b(g) = |a_{k(g)}|$ for all $g$.

The following Proposition establishes a connection among the four types of selection rules and the four norms described in Section 5.3.1.

**Proposition 5.3.4.** Consider the RtG procedure for finding the approximate maximum of the absolute values of $a$. It holds that

$$\mathbb{E}[|a_j|] = \|a\|_F,$$

where $F$ denotes the slope norm with parameter $\gamma = [\gamma_{t}^K(j)]$, the group norm, or the combined norm with parameter $\gamma = [\gamma_{t}^G(j)]$ when the selection rule is Type 0, Type 1, Type 2 or Type 3 (cf Section 5.1.2), respectively.

**Proof.**

**Type 0:** This corresponds to the deterministic case and $|a_j| = \max_j |a_j| = \|a\|_{\text{inf}}$.

**Type 1:** It follows from Proposition 5.3.2 that $P(\hat{j} = (j)) = \gamma_{t}^K(j)$, thus

$$\mathbb{E}[|a_j|] = \sum_{j=1}^{K} \gamma_{t}^K(j)|a_{(j)}| = \|a\|_S.$$

**Type 2:** For the Type 2 random selection rule, we have $P(\hat{j} = k_g) = \frac{1}{G}$ for any $g \in [G]$, thus:

$$\mathbb{E}[|a_j|] = \frac{1}{G} \sum_{g=1}^{G} b_g = \frac{1}{G} \sum_{g=1}^{G} \|a_{I_g}\|_{\text{inf}} = \|a\|_G.$$
Type 3: It follows from (5.12) that

\[ \mathbb{E}[|a_j|] = \sum_{g=1}^{G} \gamma_i^G(g) b_{(y)} = \sum_{g=1}^{G} \gamma_i^G(g) \|a_{I_g}\|_\infty = \|a\|_C . \]

\[ \square \]

5.4 Computational Guarantees for RGBM

Here we derive computational guarantees for RGBM. We first introduce some standard regularity/continuity conditions on the scalar loss function \( \ell(y, f) \) that we require in our analysis.

**Definition 5.4.1.** We denote \( \frac{\partial \ell(y, f)}{\partial f} \) as the derivative of the scalar loss function \( \ell \) wrt the prediction \( f \). We say that \( \ell \) is \( \sigma \)-smooth if for any \( y \) and predictions \( f_1 \) and \( f_2 \), it holds that

\[ \ell(y, f_1) \leq \ell(y, f_2) + \frac{\partial \ell(y, f_2)}{\partial f} (f_1 - f_2) + \frac{\sigma}{2} (f_1 - f_2)^2 . \]

We say \( \ell \) is \( \mu \)-strongly convex (with \( \mu > 0 \)) if for any \( y \) and predictions \( f_1 \) and \( f_2 \), it holds that

\[ \ell(y, f_1) \geq \ell(y, f_2) + \frac{\partial \ell(y, f_2)}{\partial f} (f_1 - f_2) + \frac{\mu}{2} (f_1 - f_2)^2 . \]

The following lists some commonly-used loss function in GBM and their continuity constants:

**\( \ell_2 \) or least squares loss:** \( \ell(y, f) = \frac{1}{2} (y - f)^2 \) is 1-smooth and 1-strongly convex.

**Huber loss:** The Huber loss function with parameter \( d > 0 \) given by

\[ l_d(y, f) = \begin{cases} \frac{1}{2} (y - f)^2 & \text{for } |f - y| \leq d \\ d |y - f| - \frac{1}{2} d^2 & \text{otherwise ,} \end{cases} \]

is 1-smooth but not strongly convex.

**Logistic loss:** We consider a regularized version of the usual logistic loss function:

\[ l_d(y, f) = \log(1 + e^{-yf}) + \frac{d}{2} f^2 \] with \( d \geq 0 \), which is \( (\frac{1}{4} + d) \)-smooth and \( d \)-strongly
convex (when \( d > 0 \)). A special case is the usual logistic loss when \( d = 0 \), which is \( \frac{1}{4} \)-smooth but not strongly convex.

**Exponential loss:** \( \ell(y, f) = \exp(-y f) \) is neither strongly convex nor smooth.

Notice that the objective function \( L(\beta) \) has an invariant subspace in the coefficient space, namely for any \( \lambda \in \text{Ker}(B) \), it holds that \( L(\beta) = L(\beta + \lambda) \). Let us denote

\[
Z(\hat{\beta}) := \left\{ \beta \mid B\beta = B\hat{\beta} \right\}
\]

as the invariant subspace of \( \hat{\beta} \). Recall that \( \mathcal{F} \in \{\inf, \mathcal{S}, \mathcal{G}, \mathcal{C}\} \), and \( \mathcal{F}^* \) is the dual norm of \( \mathcal{F} \), which is defined in Section 5.3.1. We define a distance metric in the \( \beta \)-space as:

\[
\text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1, \beta_2) := \text{Dist}_{\mathcal{F}^*}(Z(\beta_1), Z(\beta_2)) = \min_{b \in Z(\beta_1), b \in Z(\beta_2)} \|b - \hat{b}\|_{\mathcal{F}^*} = \min_{\lambda \in \text{Ker}(B)} \|\beta_1 - \beta_2 - \lambda\|_{\mathcal{F}^*},
\]

which is the usual notion of distance between subspaces in the \( \mathcal{F}^* \) norm. In particular, if \( \beta_1, \beta_2 \in Z(\hat{\beta}) \), then \( \text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1, \beta_2) = 0 \). Note \( \text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*} \) is a pseudo-norm with the following properties.

**Proposition 5.4.1.**

1. \( \text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1, \beta_2) \) is symmetric: i.e., for any \( \beta_1 \) and \( \beta_2 \), we have

\[
\text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1, \beta_2) = \text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_2, \beta_1).
\]

2. \( \text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1, \beta_2) \) is translation invariant: i.e., for any \( \beta_1, \beta_2 \) and \( \hat{\beta} \), we have

\[
\text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1 - \hat{\beta}, \beta_2 - \hat{\beta}) = \text{Dist}^{\mathcal{F}^*}_{\mathcal{F}^*}(\beta_1, \beta_2).
\]

**Proof.**
1. The proof of this part follows from

\[ \text{Dist}_{\mathcal{F}}^B(\beta_1, \beta_2) = \min_{b \in Z(\beta_1), \hat{b} \in Z(\beta_2)} \| b - \hat{b} \|_{\mathcal{F}^*} = \min_{b \in Z(\beta_1), \hat{b} \in Z(\beta_2)} \| \hat{b} - b \|_{\mathcal{F}^*} = \text{Dist}_{\mathcal{F}}^B(\beta_2, \beta_1). \]

2. The proof of this part follows from

\[ \text{Dist}_{\mathcal{F}}^B(\beta_1 - \hat{\beta}, \beta_2 - \hat{\beta}) = \min_{\lambda \in \text{Ker}(B)} \| (\beta_1 - \hat{\beta}) - (\beta_2 - \hat{\beta}) - \lambda \|_{\mathcal{F}^*} = \min_{\lambda \in \text{Ker}(B)} \| \beta_1 - \beta_2 - \lambda \|_{\mathcal{F}^*} = \text{Dist}_{\mathcal{F}}^B(\beta_1, \beta_2). \]

### 5.4.1 Minimal Cosine Angle

Here we introduce a novel geometric quantity Minimal Cosine Angle (MCA) \( \Theta \), which measures the density of the collection of weak-learners in the prediction space. As we will see later in this section, MCA plays a central role in the computational guarantees for RGBM.

**Definition 5.4.2.** The Minimal Cosine Angle (MCA) of a set of weak-learners with respect to the \( \mathcal{F} \) norm is defined as:

\[
\Theta := \min_{c \in \text{Range}(B)} \left\| \left[ \cos(B_j, c) \right]_{j=1, \ldots, K} \right\|_{\mathcal{F}^*}.
\]  

The quantity \( \Theta \) measures how "dense" the weak-learners are in the prediction space. Figure 5-3 provides an illustration in a simple 2D example when \( \mathcal{F} \) is the infinity norm. Given weak-learners \( B_1, \ldots, B_K \), we compute the cosine of the angle between each weak-learner and a direction \( c \). The \( \mathcal{F} \) norm can be viewed as an approximation of the infinity norm, which is the norm corresponding to the traditional GBM. MCA refers to the minimum (over all directions indexed by \( c \)) of such reweighted angles.

We next present several equivalent definitions of \( \Theta \):
Figure 5-3: Illustration of the relationship between $\Theta$ and density of weak-learners in a 2D example. (a), (b), (c) represent poorly-spread weak-learners, moderately spread weak-learners and densely spread weak-learners, respectively. When $\mathcal{F}$ is the infinity norm, the values of $\Theta$ are given by: (a) $\Theta^2 \approx 0$; (b) $\Theta^2 = 1/2$; and (c) $\Theta^2 \approx 0.933$ — the weak-learners are more spread out for higher values of $\Theta$.

**Proposition 5.4.2.**

$$\Theta = \min_{c \in \text{Range}(B)} \frac{\|B^T c\|_{\mathcal{F}}}{\|c\|_2} = \min_a \frac{\|B a\|_2}{\text{Dist}_{\mathcal{F}}^B(0, a)} > 0. \quad (5.15)$$

**Proof.** The first equality follows directly by rewriting (6.7). Notice that for any norm $\mathcal{F}$ in $\mathbb{R}^K$ (a finite dimensional space), there exists a scalar parameter $\gamma > 0$ such that $\|B^T c\|_{\mathcal{F}} \geq \gamma \|B^T c\|_{\infty}$. Thus

$$\Theta = \min_{c \in \text{Range}(B), \|c\|_2 = 1} \frac{\|B^T c\|_{\mathcal{F}}}{\gamma \|B^T c\|_{\infty}} > 0,$$

where the second inequality follows from the observation that $c \in \text{Range}(B)$. We now proceed to show the second equality of (5.15).

By the definition of $\text{Dist}_{\mathcal{F}}^B$, and the definition of the dual norm, we have

$$\text{Dist}_{\mathcal{F}}^B(0, a) = \min_{\lambda \in \text{Ker}(B)} \|a - \lambda\|_{\mathcal{F}^*} = \min_{\lambda \in \text{Ker}(B)} \max_{\|b\|_{\mathcal{F}} \leq 1} \langle a - \lambda, b \rangle = \max_{\|b\|_{\mathcal{F}} \leq 1} \min_{\lambda \in \text{Ker}(B)} \langle a - \lambda, b \rangle$$

$$= \max_{\|b\|_{\mathcal{F}} \leq 1, b \in \text{Range}(B^T)} \langle a, b \rangle = \max_{\|b\|_{\mathcal{F}} \leq 1, b \in \text{Range}(B^T)} |\langle a, b \rangle| = \max_{b \in \text{Range}(B^T)} \frac{|\langle a, b \rangle|}{\|b\|_{\mathcal{F}}},$$

where the third equality uses Von Neumann's Minimax Theorem, and the fourth
equality is based on the observation

\[
\min_{\lambda \in \text{Ker}(B)} \langle a - \lambda, b \rangle = \begin{cases} 
-\infty & \text{for } b \not\in \text{Range}(B^T) \\
\langle a, b \rangle & \text{for } b \in \text{Range}(B^T).
\end{cases}
\]

Therefore,

\[
\min_a \frac{\|Ba\|_2}{\text{Dist}_F^B(0, a)} = \min_{b \in \text{Range}(B^T), a} \frac{\|Ba\|_2\|b\|_F}{\|\langle a, b \rangle\|}.
\]

Denote \( P_B = B^T(BB^T)^+B \) as the projection matrix onto \( \text{Range}(B^T) \), then we have

\( P_b b = b \) for any \( b \in \text{Range}(B^T) \). Thus

\[
\min_a \frac{\|Ba\|_2}{\text{Dist}_F^B(0, a)} = \min_{b \in \text{Range}(B^T), a} \frac{\|Ba\|_2\|b\|_F}{\|\langle a, P_B b \rangle\|} = \min_{b \in \text{Range}(B^T), a} \frac{\|Ba\|_2\|b\|_F}{\|\langle Ba, (BB^T)^+Bb \rangle\|}.
\]  

(5.16)

Now denote \( c = (BB^T)^+Bb \), then \( c \in \text{Range}(B) \) and \( B^T c = P_B b = b \). Note that for any \( a \), we have \( \|Ba\|_2\|c\|_2 \geq |\langle Ba, c \rangle| \), which implies

\[
\min_a \frac{\|Ba\|_2}{|\langle Ba, c \rangle|} \geq \frac{1}{\|c\|_2}.
\]

Since \( c \in \text{Range}(B) \), there exists a vector \( a \) satisfying \( Ba = c \), which leads to

\[
\frac{\|Ba\|_2}{|\langle Ba, c \rangle|} = \frac{\|c\|_2}{\|c\|_2} = \frac{1}{\|c\|_2},
\]

from which it follows that

\[
\min_a \frac{\|Ba\|_2}{|\langle Ba, c \rangle|} = \frac{1}{\|c\|_2}.
\]  

(5.17)

Substituting \( c = (BB^T)^+Bb \) and combining (5.16) and (5.17) yields

\[
\min_a \frac{\|Ba\|_2}{\text{Dist}_F^B(0, a)} = \min_{c \in \text{Range}(B)} \frac{\|B^T c\|_F}{\|c\|_2},
\]

which furnishes the proof.

To gain additional intuition about MCA, we consider some examples:

**Example 1** (Orthogonal Basis with Infinity Norm) Suppose \( F \) is the infinity norm
and the set of weak-learners in \( \mathbb{R}^p \) forms an orthogonal basis (e.g., the discrete Fourier basis in \( \mathbb{R}^p \)), then \( \Theta = \frac{1}{\sqrt{p}} \).

**Example 2** (Orthogonal Basis with Slope Norm) Suppose \( F \) is the slope norm with a parameter sequence \( \gamma \in \mathbb{R}^p \) and the set of weak-learners in \( \mathbb{R}^p \) forms an orthogonal basis, then

\[
\Theta = \min \left\{ \frac{1}{\sqrt{2}}(\gamma_1 + \gamma_2), \ldots, \frac{1}{\sqrt{p}}(\gamma_1 + \ldots + \gamma_p) \right\} . \tag{5.18}
\]

We present a proof for (5.18) and notice that the result for Example 1 follows as a special case. Without loss of generality, we assume \( B \) to be an identity matrix. It then follows from the second equality of (5.15) that

\[
\Theta = \min_{\|a\|_{S^*} = 1} \|a\|_2 . \tag{5.19}
\]

By flipping the constraint and the objective function of (5.19) we can instead consider the following equivalent problem:

\[
\Phi = \max_{\|a\|_2 \leq 1} \|a\|_{S^*} = \max_{\|a\|_2 \leq 1} \|a\|_{S^*} ,
\]

and we have \( \Theta = \frac{1}{\Phi} \). Using the definition of the dual of the slope norm (see (5.9)), notice that for any \( i \in [p] \), it follows from the \( \ell_1-\ell_2 \) norm inequality that

\[
\sum_{j=1}^{i} |a(j)| \leq \sqrt{i} \left( \sum_{j=1}^{i} a^2(j) \right) \leq \sqrt{i} \|a\|_2 \leq \sqrt{i} ,
\]

and therefore

\[
\Phi = \max_{\|a\|_2 \leq 1} \max_{i \in [p]} \left\{ \frac{\sum_{j=1}^{i} |a(j)|}{\sum_{j=1}^{i} \gamma_j} \right\} \leq \max_{i \in [p]} \left\{ \frac{\sqrt{i}}{\sum_{j=1}^{i} \gamma_j} \right\} .
\]

Meanwhile, for any \( i \in [p] \) define \( \tilde{a}_1 = \cdots = \tilde{a}_i = \frac{1}{\sqrt{i}} \) and \( \tilde{a}_{i+1} = \cdots = \tilde{a}_p = 0 \), then we have \( \Phi \geq \|\tilde{a}\|_{S^*} = \frac{\sqrt{i}}{\sum_{j=1}^{i} \gamma_j} \). Therefore \( \Phi = \max_{i \in [p]} \left\{ \frac{\sqrt{i}}{\sum_{j=1}^{i} \gamma_j} \right\} = \frac{1}{\Theta} \) — this furnishes
**Figure 5-4:** Plot shows how $\Theta$ varies with $p$ (log-log plot) when the weak-learners are orthogonal and $\mathcal{F}$ corresponds to the slope norm with parameter $\gamma = [\gamma^p_i(j)]_j$ (see (5.10)). We show three profiles for three different values of $t$. Note that $\Theta$ is defined only for $p \geq t$. This MCA corresponds to the Type 2 random selection rule.

the proof.

**Remark 5.4.1.** Consider using a Type 1 random selection rule in RGBM, then the corresponding norm $\mathcal{F}$ is the slope norm with parameter $\gamma^p_t = [\gamma^p_i(j)]_j$ as defined in (5.10). Figure 5-4 shows the value of $\Theta$ (computed by formula (5.18)) versus the dimension $p$—we consider different values of $t$ and use an orthogonal basis. The figure suggests that $\Theta$ depends upon $p, t$ as follows:

$$
\Theta \sim \begin{cases} 
\frac{1}{\sqrt{p}} & \text{if } p \leq t^2, \\
\frac{t}{p} & \text{otherwise.}
\end{cases}
$$

**Example 3** (Binary Basis with Infinity Norm) Suppose $\mathcal{F}$ is the infinity norm, and the basis matrix $B$ satisfies $B_{i,j} \in \{-1, 0, 1\}$ — leading to $3^p$ different weak-learners. In this case,

$$
\Theta = \frac{1}{\sqrt{1^2 + (\sqrt{2} - 1)^2 + \cdots + (\sqrt{p} - \sqrt{p - 1})^2}} \propto \frac{1}{\sqrt{\ln p}}. 
$$

(5.20)

We present a proof for (5.20). Since $B_{i,j} \in \{-1, 0, 1\}$, we have:

$$
\Theta = \min_c \max_j |\cos(B_j, c)| = \min_c \max_{i \in [p]} \max_{\|B_j\|_1 = i} |\cos(B_j, c)| = \min_c \max_{i \in [p]} \frac{\sum_{k=1}^{i} |c(k)|}{\sqrt{i} \|c\|_2}. 
$$

(5.21)
Recall the definition of the dual slope norm $S^*$ in Proposition 5.3.1. Observe that
\[
\max_{i \in [p]} \frac{\sum_{k=1}^{|c_{(i)}|}}{\sqrt{i}} = \|c\|_{S^*},
\]
where $\gamma = [\sqrt{i} - \sqrt{i - 1}]_{i \in [p]}$ is the parameter of the slope norm $S$, thus
\[
\Theta = \min_c \frac{\|c\|_{S^*}}{\|c\|_2} = \min_a \frac{\|a\|_2}{\|a\|_S} = \min_{\|a\|_S = 1} \|a\|_2,
\]
where the second equality uses (5.15) with $F = S^*$ and $B$ as the identity matrix.

By flipping the constraint and the objective function, we can instead consider the following equivalent problem:
\[
\Phi := \max_{\|a\|_2 = 1} \|a\|_S = \max_{\|a\|_2 \leq 1} \|a\|_S,
\]
with $\Theta = \frac{1}{\Phi}$. By the Cauchy-Schwarz inequality, it holds that
\[
\|a\|_S^2 = \left( \sum_{i=1}^p \gamma_i |a_{(i)}| \right)^2 \leq \left( \sum_{i=1}^p \gamma_i^2 \right) \|a\|_2^2 = \left( \sum_{i=1}^p (\sqrt{i} - \sqrt{i - 1})^2 \right) \|a\|_2^2,
\]
with equality being achieved when $a \propto [\sqrt{i} - \sqrt{i - 1}]_i$. Thus we have $\Phi = \sqrt{\sum_{i=1}^p (\sqrt{i} - \sqrt{i - 1})^2}$ and $\Theta = 1/\Phi$. Notice that
\[
\frac{1}{4} \sum_{i=1}^p \frac{1}{i - 1} \leq \sum_{i=1}^p (\sqrt{i} - \sqrt{i - 1})^2 = \sum_{i=1}^p \frac{1}{(\sqrt{i} + \sqrt{i - 1})^2} \leq 1 + \frac{1}{4} \sum_{i=2}^p \frac{1}{i - 1},
\]
where the lhs and rhs of the above are both $O(\ln p)$. This implies that $\sum_{i=1}^p (\sqrt{i} - \sqrt{i - 1})^2 \propto \ln p$, thereby completing the proof. \hfill \Box

\textbf{Remark 5.4.2.} The binary basis described in Example 3 (with $\Theta = O(\frac{1}{\sqrt{\ln p}})$) is more densely distributed in the prediction space when compared to Example 1 (with $\Theta = O(\frac{1}{\sqrt{p}})$) — See Figure 5-3 (b) and (c).
5.4.2 Computational Guarantees: Strongly Convex Loss Function

We establish computational guarantees for RGBM when the scalar loss function \( \ell \) is both smooth and strongly convex. Let \( E_m \) denote the expectation over the random selection scheme at iteration \( m \). Let \( E_{\xi_m} \) denote the expectation over the random selection scheme up to iteration \( m \). The following theorem presents the linear convergence rate for RGBM.

**Theorem 5.4.1.** Let \( \ell \) be \( \mu \)-strongly convex and \( \sigma \)-smooth. Consider RGBM (Algorithm 11) or RtGCD (Algorithm 12) with either a line-search step-size rule or constant step-size rule with \( \rho = \frac{1}{\sigma} \). If \( \Theta \) denotes the value of the corresponding MCA, then for all \( M > 0 \) we have:

\[
E_{\xi_m}[L(\beta^M) - L(\beta^*)] \leq (1 - \frac{\mu}{\sigma^2})^M (L(\beta^0) - L(\beta^*)) .
\]

Notice that in the special case when \( J \) is chosen deterministically as the set of all weak-learners, Theorem 5.4.1 leads to a linear convergence rate for GBM [41]. Some prior works have also presented a linear convergence rate for GBM, but our results are different. For example, [102] shows a linear convergence rate but the constant is exponential in the number of features \( p \), except for the exponential loss [4]. [35] presents a linear convergence rate for LS-Boost (GBM with a least squares loss function) of the form \( O(\tau^M) \), where the parameter \( \tau = 1 - \lambda_{\text{min}}(B^TB)/4K \) depends upon \( \lambda_{\text{min}}(A) \), the minimal non-zero eigenvalue of a matrix \( A \). In GBM, \( K \) is usually exponentially large, thus \( \tau \) can be close to one. The constant derived herein, has a superior dependence on the number of weak-learners, and does not blow up as \( K \) becomes large. We obtain an improved rate since we employ a different analysis technique based on MCA.

**Remark 5.4.3.** We study the convergence rate of RGBM as a function of \( t \) using the

---

\(^4\)The result of [102] for the exponential loss function is superior to that presented here, as their analysis is targeted towards this loss function.
same setup considered in Remark 5.4.1. Using an “epoch” (i.e., the cost to evaluate all weak-learners across all samples) as the unit of computational cost, the cost per iteration of RGBM is $t/p$ epochs. Then the (multiplicative) improvement per epoch is

$$(1 - \frac{\mu}{\sigma})^{p/t} \sim \left\{ \begin{array}{ll} (1 - \frac{\mu}{\sigma})^{p/t} & \text{if } t \leq \sqrt{p}, \\ (1 - \frac{\mu^2}{p^2\sigma})^{p/t} & \text{otherwise}. \end{array} \right.$$  

This suggests that we should choose $t \sim \sqrt{p}$ when the weak-learners are almost orthogonal. Recall that from a coordinate descent perspective, $RtGCD$ with $t = 1$ leads to RCD, and $RtGCD$ with $t = p$ leads to GCD. Choosing $t$ to be larger than $O(\sqrt{p})$ will not lead to any improvement in the theoretical convergence rate, though it will lead to an increase in computational cost.

The following string of propositions will be needed for the proof of Theorem 5.4.1. Proposition 5.4.3 establishes a relationship among the four selection rules for choosing subset $J$ in RGBM (Algorithm 12) and the norms introduced in Section 5.3.1.

**Proposition 5.4.3.** Consider Algorithm 12 with the four types of selection rules for choosing the set $J$ as described in Section 5.1.2. For any iteration index $m$, we have

$$E_m \left[ (\nabla_{jm} L(\beta^m))^2 \right] = \left\| \left[ \nabla_j L(\beta^m)^2 \right]_j \right\|_F \geq \left\| \nabla L(\beta^m) \right\|_F^2,$$

where $F$ is the infinity norm, the slope norm with parameter $\gamma = [\gamma^K_t(j)]_j$, the group norm, or the combined norm with parameter $\gamma = [\gamma^G_t(j)]_j$ when the selection rule is Type 0, Type 1, Type 2 or Type 3, respectively.

**Proof.** The equality is a direct result of Proposition 5.3.4 with $a_j = (\nabla_j L(\beta^m))^2$. Notice that the $F$ norm of $a$ is a weighted sum of its coordinates—for notational convenience, we denote these weights by a vector $\lambda \in \mathbb{R}^K$ that satisfies: $\left\| \left[ \nabla_j L(\beta^m)^2 \right]_j \right\|_F = \sum_j \lambda_j (\nabla_j L(\beta^m))^2$; and $\lambda \geq 0$, $\sum \lambda_j = 1$. Thus we have

$$\left\| \left[ \nabla_j L(\beta^m)^2 \right]_j \right\|_F = \left( \sum \lambda_j \right) \left( \sum \lambda_j (\nabla_j L(\beta^m))^2 \right) \geq \left( \sum \lambda_j |\nabla_j L(\beta^m)| \right)^2 = \left\| \nabla L(\beta^m) \right\|_F^2,$$

151
where the inequality follows from the Cauchy Schwarz inequality.

The following proposition can be viewed as a generalization of the mean-value inequality.

**Proposition 5.4.4.** For $a \in \text{Range}(B^T)$ and $t > 0$, it holds that

$$\min_{\beta} \left\{ \langle a, \beta - \beta^* \rangle + \frac{t}{2} \text{Dist}_{\mathcal{F}^*}(\beta, \beta^*)^2 \right\} = \frac{1}{2t} \|a\|_{\mathcal{F}}^2.$$

**Proof:** Let $b = \beta - \beta^*$, $\text{Ker}(B) = \{ \lambda \mid B\lambda = 0 \}$ and $c = b + \lambda$. By the definition of $\text{Dist}_{\mathcal{F}^*}$, we have:

$$\min_{\beta} \left\{ \langle a, \beta - \beta^* \rangle + \frac{t}{2} \text{Dist}_{\mathcal{F}^*}(\beta, \beta^*)^2 \right\} = \min_{b \in \text{Ker}(B)} \min_{\lambda} \left\{ \langle a, b \rangle + \frac{t}{2} \|b + \lambda\|^2_{\mathcal{F}^*} \right\}$$

$$= \min_{\lambda \in \text{Ker}(B)} \left\{ -\langle a, \lambda \rangle + \min_{b+\lambda} \langle a, b + \lambda \rangle + \frac{t}{2} \|b + \lambda\|^2_{\mathcal{F}^*} \right\} = \min_{\lambda \in \text{Ker}(B)} \min_{c} \left\{ \langle a, c \rangle + \frac{t}{2} \|c\|^2_{\mathcal{F}^*} \right\}$$

where the third equality considers $a \in \text{Range}(B^T)$ and makes use of the observation that $\langle a, \lambda \rangle = 0$ for $\lambda \in \text{Ker}(B)$. Notice that

$$\frac{t}{2} \|c\|^2_{\mathcal{F}^*} + \frac{1}{2t} \|a\|^2_{\mathcal{F}} \geq \|c\|_{\mathcal{F}^*} \|a\|_{\mathcal{F}} \geq |\langle a, c \rangle|,$$

thus $\min_c \{ \langle a, c \rangle + \frac{t}{2} \|c\|^2_{\mathcal{F}^*} \} \leq -\frac{1}{2t} \|a\|^2_{\mathcal{F}}$. On the other hand, if $\hat{c} = \frac{1}{t} \|a\|_{\mathcal{F}} \arg \min_{\|c\|_{\mathcal{F}^*} \leq 1} \langle a, c \rangle$, then we have

$$\|\hat{c}\|_{\mathcal{F}^*} = \frac{1}{t} \|a\|_{\mathcal{F}} \quad \text{and} \quad \langle a, \hat{c} \rangle = -\frac{1}{t} \|a\|_{\mathcal{F}} \max_{\|c\|_{\mathcal{F}^*} \leq 1} \langle a, c \rangle = -\frac{1}{t} \|a\|^2_{\mathcal{F}},$$

whereby $\langle a, \hat{c} \rangle + \frac{t}{2} \|\hat{c}\|^2_{\mathcal{F}^*} = -\frac{1}{2t} \|a\|^2_{\mathcal{F}}$. Therefore it holds that

$$\min_c \{ \langle a, c \rangle + \frac{t}{2} \|c\|^2_{\mathcal{F}^*} \} = -\frac{1}{2t} \|a\|^2_{\mathcal{F}},$$
which furnishes the proof.

Proposition 5.4.5. If $\ell$ is $\mu$-strongly convex, then it holds for any $\beta$ and $\hat{\beta}$ that

$$L(\hat{\beta}) \geq L(\beta) + \langle \nabla L(\beta), \hat{\beta} - \beta \rangle + \frac{1}{2} \mu \Theta^2 \text{Dist}^B_\mathcal{F}(\hat{\beta}, \beta).$$

Proof. Since $\ell$ is $\mu$-strongly convex, we have

$$L(\hat{\beta}) = \sum_{i=1}^{n} \ell(y_i, B_i \hat{\beta})$$

$$\geq \sum_{i=1}^{n} \left\{ \ell(y_i, B_i \beta) + \frac{\partial \ell(y_i, B_i \hat{\beta})}{\partial f} \langle B_i, \hat{\beta} - \beta_i \rangle + \frac{\mu}{2} \|B_i\|_2^2 \|\hat{\beta} - \beta_i\|^2 \right\}$$

$$= L(\beta) + \langle \nabla L(\beta), \hat{\beta} - \beta \rangle + \frac{\mu}{2} \|B(\hat{\beta} - \beta)\|_2^2$$

$$\geq L(\beta) + \langle \nabla L(\beta), \hat{\beta} - \beta \rangle + \frac{\mu \Theta^2}{2} \text{Dist}^B_\mathcal{F}(0, \hat{\beta} - \beta)^2$$

$$= L(\beta) + \langle \nabla L(\beta), \hat{\beta} - \beta \rangle + \frac{\mu \Theta^2}{2} \text{Dist}^B_\mathcal{F}(\hat{\beta}, \beta)^2,$$

where the second inequality follows from Proposition 5.4.2, and the last equality utilizes the symmetry and translation invariance of Dist$^B_\mathcal{F}$. (Proposition 5.4.1). \qed

Proof of Theorem 5.4.1: For either the line-search step-size rule or the constant
step-size rule, it holds that

$$L(\beta^{m+1}) \leq L(\beta^m - \frac{1}{\sigma} \nabla_{j_m} L(\beta^m) e_{j_m})$$

$$= \sum_{i=1}^{n} \{ \ell(y_i, B_{i, \beta^m} - \frac{1}{\sigma} \nabla_{j_m} L(\beta^m) B_{i,j_m}) \}$$

$$\leq \sum_{i=1}^{n} \left\{ \ell(y_i, B_{i, \beta^m}) - g_i B_{i,j_m} \nabla_{j_m} L(\beta^m) + \frac{1}{2\sigma} B_{i,j_m}^2 (\nabla_{j_m} L(\beta^m))^2 \right\}$$

$$= L(\beta^m) - \frac{1}{2\sigma} (\nabla_{j_m} L(\beta^m))^2,$$

(5.24)

where the second inequality uses the fact that the loss function \( \ell \) is \( \sigma \)-smooth, and the last equality follows from \( \|B_{ij}\|^2 = 1 \). Thus \( L(\beta^m) \leq L(\beta^{m+1}) \). As a result of Proposition 5.4.3, taking expectation over both sides of (5.24) with respect to \( \mathbb{E}_{m+1} \) yields

$$\mathbb{E}_{m+1}[L(\beta^{m+1})] \leq L(\beta^m) - \frac{1}{2\sigma} \|\nabla L(\beta^m)\|^2_{\mathcal{F}}.$$

(5.25)

Meanwhile, it follows from Proposition 5.4.5 that

$$L(\beta^*) = \min_{\beta} L(\beta)$$

$$= \min_{\beta} \left[ L(\beta^m) + \langle \nabla L(\beta^m), \beta - \beta^m \rangle + \frac{\mu\Theta^2}{2} \text{Dist}_{\mathcal{F}}(\beta, \beta^m) \right]$$

(5.26)

$$= L(\beta^m) - \frac{1}{2\mu\Theta^2} \|\nabla L(\beta^m)\|^2_{\mathcal{F}},$$

where the last equality utilizes Proposition 5.4.4. Note that (5.26) together with (5.25) leads to

$$\mathbb{E}_{m+1}[L(\beta^{m+1})] - L(\beta^*) \leq L(\beta^m) - L(\beta^*) - \frac{1}{2\sigma} \|\nabla L(\beta^m)\|^2_{\mathcal{F}} \leq (1 - \frac{\mu\Theta^2}{\sigma}) (L(\beta^m) - L(\beta^*)).$$
and finally (5.22) follows by a telescoping argument.

\[ \]

5.4.3 Computational Guarantees: Non-Strongly Convex Loss Function

Define the initial level set of the loss function in the $\beta$-space as

$$\mathcal{L}_S_0 = \{ \beta \mid L(\beta) \leq L(\beta^0) \},$$

and its maximal distance to the optimal solution set in $\text{Dist}_{\mathcal{F}}$ as:

$$\text{Dist}_0 = \max_{\beta \in \mathcal{L}_S_0} \text{Dist}_{\mathcal{F}}^B(\beta, \beta^*).$$

$\mathcal{L}_S_0$ is unbounded if $Z(\beta^0)$ is unbounded. But interestingly, $\mathcal{L}_S_0$ is bounded in $\text{Dist}_{\mathcal{F}}^*$, i.e. $\text{Dist}_0 < \infty$, when the scalar loss function $\ell$ has a bounded level set.

**Proposition 5.4.6.** Suppose $\ell$ has a bounded level set, then $\text{Dist}_0$ is finite.

**Proof.** Since the convex function $\ell$ has a bounded level set, the set $\{B(\beta - \beta^*) \mid \beta \in \mathcal{L}_S_0\}$ is bounded. Thus there is a finite constant $C$ such that $\max_{\beta \in \mathcal{L}_S_0} \|B(\beta - \beta^*)\|_2 \leq C$. Therefore,

$$\text{Dist}_0 = \max_{\beta \in \mathcal{L}_S_0} \text{Dist}_{\mathcal{F}}^B(0, \beta - \beta^*)$$

$$\leq \max_{\|B(\beta - \beta^*)\|_2 \leq C} \text{Dist}_{\mathcal{F}}^B(0, \beta - \beta^*)$$

$$= \max_{\|B\| \leq C} \text{Dist}_{\mathcal{F}}^B(0, a)$$

$$\leq \max_{\|B\| \leq C} \frac{\|B\|_2}{\Theta}$$

$$= \frac{C}{\Theta}.$$
where the second inequality follows from Proposition 5.4.2.

The following theorem states convergence guarantees in expectation for Algorithm 11 and Algorithm 12 for a non-strongly convex loss function $\ell$.

**Theorem 5.4.2.** Consider RGBM (Algorithm 11) or equivalently RtGCD (Algorithm 12) with either line-search step-size rule or constant step-size rule. If $\ell$ is a $\sigma$-smooth function and has a bounded level set, it holds for all $M \geq 0$ that

$$
\mathbb{E}_{t,M} [L(\beta^M) - L(\beta^*)] \leq \frac{1}{L(\beta^0) - L(\beta^*)} + \frac{M}{2\sigma\text{Dist}_0^2} \leq \frac{2\sigma\text{Dist}_0^2}{M}.
$$

The following proposition is a generalization of the Cauchy-Schwarz Inequality.

**Proposition 5.4.7.** For $a \in \text{Range}(B^T)$, it holds that

$$
\|a\|_F \text{Dist}_{F,*}^B(\beta, \hat{\beta}) \geq \langle a, \beta - \hat{\beta} \rangle.
$$

**Proof.** Assume $a = B^T s$ and let $t = \arg\min_{t \in \mathcal{A}(\hat{\beta})} \|\beta - t\|_{F,*}$, then it holds that

$$
\|a\|_F \text{Dist}_{F,*}^B(\beta, \hat{\beta}) = \|B^T s\|_F \|\beta - t\|_{F,*} \geq \langle B^T s, \beta - t \rangle = \langle s, B\beta - Bt \rangle
$$

$$
= \langle s, B\beta - B\hat{\beta} \rangle = \langle B^T s, \beta - \hat{\beta} \rangle = \langle a, \beta - \hat{\beta} \rangle. \quad \square
$$

**Proof of Theorem 5.4.2:** Recall from (5.25) that for both step-size rules it holds that

$$
\mathbb{E}_{m+1} [L(\beta^{m+1})] \leq L(\beta^m) - \frac{1}{2\sigma}\|\nabla L(\beta^m)\|_F^2,
$$

thus $\beta^m \in \mathcal{L}S_0$. Noticing $\nabla L(\beta^m) \in \text{Range}(B^T)$ and by using Proposition 5.4.7 we have
\[ \mathbb{E}_{m+1}[L(\beta^{m+1})] \leq L(\beta^m) - \frac{(\nabla L(\beta^{m}) - \beta^*)^2}{2\sigma \text{Dist}_0^2} \leq L(\beta^m) - \frac{(\nabla L(\beta^{m}) - \beta^*)^2}{2\sigma \text{Dist}_0^2} \]

\[ \leq L(\beta^m) - \frac{(L(\beta^m) - L(\beta^*))^2}{2\sigma \text{Dist}_0^2}, \]

where the second inequality is because \( \beta^m \in LS_0 \), and the third inequality follows from the convexity of \( L \). Taking expectation with respect to \( \xi_m \), we arrive at

\[ \mathbb{E}_{\xi_{m+1}}[L(\beta^{m+1})] \leq \mathbb{E}_{\xi_m}[L(\beta^m)] - \frac{\mathbb{E}_{\xi_m}[(L(\beta^m) - L(\beta^*))^2]}{2\sigma \text{Dist}_0^2} \]

\[ \leq \mathbb{E}_{\xi_m}[L(\beta^m)] - \frac{(\mathbb{E}_{\xi_m}[L(\beta^m) - L(\beta^*)]^2}{2\sigma \text{Dist}_0^2} \]

Now define \( \delta_m := \mathbb{E}_{\xi_m}[L(\beta^m) - L(\beta^*)] \), then we have \( \delta_m \geq 0 \) and

\[ \delta_{m+1} \leq \delta_m - \frac{\delta_m^2}{2\sigma \text{Dist}_0^2}. \]

Noticing that \( \delta_{m+1} = \mathbb{E}_{\xi_m}[\mathbb{E}_{m+1}[L(\beta^{m+1}) | \xi_m] \leq \mathbb{E}_{\xi_m}[L(\beta^m)] = \delta_m \), we have:

\[ \delta_{m+1} \leq \delta_m - \frac{\delta_m \delta_{m+1}}{2\sigma \text{Dist}_0^2}. \]

Dividing both sides by \( \delta_m \delta_{m+1} \), we arrive at

\[ \frac{1}{\delta_{m+1}} \geq \frac{1}{\delta_m} + \frac{1}{2\sigma \text{Dist}_0^2}, \]

whereby

\[ \frac{1}{\delta_M} \geq \frac{1}{\delta_0} + \frac{M}{2\sigma \text{Dist}_0^2}, \]

which furnishes the proof. \( \square \)
5.5 Numerical Experiments

In this section, we present computational experiments discussing the performance of RGBM for solving classification and regression problems with tree stumps as weak-learners.

Datasets: The datasets we use in the numerical experiments were gathered from the LIBSVM library [26]. Table 6.2 summarizes the basic statistics of these datasets. For each dataset, we randomly choose 80% as the training and the remaining as the testing dataset. To be consistent with our theory, we use a regularized logistic loss with a small parameter $d = 0.0001$ for the classification problems; and $\ell_2$ loss for the regression problem.

<table>
<thead>
<tr>
<th>dataset</th>
<th>task</th>
<th># samples</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>a9a</td>
<td>classification</td>
<td>32561</td>
<td>123</td>
</tr>
<tr>
<td>colon-cancer</td>
<td>classification</td>
<td>62</td>
<td>2000</td>
</tr>
<tr>
<td>rcv1</td>
<td>classification</td>
<td>20242</td>
<td>47236</td>
</tr>
<tr>
<td>YearPrediction</td>
<td>regression</td>
<td>463715</td>
<td>90</td>
</tr>
</tbody>
</table>

Table 5.1: Basic statistics of the (real) datasets used in numerical experiments. The training/testing datasets are obtained by a 80%/20% (random) split on these sample sizes.

RGBM with Tree Stumps: All algorithms consider tree stumps (5.5) as the weak-learners, as described in Section 5.1.2. In our experiments (involving datasets with $n > 10,000$), to reduce the computational cost, we decrease the number of candidate splits for each feature by considering 100 quantiles instead of all $n$ quantiles (corresponding to $n$ samples). (We note that this simply reduces the number of weak learners considered, and our methodological framework applies.) This strategy is commonly used in implementations of GBM e.g, XGBoost [27]. For each feature $g$, we consider the candidate splitting points according to the percentiles of its empirical distribution, thus there are in total $100p$ weak-learners. All the tree stumps that perform a split on one feature is considered a group—leading to $p$ groups. In RGBM, we randomly choose $t$ out of $p$ features and consider the $100t$ features as the set $J$,
among which we pick the best weak-learner to perform an update. The values of $t$ are chosen on a geometrically spaced grid from 1 to $p$ with four values for each dataset. In particular, the case $t = p$ corresponds to traditional GBM.

**Performance Measures:** Figure 5-5 shows the performance of RGBM with different $t$ values. The $x$-axis is the (standardized) computational cost called “Epochs” (cf Figure 5-1) — this corresponds to the number of times the algorithm makes a pass across all the weak-learners. The $y$-axis denotes the quality of solution (or the data-fidelity) obtained, i.e., the objective value, for both the training and testing datasets.

**Comparisons:** For all of the datasets, RGBM with a smaller $t$ value has a better training performance with the same number of epochs. This demonstrates the (often significant) computational gains possible by using RGBM. The colon-cancer dataset is a high-dimensional problem with $p \gg n$; and its training/testing profile is somewhat different from the other datasets. In many examples, we observe that a choice of $t$ in the interior of its range of possible values, leads to a model with best test performance — this empirically suggests that RGBM can lead to better predictive models due to the inherent regularization imparted via the randomization scheme. For datasets with $n \gg p$, the profile of the testing loss is similar to that of the training loss.
Figure 5-5: Plots showing the training and testing loss versus number of standardized iterations aka epochs for four different datasets. We consider RGBM for different $t$ values (with the largest corresponding to GBM). The general observations are similar to that in Figure 5-1 — we get significant computational savings by using a smaller value of $t$, without any loss in training/testing error.
Chapter 6

Accelerated Gradient Boosting Machine

Gradient Boosting Machine (GBM) [41] is an extremely powerful supervised learning algorithm that is widely used in practice. GBM routinely features as a leading algorithm in machine learning competitions such as Kaggle and the KDDCup [27]. In this work, we propose Accelerated Gradient Boosting Machine (AGBM) by incorporating Nesterov’s acceleration techniques into the design of GBM. The difficulty in accelerating GBM lies in the fact that weak (inexact) learners are commonly used, and therefore the errors can accumulate in the momentum term. To overcome it, we design a “corrected pseudo residual” and fit best weak learner to this corrected pseudo residual, in order to perform the z-update. Thus, we are able to derive novel computational guarantees for AGBM. This is the first GBM type of algorithm with theoretically-justified accelerated convergence rate. Finally, we demonstrate with a number of numerical experiments the effectiveness of AGBM over conventional GBM in obtaining a model with good training and/or testing data fidelity.
6.1 Introduction

Gradient Boosting Machine (GBM) [41] is a powerful supervised learning algorithm that combines multiple weak-learners into an ensemble with excellent predictive performance. GBM works very well for a number of tasks like spam filtering, online advertising, fraud detection, anomaly detection, computational physics (e.g., the Higgs Boson discovery), etc; and has routinely featured as a top algorithm in Kaggle competitions and the KDDCup [27]. GBM can naturally handle heterogeneous datasets (highly correlated data, missing data, categorical data, etc). It is also quite easy to use with several publicly available implementations: scikit-learn [84], R gbm [93], LightGBM [52], XGBoost [27], TF Boosted Trees [88], etc.

In spite of the practical success of GBM, there is a considerable gap in its theoretical understanding. The traditional interpretation of GBM is to view it as a form of steepest descent in functional space [67, 41]. While this interpretation serves as a good starting point, such framework lacks rigorous non-asymptotic convergence guarantees, especially when compared to the growing body of literature on first order convex optimization.

In convex optimization literature, Nesterov’s acceleration is a successful technique to speed up the convergence of first-order methods. In this work, we show how to incorporate Nesterov momentum into the gradient boosting framework in order to obtain an accelerated gradient boosting machine.

6.1.1 Our contributions

We propose the first accelerated gradient boosting algorithm that comes with strong theoretical guarantees and can be used with any type of weak learner. In particular:

- We propose a novel accelerated gradient boosting algorithm (AGBM) (Section 6.3) and prove (Section 6.4) that it reduces the empirical loss at a rate
of $O(1/m^2)$ after $m$ iterations, improving upon the $O(1/m)$ rate obtained by traditional gradient boosting methods.

- We propose a variant of AGBM, taking advantage of strong convexity of loss function, which achieves linear convergence (Section 6.5). We also list the conditions (on the loss function) under which AGBMs would be beneficial.

- With a number of numerical experiments with weak tree learners (one of the most popular type of GBMs) we confirm the effectiveness of AGBM.

Apart from theoretical contributions, we paved the way for speeding up some practical applications of GBMs, which currently require a large number of boosting iterations. For example, GBMs with boosted trees for multi-class problems are commonly implemented as a number of one-vs-rest learners, resulting in more complicated boundaries [38] and a potentially a larger number of boosting iterations required. Additionally, it is a common practice to build many very weak learners for problems where it is easy to overfit. Such large ensembles result not only in slow training time, but also slower inference. AGBMs can be potentially beneficial for all these applications.

6.1.2 Related Literature

Convergence Guarantees for GBM: After being first introduced by Friedman et al. [41], several works established its guaranteed convergence, without explicitly stating the convergence rate [28, 67]. Subsequently, when the loss function is both smooth and strongly convex, [13] proved an exponential convergence rate—more precisely that $O(\exp(1/\varepsilon^2))$ iterations are sufficient to ensure that the training loss is within $\varepsilon$ of its optimal value. [102] then studied the primal-dual structure of GBM and demonstrated that in fact only $O(\log(1/\varepsilon))$ iterations are needed. However the constants in their rate were non-standard and less intuitive. This result was recently improved upon by [35] and [66], who showed a similar convergence rate but with more transparent constants such as the smoothness and strong convexity constant of the loss function, as well as the density of weak learners. Additionally, if the loss function
is assumed to be smooth and convex (but not necessarily strongly convex), [66] also showed that $O(1/\varepsilon)$ iterations are sufficient. We refer the reader to [102], [35] and [66] for a more detailed literature review of the theoretical results of GBM convergence.

**Accelerated Gradient Methods:** For optimizing a smooth convex function, [75] showed that the standard gradient descent algorithm can be made much faster, resulting in the *accelerated* gradient descent method. While gradient descent requires $O(1/\varepsilon)$ iterations, accelerated gradient methods only require $O(1/\sqrt{\varepsilon})$. In general, this rate of convergence is optimal and cannot be improved upon [77]. Since its introduction in 1983, the mainstream research community’s interest in Nesterov’s accelerated method started around 15 years ago; yet even today most researchers struggle to find basic intuition as to what is really going on in accelerated methods. Such lack of intuition about the estimation sequence proof technique used by [77] has motivated many recent works trying to explain this acceleration phenomenon [99, 109, 48, 59, 43, 2, 18]. Some have recently attempted to give a physical explanation of acceleration techniques by studying the continuous-time interpretation of accelerated gradient descent via dynamical systems [99, 109, 48].

**Accelerated Greedy Coordinate and Matching Pursuit Methods:** Recently, [60] and [63] discussed how to accelerate matching pursuit and greedy coordinate descent algorithms respectively. Their methods however require a random step and are hence only ‘semi-greedy’, which does not fit in the boosting framework.

**Accelerated GBM:** Recently, [12] and [33] proposed accelerated versions of GBM by directly incorporating Nesterov’s momentum in GBM, however, no theoretical justification was provided. Furthermore, as we argue in Section 6.5.2, their proposed algorithm may not converge to the optimum.

### 6.2 Gradient Boosting Machine

We consider a supervised learning problem with $n$ training examples $(x_i, y_i), i = 1, \ldots, n$ such that $x_i \in \mathbb{R}^p$ is the feature vector of the $i$-th example and $y_i$ is a label
(in a classification problem) or a continuous response (in a regression problem). In
the classical version of GBM [41], we assume we are given a base class of learners
\( \mathcal{B} \) and that our target function class is the linear combination of such base learners
(denoted by \( \text{lin}(\mathcal{B}) \)). Let \( \mathcal{B} = \{ b_\tau(x) \in \mathbb{R} \} \) be a family of learners parameterized by
\( \tau \in \mathcal{T} \). The prediction corresponding to a feature vector \( x \) is given by an additive
model of the form:

\[
    f(x) := \left( \sum_{m=1}^{M} \beta_m b_{\tau_m}(x) \right) \in \text{lin}(\mathcal{B}),
\]

where \( b_{\tau_m}(x) \in \mathcal{B} \) is a weak-learner and \( \beta_m \) is its corresponding additive coefficient.
Here, \( \beta_m \) and \( \tau_m \) are chosen in an adaptive fashion in order to improve the data-fidelity
as discussed below. Examples of learners commonly used in practice include wavelet
functions, support vector machines, and classification and regression trees [39]. We
assume the set of weak learners \( \mathcal{B} \) is scalable, namely that the following assumption
holds.

**Assumption 6.2.1.** If \( b(\cdot) \in \mathcal{B} \), then \( \lambda b(\cdot) \in \mathcal{B} \) for any \( \lambda > 0 \).

Assumption 6.2.1 holds for most of the set of weak learners we are interested in.
Indeed scaling a weak learner is equivalent to modifying the coefficient of the weak
learner, so it does not change the structure of \( \mathcal{B} \).

The goal of GBM is to obtain a good estimate of the function \( f \) that approximately
minimizes the empirical loss:

\[
    L^* = \min_{f \in \text{lin}(\mathcal{B})} \left\{ L(f) := \sum_{i=1}^{n} \ell(y_i, f(x_i)) \right\}
\]

where \( \ell(y_i, f(x_i)) \) is a measure of the data-fidelity for the \( i \)-th sample for the loss
function \( \ell \).

### 6.2.1 Best Fit Weak Learners

The original version of GBM by [41], presented in Algorithm 13, can be viewed
as minimizing the loss function by applying an approximated steepest descent algo-
GBM starts from a null function $f^0 = 0$ and at each iteration computes the pseudo-residual $r^m$ (namely, the negative gradient of the loss function with respect to the predictions so far $f^m$):

$$r^m_i = - \frac{d \ell(y_i, f^m(x_i))}{df^m(x_i)}.$$  

(6.3)

Then a weak-learner that best fits the current pseudo-residual in terms of the least squares loss is computed as follows:

$$\tau_m = \arg \min_{\tau \in \mathcal{T}} \sum_{i=1}^n (r^m_i - \tau(x_i))^2.$$  

(6.4)

This weak-learner is added to the model with a coefficient found via a line search.

As the iterations progress, GBM leads to a sequence of functions $\{f^m\}_{m \in [M]}$ (where $[M]$ is a shorthand for the set $\{1, \ldots, M\}$). The usual intention of GBM is to stop early—before one is close to a minimum of Problem (6.2)—with the hope that such a model will lead to good predictive performance [41, 35, 116, 19].

**Algorithm 13** Gradient Boosting Machine (GBM) [41]

**Initialization.** Initialize with $f^0(x) = 0$.

For $m = 0, \ldots, M - 1$ do:

**Perform Updates:**

1. Compute pseudo residual: $r^m = - \left[ \frac{\partial \ell(y_i, f^m(x_i))}{\partial f^m(x_i)} \right]_{i=1}^n$.
2. Find the parameters of the best weak-learner: $\tau_m = \arg \min_{\tau \in \mathcal{T}} \sum_{i=1}^n (r^m_i - \tau(x_i))^2$.
3. Choose the step-size $\eta_m$ by line-search: $\eta_m = \arg \min_{\eta} \sum_{i=1}^n \ell(y_i, f^m(x_i) + \eta b_{\tau_m}(x_i))$.
4. Update the model $f^{m+1}(x) = f^m(x) + \eta_m b_{\tau_m}(x)$.

**Output.** $f^M(x)$.

Perhaps the most popular set of learners are classification and regression trees (CART) [16], resulting in Gradient Boosted Decision Tree models (GBDTs). These are the models that we are using for our numerical experiments. At the same time, we would like to highlight that our algorithm is not tied to a particular type of a weak learner and is a general algorithm.
Given the success of accelerated gradient descent as a first order optimization method, it seems natural to attempt to accelerate the GBMs. As a warm-up, we first look at how to obtain an accelerated boosting algorithm when our class of learners $\mathcal{B}$ is strong (complete) and can exactly fit any pseudo-residuals. This assumption is quite unreasonable but will serve to understand the connection between boosting and first order optimization. We then describe our actual algorithm which works for any class of weak learners.

### 6.3.1 Boosting with strong learners

In this subsection, we assume the class of learners $\mathcal{B}$ is strong, i.e. for any pseudo-residual $r \in \mathbb{R}^n$, there exists a learner $b(x) \in \mathcal{B}$ such that

$$b(x_i) = r_i, \forall i \in [n].$$

Of course the entire point of boosting is that the learners are weak and thus the class is not strong, therefore this is not a realistic assumption. Nevertheless this section will provide the intuitions on how to develop AGBM.

In the GBM we compute the pseudo-residual $r^m$ in (6.3) to be the negative gradient of the loss function over the predictions so far. A gradient descent step in a functional
space would try to find $f^{m+1}$ such that for $i \in \{1, \ldots, n\}$

$$f^{m+1}(x_i) = f^m(x_i) + \eta r_i^m.$$  

Here $\eta$ is the step-size of our algorithm. Since our class of learners is rich, we can choose $b^m(x) \in B$ to exactly satisfy the above equation.

Thus GBM (Algorithm 13) then has the following update:

$$f^{m+1} = f^m + \eta b^m,$$

where $b^m(x_i) = r_i^m$. In other words, GBM performs exactly functional gradient descent when the class of learners is strong, and so it converges at a rate of $O(1/m)$. Akin to the above argument, we can perform functional accelerated gradient descent, which has the accelerated rate of $O(1/m^2)$. In the accelerated method, we maintain three model ensembles: $f$, $g$, and $h$ of which $f(x)$ is the only model which is finally used to make predictions during the inference time. Ensemble $h(x)$ is the momentum sequence and $g(x)$ is a weighted average of $f$ and $h$ (refer to Table 6.1 for list of all notations used). These sequences are updated as follows for a step-size $\eta$ and $\{\theta_m = 2/(m + 2)\}$:

$$g^m = (1 - \theta_m)f^m + \theta_m h^m$$

$$f^{m+1} = g^m + \eta b^m : \text{primary model}$$  \hspace{1cm} (6.5)

$$h^{m+1} = h^m + \eta/\theta_m b^m : \text{momentum model}$$

where $b^m(x)$ satisfies for $i \in 1, \ldots, n$

$$b^m(x_i) = -\frac{d \ell(y_i, g^m(x_i))}{dg^m(x_i)}.$$  \hspace{1cm} (6.6)

Note that the psuedo-residual is computed w.r.t. $g$ instead of $f$. The update above can be rewritten as

$$f^{m+1} = f^m + \eta b^m + \theta_m(h^m - f^m).$$
If \( \theta_m = 0 \), we see that we recover the standard functional gradient descent with step-size \( \eta \). For \( \theta_m \in (0, 1] \), there is an additional momentum in the direction of \( (h^m - f^m) \).

The three sequences \( f, g, \) and \( h \) match exactly those used in typical accelerated gradient descent methods (see [77, 104] for details).

### 6.3.2 Boosting with weak learners

In this subsection, we consider the general case without assuming that the class of learners is strong. Indeed, the class of learners \( \mathcal{B} \) is usually quite simple and it is very likely that for any \( \tau \in \mathcal{T} \), it is impossible to exactly fit the residual \( r^m \). We call this case boosting with weak learners. Our task then is to modify (6.5) to obtain a truly accelerated gradient boosting machine.

#### Algorithm 14 Accelerated Gradient Boosting Machine (AGBM)

**Input.** Starting function \( f^0(x) = 0 \), step-size \( \eta \), momentum-parameter \( \gamma \in (0, 1] \), and data \( X, y = (x_i, y_i)_{i \in [n]} \).

**Initialization.** \( h^0(x) = f^0(x) \) and sequence \( \theta_m = \frac{2}{m+2} \).

For \( m = 0, \ldots, M - 1 \) do:

**Perform Updates:**

1. Compute a linear combination of \( f \) and \( h \): \( g^m(x) = (1 - \theta_m)f^m(x) + \theta_m h^m(x) \).
2. Compute pseudo residual: \( r^m = -\left[ \sum_{i=1}^{n} \frac{\partial_s[y_i, g^m(x_i)]}{\partial y_i(x_i)} \right] \).
3. Find the best weak-learner for pseudo residual: \( \tau_{m,1} = \arg \min_{\tau \in \mathcal{T}} \sum_{i=1}^{n} (r^m_i - b^m_\tau(x_i))^2 \).
4. Update the model: \( f^{m+1}(x) = g^m(x) + \eta b^m_{\tau_{m,1}}(x) \).
5. Update the corrected residual: \( c^m_i = \begin{cases} r^m_i & \text{if } m = 0 \\ r^m_i + \frac{m+1}{m+2}(c^m_{i-1} - b^m_{\tau_{m-1,2}}(x_i)) & \text{o.w.} \end{cases} \).
6. Find the best weak-learner for the corrected residual: \( \tau_{m,2} = \arg \min_{\tau \in \mathcal{T}} \sum_{i=1}^{n} (c^m_i - b^m_\tau(x_i))^2 \).
7. Update the momentum model: \( h^{m+1}(x) = h^m(x) + \gamma \eta / \theta_m b^m_{\tau_{m,2}}(x) \).

**Output.** \( f^M(x) \).

The full details are summarized in Algorithm 14 but we will highlight two key differences from (6.5).
First, the update to the $f$ sequence is replaced with a weak-learner which best approximates $r^m$ similar to (6.5). In particular, we compute pseudo-residual $r^m$ computed w.r.t. $g$ as in (6.6) and find a parameter $\tau_{m,1}$ such that

$$\tau_{m,1} = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^{n} (r^m_i - b_r(x_i))^2.$$  

Secondly, and more crucially, the update to the momentum model $h$ is decoupled from the update to the $f$ sequence. We use an error-corrected pseudo-residual $c^m$ instead of directly using $r^m$. Suppose that at iteration $m-1$, a weak-learner $b_{r_{m-1},2}$ was added to $h^{m-1}$. Then error corrected residual is defined inductively as follows: for $i \in \{1, \ldots, n\}$

$$c^m_i = r^m_i + \frac{m+1}{m+2} (c^m_{i-1} - b_{r_{m-1},2}(x_i)),$$

and then we compute

$$\tau_{m,2} = \arg\min_{\tau \in \mathcal{T}} \sum_{i=1}^{n} (c^m_i - b_r(x_i))^2.$$  

Thus at each iteration two weak learners are computed—$b_{r_{m,1}}(x)$ approximates the residual $r^m$ and the $b_{r_{m,2}}(x)$, which approximates the error-corrected residual $c^m$. Note that if our class of learners is complete then $c^m_{i-1} = b_{r_{m-1},2}(x_i)$, $c^m = r^m$ and $\tau_{m,1} = \tau_{m,2}$. This would revert back to our accelerated gradient boosting algorithm for strong-learners described in (6.5).

6.4 Convergence Analysis of AGBM

We first formally define the assumptions required and then outline the computational guarantees for AGBM.
6.4.1 Assumptions

Let’s introduce some standard regularity/continuity constraints on the loss function that we require in our analysis.

Definition 6.4.1. We denote \( \frac{\partial \ell(y, f)}{\partial f} \) as the derivative of the bivariant loss function \( \ell \) w.r.t. the prediction \( f \). We say that \( \ell \) is \( \sigma \)-smooth if for any \( y \) and predictions \( f_1 \) and \( f_2 \), it holds that

\[
\ell(y, f_1) \leq \ell(y, f_2) + \frac{\partial \ell(y, f_2)}{\partial f}(f_1 - f_2) + \frac{\sigma}{2}(f_1 - f_2)^2.
\]

We say \( \ell \) is \( \mu \)-strongly convex (with \( \mu > 0 \)) if for any \( y \) and predictions \( f_1 \) and \( f_2 \), it holds that

\[
\ell(y, f_1) \geq \ell(y, f_2) + \frac{\partial \ell(y, f_2)}{\partial f}(f_1 - f_2) + \frac{\mu}{2}(f_1 - f_2)^2.
\]

Note that \( \mu \leq \sigma \) always. Smoothness and strong-convexity mean that the function \( \ell(x) \) is (respectively) upper and lower bounded by quadratic functions. Intuitively, smoothness implies that that gradient does not change abruptly and hence \( \ell(x) \) is never ‘sharp’. Strong-convexity implies that \( \ell(x) \) always has some ‘curvature’ and is never ‘flat’.

The notion of Minimal Cosine Angle (MCA) introduced in \[66\] plays a central role in our convergence rate analysis of GBM. MCA measures how well the weak-learner \( b_{\tau(r)}(X) \) approximates the desired residual \( r \).

Definition 6.4.2. Let \( r \in \mathbb{R}^n \) be a vector. The Minimal Cosine Angle (MCA) is defined as the similarity between \( r \) and the output of the best-fit learner \( b_{\tau}(X) \):

\[
\Theta := \min_{r \in \mathbb{R}^n} \max_{\tau \in \mathcal{T}_m} \cos(r, b_{\tau}(X)),
\]

where \( b_{\tau}(X) \in \mathbb{R}^n \) is a vector of predictions \( [b_{\tau}(x_i)]_i \).

The quantity \( \Theta \in (0, 1] \) measures how “dense” the learners are in the prediction space. For strong learners (in Section 6.3.1), the prediction space is complete, and
\( \Theta = 1 \). For a complex space of learners \( T \) such as deep trees, we expect the prediction space to be dense and that \( \Theta \approx 1 \). For a simpler class such as tree-stumps \( \Theta \) would be much smaller. Refer to [66] for a discussion of \( \Theta \).

### 6.4.2 Computational Guarantees

We are now ready to state the main theoretical result of this chapter.

**Theorem 6.4.1.** Consider Accelerated Gradient Boosting Machine (Algorithm 14). Suppose \( f \) is \( \sigma \)-smooth, the step-size \( \eta \leq \frac{1}{\sigma} \) and the momentum parameter \( \gamma \leq \Theta^4/(4+\Theta^2) \), where \( \Theta \) is the MCA introduced in Definition 6.4.2. Then for all \( M \geq 0 \), we have:

\[
L(f^M) - L(f^*) \leq \frac{1}{2\eta\gamma(M+1)^2} \left\| f^*(X) \right\|_2^2.
\]

Before showing the full proof, we describe a proof sketch to highlight the major idea behind the proofs.

**Proof Sketch.** Here we only give an outline—the full proof can be found in the Appendix (Section ??). We use the potential-function based analysis of accelerated method (cf. [104, 109]). Recall that \( \theta_m = \frac{2^{m+2}}{m+2} \). For the proof, we introduce the following vector sequence of auxiliary ensembles \( \hat{h} \) as follows:

\[
\hat{h}^0(X) = 0, \quad \hat{h}^{m+1}(X) = \hat{h}^m(X) + \frac{\eta\gamma}{\theta_m} r^m.
\]

The sequence \( \hat{h}^m(X) \) is in fact closely tied to the sequence \( h^m(X) \) as we demonstrate in the Appendix (Lemma 6.4.2). Let \( f^* \) be any function which obtains the optimal loss (6.2)

\[
f^* \in \arg\min_{f \in \text{lin}(S)} \left\{ L(f) := \sum_{i=1}^n \ell(y_i, f(x_i)) \right\}.
\]
Let us define the following sequence of potentials:

\[
V^m(f^*) = \begin{cases} 
\frac{1}{2} \| f^*(X) - \hat{h}^0(X) \|^2 & \text{if } m = 0, \\
\frac{m}{\theta^2_{m-1}} (L(f^m) - L^*) + \frac{1}{2} \| f^*(X) - \hat{h}^m(X) \|^2 & \text{o.w}
\end{cases}
\]

Typical proofs of accelerated algorithms show that the potential \( V^m(f^*) \) is a decreasing sequence. In boosting, we use the weak-learner that fits the pseudo-residual of the loss. This can guarantee sufficient decay to the first term of \( V^m(f^*) \) related to the loss \( L(f) \). However, there is no such guarantee that the same weak-learner can also provide sufficient decay to the second term as we do not apriori know the optimal ensemble \( f^* \). That is the major challenge in the development of AGBM.

We instead show that the potential decreases at least by \( \delta_m \):

\[
V^{m+1}(f^*) \leq V^m(f^*) + \delta_m,
\]

where \( \delta_m \) is an error term depending on \( \Theta \) (see Lemma 6.4.4 for the exact definition of \( \delta_m \) and proof of the claim). By telescope, it holds that

\[
\frac{\eta \gamma}{\theta^2_m} (L(f^{m+1}) - L(f^*)) \leq V^{m+1}(f^*) \\
\leq \sum_{j=0}^m \delta_j + \frac{1}{2} \| f^*(X) - \hat{h}^0(X) \|^2.
\]

Finally a careful analysis of the error term (Lemma 6.4.6) shows that \( \sum_{j=0}^m \delta_j \leq 0 \) for any \( m \geq 0 \). Therefore,

\[
L(f^{m+1}) - L(f^*) \leq \frac{\theta^2_m}{2\eta \gamma} \| f^*(X) \|^2,
\]

which furnishes the proof by letting \( m = M - 1 \) and substituting the value of \( \theta_m \). \( \square \)

**Remark 6.4.1.** Theorem 6.4.1 implies that to get a function \( f^M \) such that the error \( L(f^M) - L(f^*) \leq \varepsilon \), we need number of iterations \( M \sim O \left( \frac{1}{\Theta^2 \sqrt{\varepsilon}} \right) \). In contrast, standard gradient boosting machines, as proved in [66], require \( M \sim O \left( \frac{1}{\Theta \varepsilon} \right) \). This
means that for small values of $\varepsilon$, AGBM (Algorithm 14) can require far fewer weak learners than GBM (Algorithm 13).

Now we prove Theorem 6.4.1. Let’s start with some new notations. Define scalar constants $s = \gamma/\Theta^2$ and $t := (1 - s)/2 \in (0, 1)$. We mostly only need $s + t \leq 1$—the specific values of $\gamma$ and $t$ are needed only in Lemma 6.4.6. Then define

$$\alpha_m := \frac{\eta \gamma}{\theta_m} = \frac{\eta s \Theta^2}{\theta_m},$$

then the definitions of the sequences $\{r^m\}$, $\{c^m\}$, $\hat{h}^m(X)$ and $\{\theta_m\}$ from Algorithm 3 can be simplified as:

$$\theta_m = \frac{2}{m + 2},$$

$$r^m = -\left[ \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)} \right]_{i=1...n},$$

$$c^m = r^m + (\alpha_{m-1}/\alpha_m) \left( c^{m-1} - b_{r_{m-1}}(X) \right),$$

$$\hat{h}^{m+1}(X) = \hat{h}^m(X) + \alpha_m r^m.$$  

The sequence $\hat{h}^m(X)$ is in fact closely tied to the sequence $h^m(X)$ as we show in the next lemma. For notational convenience, we define $c^{-1} = b_{r_{-1}}(X) = 0$ and similarly $\alpha^{-1} = \theta_{-1} = 0$ throughout the proof.

**Lemma 6.4.1.**

$$\hat{h}^{m+1}(X) = h^{m+1}(X) + \alpha_m (c_m - b_{r_{m,2}}(X)).$$

**Proof.** Observe that

$$\hat{h}^{m+1}(X) = \sum_{j=0}^m \alpha_j r^j \quad \text{and that} \quad h^{m+1}(X) = \sum_{j=0}^m \alpha_j b_{r_{j,2}}(X).$$

174
Then we have
\[
\hat{h}^{m+1}(X) - h^{m+1}(X) = \sum_{j=0}^{m} \alpha_j (r^j - b_{r_j,2}(X))
\]
\[
= \sum_{j=0}^{m} \alpha_j (r^j - \frac{\alpha_{j-1}}{\alpha_j} b_{r_{j-1}}(X)) - \alpha_m b_{r_m,2}(X)
\]
\[
= \sum_{j=0}^{m} \alpha_j (c^j - \frac{\alpha_{j-1}}{\alpha_j} c^{j-1}) - \alpha_m b_{r_m,2}(X)
\]
\[
= \sum_{j=0}^{m} (\alpha_j c^j - \alpha_{j-1} c^{j-1}) - \alpha_m b_{r_m,2}(X)
\]
\[
= \alpha_m (c_m - b_{r_m,2}(X)) ,
\]
where the third equality is due to the definition of $c^m$.

Lemma 6.4.2 presents the fact that there is sufficient decay of the loss function:

**Lemma 6.4.2.**

\[
L(f^{m+1}) \leq L(g^m) - \frac{\eta \Theta^2}{2} \| r^m \|^2 .
\]

**Proof.** Recall that $\tau_{m,1}$ is chosen such that

\[
\tau_{m,1} = \arg\min_{\tau \in T} \| b_\tau(X) - r^m \|^2 .
\]

Since the class of learners $T$ is scalable (Assumption 6.2.1), we have

\[
\| b_{\tau_{m,1}}(X) - r^m \|^2 = \min_{\tau \in T} \| b_\tau(X) - r^m \|^2
\]
\[
= \| r^m \|^2 \left( 1 - \arg\max_{\tau \in T} \cos(r^m, b_\tau(X))^2 \right)
\]
\[
\leq \| r^m \|^2 \left( 1 - \Theta^2 \right) ,
\]

where the last inequality is because of the definition of $\Theta$, and the second equality is
due to the simple fact that for any two vectors $a$ and $b$,

$$\min_{\sigma \in \mathbb{R}} \|\sigma a - b\|^2 = \|a\|^2 - \max_{\sigma \in \mathbb{R}} \left( \sigma \langle a, b \rangle - \frac{\sigma^2}{2} \|b\|^2 \right) = \|a\|^2 - \|a\|^2 \frac{\langle a, b \rangle}{\|a\|^2 \|b\|^2}.$$  

Now recall that $L(f^{m+1}) = \sum_{i=1}^{n} l(y_i, f^{m+1}(x_i))$ and that $f^{m+1}(x) = g^m(x) + \eta b_{\tau_m}(x)$. Since the loss function $l(y_i, x)$ is $\sigma$-smooth and step-size $\eta \leq \frac{1}{\sigma}$, it holds that

$$L(f^{m+1}) = \sum_{i=1}^{n} l(y_i, f^{m+1}(x_i))$$

$$\leq \sum_{i=1}^{n} l(y_i, g^m(x_i) + \eta b_{\tau_m}(x_i))$$

$$\leq \sum_{i=1}^{n} \left( l(y_i, g^m(x_i)) + \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)} (\eta b_{\tau_m}(x_i)) + \frac{\sigma}{2} (\eta b_{\tau_m}(x_i))^2 \right)$$

$$\leq \sum_{i=1}^{n} \left( l(y_i, g^m(x_i)) + \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)} (\eta b_{\tau_m}(x_i)) + \frac{\eta}{2} (b_{\tau_m}(x_i))^2 \right)$$

$$= \sum_{i=1}^{n} \left( l(y_i, g^m(x_i)) - r_i^m (\eta b_{\tau_m}(x_i)) + \frac{1}{2\eta} (b_{\tau_m}(x_i))^2 \right)$$

$$= L(g^m) - \eta \langle r^m, b_{\tau_m}(X) \rangle + \frac{\eta}{2} \|b_{\tau_m}(X)\|^2$$

$$= L(g^m) + \frac{\eta}{2} \|b_{\tau_m}(X)\|^2 - \frac{\eta}{2} \|r^m\|^2$$

$$\leq L(g^m) - \frac{\Theta^2 \eta}{2} \|r^m\|^2,$$

where the final inequality follows from (6.8). This furnishes the proof of the lemma. 

Lemma 6.4.3 is a basic fact of convex function, and it is commonly used in the convergence analysis in accelerated method.

Lemma 6.4.3. For any function $f$ and $m \geq 0$,

$$L(g^m) + \theta_m \langle r^m, h^m(X) - f(X) \rangle \leq \theta_m L(f) + (1 - \theta_m) L(f^m).$$
Proof. For any function \( f \), it follows from the convexity of the loss function \( l \) that

\[
L(g^m) + \langle r^m, g^m(X) - f(X) \rangle = \sum_{i=1}^{n} l(y_i, g^m(x_i)) + \frac{\partial l(y_i, g^m(x_i))}{\partial g^m(x_i)} (f(x_i) - g^m(x_i))
\leq \sum_{i=1}^{n} l(y_i, f(x_i)) = L(f).
\] (6.9)

Substituting \( f = f^m \) in (6.9), we get

\[
L(g^m) + \langle r^m, g^m(X) - f^m(X) \rangle \leq L(f^m). \tag{6.10}
\]

Also recall that \( g^m(X) = (1 - \theta_m)f^m(X) + \theta_m h^m(X) \). This can be rewritten as

\[
\theta_m(g^m(X) - h^m(X)) = (1 - \theta_m)(f^m(X) - g^m(X)). \tag{6.11}
\]

Putting (6.9), (6.10), and (6.11) together:

\[
L(g^m) + \theta_m \langle r^m, h^m(X) - f(X) \rangle \\
= L(g^m) + \theta_m \langle r^m, g^m(X) - f(X) \rangle + \theta_m \langle r^m, h^m(X) - g^m(X) \rangle \\
= \theta_m[L(g^m) + \langle r^m, g^m(X) - f(X) \rangle] + (1 - \theta_m)[L(g^m) + \langle r^m, g^m(X) - f^m(X) \rangle] \\
\leq \theta_m L(f) + (1 - \theta_m)L(f^m),
\]

which furnishes the proof. \( \square \)

We are ready to prove the key lemma which gives us the accelerated rate of convergence.

**Lemma 6.4.4.** Define the following potential function \( V(f) \) for any given output function \( f \):

\[
V^m(f) = \frac{\alpha_{m-1}}{\theta_{m-1}} (L(f^m) - L(f)) + \frac{1}{2} \| f(X) - h^m(X) \|^2. \tag{6.12}
\]
At every step, the potential decreases at least by \( \delta_m \):

\[
V^{m+1}(f) \leq V^m(f) + \delta_m,
\]

where \( \delta_m \) is defined as:

\[
\delta_m := \frac{s\alpha_{m-1}^2}{2t} \|c^{m-1} - b_{r_{m-1}}(X)\|^2 - (1 - s - t)\alpha_m^2 \|r^m\|^2.
\] (6.13)

Proof. Recall that \( c^{-1} = b_{r_{k-1}}(X) \) = 0 and \( \frac{\alpha_{k-1}}{\theta_{k-1}} \) = 0. It follows from Lemma 6.4.2 that:

\[
L(f^{m+1}) - L(g^m) + \frac{(1 - s)\eta \Theta^2}{2} \|r^m\|^2 \\
\leq -\frac{s\eta \Theta^2}{2} \|r^m\|^2 \\
= -\alpha_m \theta_m \|r^m\|^2 + \frac{\alpha_m \theta_m}{2} \|r^m\|^2 \\
= \theta_m \langle r^m, \hat{h}^m(X) - \hat{h}^{m+1}(X) \rangle + \frac{\theta_m}{2\alpha_m} \|\hat{h}^m(X) - \hat{h}^{m+1}(X)\|^2 \\
= \theta_m \langle r^m, \hat{h}^m(X) - f(X) \rangle + \frac{\theta_m}{2\alpha_m} \left( \|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right),
\]

where the second equality is by the definition of \( \hat{h}^m(x) \) and the third is just mathematical manipulation of the equation (it is also called three-point property). By rearranging the above inequality, we have

\[
L(f^{m+1}) + \frac{(1 - s)\eta \Theta^2}{2} \|r^m\|^2 \\
\leq L(g^m) + \langle r^m, \hat{h}^m(X) - f(X) \rangle + \frac{\theta_m}{2\alpha_m} \left( \|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right) \\
= L(g^m) + \theta_m \langle r^m, h^m(X) - f(X) \rangle + \frac{\theta_m}{2\alpha_m} \left( \|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right) + \theta_m \langle r^m, \hat{h}^m(X) - h^m \rangle. \\
\leq \theta_m L(f) + (1 - \theta_m)L(f^m) + \frac{\theta_m}{2\alpha_m} \left( \|f(X) - \hat{h}^m(X)\|^2 - \|f(X) - \hat{h}^{m+1}(X)\|^2 \right) + \theta_m \alpha_{m-1} \langle r^m, c^{m-1} - b_{r_{m-1}}(X) \rangle
\]

where the first inequality uses Lemma 6.4.3 and the last inequality is due to the fact that \( \hat{h}^m(X) - h^m(X) = \alpha_{m-1}(c^{m-1} - b_{r_{m-1}}(X)) \) from Lemma 6.4.1. Rearranging the
terms and multiplying by \((\alpha_m/\theta_m)\) leads to

\[
\frac{\alpha_m}{\theta_m} (L(f^{m+1}) - L(f)) + \frac{1}{2} \|f(X) - \hat{h}^{m+1}(X)\|^2 \\
\leq \frac{\alpha_m(1 - \theta_m)}{\theta_m} (L(f^{m}) - L(f)) + \frac{1}{2} \|f(X) - \hat{h}^m(X)\|^2 + \alpha_m \alpha_{m-1} \left( r^m, (c^{m-1} - b_{r_{m-1}}(X)) \right) - \frac{(1 - s)\eta \Theta^2 \alpha_m}{2\theta_m} \|r^m\|^2.
\]

Let us examine first the term \(A\):

\[
\frac{\alpha_m(1 - \theta_m)}{\theta_m} = (\eta \Theta^2 s) \frac{1}{\theta_m^2} \leq (\eta \Theta^2 s) \frac{1}{\theta_{m-1}^2} = \frac{\alpha_{m-1}}{\theta_{m-1}}.
\]

We have thus far shown that

\[V^{m+1}(f) \leq V^m(f) + B,
\]

and we now need to show that \(B \leq \delta_m\). Using Mean-Value inequality, the first term in \(B\) can be bounded as

\[
\alpha_m \alpha_{m-1} \left( r^m, (c^{m-1} - b_{r_{m-1}}(X)) \right) \leq \frac{\alpha_m^2 \eta}{2s} \|r^m\|^2 + \frac{\alpha_{m-1} s}{2t} \|c^{m-1} - b_{r_{m-1}}(X)\|^2.
\]

Substituting it in \(B\) shows:

\[
B = \alpha_m \alpha_{m-1} \left( r^m, (c^{m-1} - b_{r_{m-1}}(X)) \right) - \frac{(1 - s)\eta \Theta^2 \alpha_m}{2\theta_m} \|r^m\|^2 \\
\leq \frac{\alpha_m^2 \eta}{2s} \|r^m\|^2 + \frac{\alpha_{m-1} s}{2t} \|c^{m-1} - b_{r_{m-1}}(X)\|^2 - \frac{(1 - s)\eta \Theta^2 \alpha_m}{2\theta_m} \|r^m\|^2 \\
= \frac{\alpha_{m-1} s}{2t} \|c^{m-1} - b_{r_{m-1}}(X)\|^2 - (1 - s - t) \frac{\alpha_m^2 \eta}{2s} \|r^m\|^2 \\
= \delta_m,
\]

which finishes the proof.

Unlike the typical proofs of accelerated algorithms, which usually shows that the potential \(V^m(f)\) is a decreasing sequence, there is no guarantee that the potential \(V^m(f)\) is decreasing in the boosting setting due to the use of weak learners. Instead,
we are able to prove that:

**Lemma 6.4.5.** For any given \( m \), it holds that \( \sum_{j=0}^{m} \delta_j \leq 0 \).

**Proof.** We can rewrite the statement of the lemma as:

\[
\sum_{j=0}^{m-1} \alpha_j^2 \| c^j - b_{r_j,2}(X) \|^2 \leq \frac{t(1 - s - t)}{s^2} \sum_{j=0}^{m} \alpha_j^2 \| r^j \|^2. \tag{6.14}
\]

Here, let us focus on the term \( \| c^{j+1} - b_{r_{j+1},2}(X) \|^2 \) for a given \( j \). We have that

\[
\| c^{j+1} - b_{r_{j+1},2}(X) \|^2 \leq (1 - \Theta^2) \| c^{j+1} \|^2
\]

\[
= (1 - \Theta^2) \| r^{j+1} + \frac{\theta_{j+1}}{\theta_j} (c^j - b_{r_j,2}(X)) \|^2
\]

\[
\leq (1 - \Theta^2)(1 + \rho) \| r^{j+1} \|^2 + (1 - \Theta^2)(1 + 1/\rho) \| \frac{\theta_{j+1}}{\theta_j} (c^j - b_{r_j,2}(X)) \|^2
\]

\[
\leq (1 + \rho)(1 - \Theta^2) \| r^{j+1} \|^2 + (1 - \Theta^2)(1 + 1/\rho) \| (c^j - b_{r_j,2}(X)) \|^2,
\]

where the first inequality follows from our assumption about the density of the weak-learner class \( \mathcal{B} \) (the same of the argument in (6.8)), the second inequality holds for any \( \rho \geq 0 \) due to Mean-Value inequality, and the last inequality is from \( \theta_{j+1} \leq \theta_j \).

We now derives a recursive bound on the left side of (6.14). From this, (6.14) follows from an elementary fact of recursive sequence as stated in Lemma 6.4.6 with \( a_j = \alpha_j^2 \| c^j - b_{r_j,2}(X) \|^2 \) and \( c_j = \alpha_j^2 \| r^j \|^2 \). \(\square\)

**Remark 6.4.2.** If \( c^m = b_{r_{m,2}}(X) \) (i.e. our class of learners \( \mathcal{B} \) is strong), then

\[
\delta_m = -(1 - s - t) \frac{\alpha_m^2}{2s^2} \| r^m \|^2 \leq 0.
\]

Lemma 6.4.6 is an elementary fact of recursive sequence used in the proof of Lemma 6.4.5.

**Lemma 6.4.6.** Given two sequences \( \{a_j \geq 0\} \) and \( \{c_j \geq 0\} \) such that the following holds for any \( \rho \geq 0 \),

\[
a_{j+1} \leq (1 - \Theta^2)[(1 + 1/\rho)a_j + (1 + \rho)c_{j+1}],
\]

180
then the sum of the terms \( a_j \) can be bounded as

\[
\sum_{j=0}^{m} a_j \leq \frac{t(1-s-t)}{s^2} \sum_{j=0}^{m} c_j .
\]

**Proof.** The recursive bound on \( a_j \) implies that

\[
a_j \leq (1 - \Theta^2)[(1 + 1/\rho)a_{j-1} + (1 + \rho)c_j]
\]

\[
\leq \sum_{k=0}^{j} [(1 + 1/\rho)(1 - \Theta^2)]^{j-k}(1 + \rho)(1 - \Theta^2)c_k .
\]

Summing both the terms gives

\[
\sum_{j=0}^{m} a_j \leq \sum_{j=0}^{m} \sum_{k=0}^{j} [(1 + 1/\rho)(1 - \Theta^2)]^{j-k}(1 + \rho)(1 - \Theta^2)c_k
\]

\[
= \sum_{k=0}^{m} \sum_{j=k}^{m} [(1 + 1/\rho)(1 - \Theta^2)]^{j-k}(1 + \rho)(1 - \Theta^2)c_k
\]

\[
\leq \sum_{k=0}^{m} \left( \sum_{j=0}^{\infty} [(1 + 1/\rho)(1 - \Theta^2)]^{j} \right)(1 + \rho)(1 - \Theta^2)c_k
\]

\[
= \frac{(1 + \rho)(1 - \Theta^2)}{1 - (1 + 1/\rho)(1 - \Theta^2)} \sum_{k=0}^{m} c_k
\]

\[
= \frac{(1 + \rho)(1 - \Theta^2)}{\Theta^2 - (1 - \Theta^2)/\rho} \sum_{k=0}^{m} c_k
\]

\[
= \frac{2(1 + \rho)(1 - \Theta^2)}{\Theta^2} \sum_{k=0}^{m} c_k
\]

\[
= \frac{2(2 - \Theta^2)(1 - \Theta^2)}{\Theta^4} \sum_{k=0}^{m} c_k
\]

where in the last two equalities we chose \( \rho = \frac{2(1-\Theta^2)}{\Theta^2} \). Now recall that \( s \leq \frac{\Theta^2}{4+\Theta^2} \in (0, 1) \).
and that $t = (1 - s)/2$:

$$
\sum_{j=0}^{m} a_j \leq \frac{2(2 - \Theta^2)(1 - \Theta^2)}{\Theta^4} \sum_{k=0}^{m} c_k \\
\leq \frac{4}{\Theta^4} \sum_{k=0}^{m} c_k \\
= \left( \frac{4 + \Theta^2}{\Theta^2} - 1 \right)^2 \frac{1}{4} \sum_{k=0}^{m} c_k \\
\leq \left( \frac{1}{s} - 1 \right)^2 \frac{1}{4} \sum_{k=0}^{m} c_k \\
= \frac{(1 - s)^2}{4s^2} \sum_{k=0}^{m} c_k \\
= \frac{t(1 - s - t)}{s^2} \sum_{k=0}^{m} c_k .
$$

\hfill \Box

Lemma 6.4.4 and Lemma 6.4.5 directly result in our major theorem:

**Proof of Theorem 4.1** It follows from Lemma 6.4.4 and Lemma 6.4.5 that

$$
V^M(f^*) \leq V^{M-1}(f^*) + \delta_m \leq V^0(f^*) + \sum_{j=0}^{M-1} \delta_j \leq \frac{1}{2} \|f^0(X) - f^*(X)\|^2.
$$

Notice $V^M(f^*) \geq \frac{\alpha_m}{\theta_{m-1}} (L(f^M) - L(f^*))$ as the term $\frac{1}{2} \|f^M(X) - f^*(X)\|^2 \geq 0$, which induces that

$$
L(f^M) - L(f^*) \leq \frac{\theta_{M-1}}{2\sigma_{M-1}} \|f^0(X) - f^*(X)\|^2 = \frac{1}{2\eta} \cdot \frac{\|f^0(X) - f^*(X)\|^2}{M^2}.
$$

\hfill \Box
6.5 Extensions and Variants

In this section we study two more practical variants of AGBM. First we see how to restart the algorithm to take advantage of strong convexity of the loss function. Then we will study a straight-forward approach to accelerated GBM, which we call vanilla accelerated gradient boosting machine (VAGBM), a variant of the recently proposed algorithm in [121], however without any theoretical guarantees.

6.5.1 Restart and Linear Convergence

It is more common to show a linear rate of convergence for GBM methods by additionally assuming that the function $l(x)$ is $\mu$-strongly convex (e.g. [66]). It is then relatively straight-forward to recover an accelerated linear rate of convergence by restarting Algorithm 14.

**Algorithm 15** Accelerated Gradient Boosting Machine with Restart (AGBMR)

**Input:** Starting function $\tilde{f}^0(x)$, step-size $\eta$, momentum-parameter $\gamma \in (0,1]$, strong-convexity parameter $\mu$.

For $p = 0, \ldots, P - 1$ do:

1. Run AGBM (Algorithm 14) initialized with $f^0(x) = \tilde{f}^p(x)$:
   - **Option 1:** for $M = \sqrt{\frac{2}{\eta \mu}}$ iterations.
   - **Option 2:** until $L(f^m) > L(f^{m-1})$.
2. Set $f^{p+1}(x) = f^M(x)$.

**Output:** $\tilde{f}^P(x)$.

**Theorem 6.5.1.** Consider Accelerated Gradient Boosting with Restarts with Option 1 (Algorithm 15). Suppose that $l(x)$ is $\sigma$-smooth and $\mu$-strongly convex. If the stepsize $\eta \leq \frac{1}{\sigma}$ and the momentum parameter $\gamma \leq \Theta^4/(4 + \Theta^2)$, then for any $p$ and optimal loss $L(f^*)$,

$$L(\tilde{f}^{p+1}) - L^* \leq \frac{1}{2}(L(\tilde{f}^p) - L(f^*)) .$$

**Proof.** The loss function $l(x)$ is $\mu$-strongly convex, which implies that

$$\frac{\mu}{2} \|f(X) - f^*(X)\|_2^2 \leq L(f) - L(f^*) .$$
Substituting this in Theorem 6.4.1 gives us that
\[ L(f^M) - L(f^*) \leq \frac{1}{\mu \eta \gamma (M + 1)^2} (L(f^0) - L(f^*)) . \]

Recalling that \( f^0(x) = \hat{f}^p(x) \), \( f^M(x) = \hat{f}^{p+1}(x) \), and \( M^2 = 2/\eta \mu \gamma \) gives us the required statement. \( \square \)

The restart strategy in Option 1 requires knowledge of the strong-convexity constant \( \mu \). Alternatively, one can also use adaptive restart strategy (Option 2) which is known to have good empirical performance [83].

**Remark 6.5.1.** Theorem 6.5.1 shows that \( M = O \left( \frac{1}{\sigma^2} \sqrt{\frac{C}{\mu} \log(1/\epsilon)} \right) \) weak learners are sufficient to obtain an error of \( \epsilon \) using ABGMR (Algorithm 15). In contrast, standard GBM (Algorithm 13) requires \( M = O \left( \frac{1}{\sigma^2} \log(1/\epsilon) \right) \) weak learners. Thus AGBMR is significantly better than GBM only if the condition number is large i.e. \((\sigma/\mu \geq 1)\). When \( l(y, f) \) is the least-squares loss, \((\mu = \sigma = 1)\) we would see no advantage of acceleration. However for more complicated functions with \((\sigma \gg \mu)\) (e.g. logistic loss or exp loss), AGBMR might result in an ensemble that is significantly better (e.g. obtaining lower training loss) than that of GBM for the same number of weak learners.

### 6.5.2 A Vanilla Accelerated Gradient Boosting Method

A natural question to ask is whether, instead of adding two learners at each iteration, we can get away with adding only one? Below we show how such an algorithm would look like and argue that it may not always converge.

Following the updates in Equation (6.5), we can get a direct acceleration of GBM by using the weak learner fitting the gradient. This leads to an Algorithm 16.

Algorithm 16 is equivalent to the recently developed accelerated gradient boosting machines algorithm [12, 33]. Unfortunately, it **may not always converge** to an optimum or may even **diverge**. This is because \( b_{rm} \) from Step (2) is only an
Algorithm 16 Vanilla Accelerated Gradient Boosting Machine (VAGBM)

**Input.** Starting function \( f^0(x) = 0 \), step-size \( \eta \), momentum parameter \( \gamma \in (0, 1] \).

**Initialization.** \( h^0(x) = f^0(x) \), and sequence \( \theta_m = \frac{2}{m+2} \).

For \( m = 0, \ldots, M-1 \) do:

**Perform Updates:**
1. Compute a linear combination of \( f \) and \( h \): \( g^m(x) = (1 - \theta_m) f^m(x) + \theta_m h^m(x) \).
2. Compute pseudo residual: \( r^m = -\frac{\partial f(y_i, g^m(x_i))}{\partial g^m(x_i)} \).
3. Find the best weak-learner for pseudo residual: \( T_m = \arg \min_{T \in \mathcal{T}} \sum_{i=1}^n (r_i^m - b_T(x_i))^2 \).
4. Update the model: \( f^{m+1}(x) = g^m(x) + \eta b_{T_m}(x) \).
5. Update the momentum model: \( h^{m+1}(x) = h^m(x) + \frac{\eta}{\theta_m} b_{T_m}(x) \).

**Output.** \( f^M(x) \).

approximate-fit to \( r^m \), meaning that we only take an approximate gradient descent step. While this is not an issue in the non-accelerated version, in Step (2) of Algorithm 16, the momentum term pushes the \( h \) sequence to take a large step along the approximate gradient direction. This exacerbates the effect of the approximate direction and can lead to an additive accumulation of error as shown in [30]. In Section 6.6.1, we see that this is not just a theoretical concern, but that Algorithm 16 also diverges in practice in some situations.

**Remark 6.5.2.** Our corrected residual \( c^m \) in Algorithm 14 was crucial to the theoretical proof of converge in Theorem 6.4.1. One extension could be to introduce \( \gamma \in (0, 1) \) in step (5) of Algorithm 16 just as in Algorithm 14.

### 6.6 Numerical Experiments

In this section, we present the results of computational experiments and discuss the performance of AGBM with trees as weak-learners. Subsection 6.6.1 demonstrates that the algorithm described in Section 6.5.2 may diverge numerically; Subsection 6.6.2 shows training and testing performance for GBM and AGBM with different parameters; and Subsection 6.6.3 compares the performance of GBM and AGBM with best tuned parameters. The code for the numerical experiments will be also...
Datasets: Table 6.2 summaries the basic statistics of the LIBSVM datasets that were used. For each dataset, we randomly choose 80% as the training and the remaining as the testing dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>task</th>
<th># samples</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>ala</td>
<td>classification</td>
<td>1605</td>
<td>123</td>
</tr>
<tr>
<td>wla</td>
<td>classification</td>
<td>2477</td>
<td>300</td>
</tr>
<tr>
<td>diabetes</td>
<td>classification</td>
<td>768</td>
<td>8</td>
</tr>
<tr>
<td>german</td>
<td>classification</td>
<td>1000</td>
<td>24</td>
</tr>
<tr>
<td>housing</td>
<td>regression</td>
<td>506</td>
<td>13</td>
</tr>
<tr>
<td>eunite2001</td>
<td>regression</td>
<td>336</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 6.2: Basic statistics of the (real) datasets used.

AGBM with CART trees: In our experiments, all algorithms use CART trees as the weak learners. For classification problems, we use logistic loss function, and for regression problems, we use least squares loss. To reduce the computational cost, for each split and each feature, we consider 100 quantiles (instead of potentially all \( n \) values). These strategies are commonly used in implementations of GBM like [27, 88].

6.6.1 Evidence that VAGBM May Diverge

Figure 6-1 shows the training loss versus the number of trees for the housing dataset with step-size \( \eta = 1 \) and \( \eta = 0.3 \) for VAGBM and for AGBM with different parameters \( \gamma \). The \( x \)-axis is number of trees added to the ensemble (recall that our AGBM algorithm adds two trees to the ensemble per iteration, so the number of boosting iterations of VAGBM and AGBM is different). As we can see, when \( \eta \) is large, the training loss for VAGBM diverges very fast while our AGBM with proper parameter \( \gamma \) converges. When \( \eta \) gets smaller, the training loss for VAGBM may decay faster than our AGBM at the begining, but it gets stuck and never converges to the true optimal
solution. Eventually the training loss of VAGBM may even diverge. On the other hand, our theory guarantees that AGBM always converges to the optimal solution.

\[ \eta = 1 \quad \eta = 0.3 \]

Figure 6-1: Training loss versus number of trees for VAGBM (which doesn’t converge) and AGBM with different parameters $\gamma$.

### 6.6.2 AGBM Sensitivity to the hyperparameters

In this section we look at how the two parameters $\eta$ and $\gamma$ affect the performance of AGBM. Figure 6-2 shows the training loss and the testing loss versus the number of trees for the a1a dataset with two different step-sizes $\eta = 4$ and $\eta = 0.1$ (recall AGBM adds two trees per iteration). When the step-size $\eta$ is large (with logistic loss, the largest step-size to guarantee the convergence is $\eta = 1/\sigma = 4$), the training loss decays very fast, and the traditional GBM can converge even faster than our AGBM at the beginning. But the testing performance is suffering, demonstrating that such a fast (due to the learning rate) convergence can result in severe overfitting. In this case, our AGBM with proper parameter $\gamma$ has a slightly better testing performance. When the step-size becomes smaller, the testing performance of all algorithms becomes more stable, though the training becomes slower. AGBM with proper $\gamma$ may require less number of iterations/trees to get a good training/testing performance.
6.6.3 Experiments with Fine Tuning

In this section we look at the testing performance of GBM, VAGBM and AGBM on six datasets with hyperparameter tuning. We consider depth 5 trees as weak-learners. We early stop the splitting when the gain smaller than 0.001 (roughly $1/n$ for these datasets). The hyper-parameters and their ranges we tuned are:

- step size ($\eta$): 0.01, 0.03, 0.1, 0.3, 1 for least squares loss and 0.04, 0.12, 0.4, 1.2, 4 for logistic loss;
- number of trees: 10, 11, $\ldots$, 100;
- momentum parameter $\gamma$ (only for AGBM): 0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.5, 1.

For each dataset, we randomly choose 80% as the training dataset and the remainder was used as the final testing dataset. We use 5-fold cross validation on the training
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Training</th>
<th>Testing</th>
<th># iter</th>
<th># trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GBM</td>
<td>0.2187</td>
<td>0.3786</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>a1a</td>
<td>VAGBM</td>
<td>0.2454</td>
<td>0.3661</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>AGBM</td>
<td>0.1994</td>
<td>0.3730</td>
<td>33</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>GBM</td>
<td>0.0262</td>
<td>0.0578</td>
<td>84</td>
<td>84</td>
</tr>
<tr>
<td>w1a</td>
<td>VAGBM</td>
<td>0.0409</td>
<td>0.0578</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>AGBM</td>
<td>0.0339</td>
<td>0.0552</td>
<td>47</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>GBM</td>
<td>0.297</td>
<td>0.462</td>
<td>87</td>
<td>87</td>
</tr>
<tr>
<td>diabetes</td>
<td>VAGBM</td>
<td>0.271</td>
<td>0.462</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>AGBM</td>
<td>0.297</td>
<td>0.458</td>
<td>47</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>GBM</td>
<td>0.244</td>
<td>0.505</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>german</td>
<td>VAGBM</td>
<td>0.288</td>
<td>0.514</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>AGBM</td>
<td>0.305</td>
<td>0.485</td>
<td>35</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>GBM</td>
<td>0.2152</td>
<td>4.6603</td>
<td>93</td>
<td>93</td>
</tr>
<tr>
<td>housing</td>
<td>VAGBM</td>
<td>0.5676</td>
<td>5.8090</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>AGBM</td>
<td>0.215</td>
<td>4.5074</td>
<td>35</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>GBM</td>
<td>36.73</td>
<td>270.1</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>cunite2001</td>
<td>VAGBM</td>
<td>28.99</td>
<td>245.2</td>
<td>58</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>AGBM</td>
<td>26.74</td>
<td>245.4</td>
<td>24</td>
<td>48</td>
</tr>
</tbody>
</table>

Table 6.3: Performance of after tuning hyper-parameters.

dataset to tune the hyperparameters. Instead of going through all possible hyperparameters, we utilize randomized search (*RandomizedSearchCV* in scikit-learn). As AGBM has more parameters (namely $\gamma$), we did proportionally more iterations of random search for AGBM. Table 6.3 presents the performance of GBM, VAGBM and AGBM with the tuned parameters. As we can see, the accelerated methods (AGBM and VAGBM) in general require less numbers of iterations to get similar or slightly better testing performance than GBM. Compared with VAGBM, AGBM adds two trees per iteration, and that can be more expensive, but the performance of AGBM can be more stable, for example, the testing error of VAGBM for housing dataset is much larger than AGBM.

### 6.7 Additional Discussions

Below we include some additional discussions which help to understand the relevance of our results when applied to frameworks typically used in practice.
6.7.1 Line search in Boosting

Traditionally the analysis of gradient boosting methods has focused on algorithms which use line search to select the step-size $\eta$ (e.g. Algorithm 13). Analysis of gradient descent suggests that is not necessary—using a fixed step-size of $1/\beta$ where $l(x)$ is $\beta$-smooth is sufficient [66]. Our accelerated Algorithm 14 also adopts this fixed step-size strategy. In fact, even the standard boosting libraries (XGBoost and TFBT) typically use a fixed (but tuned) step-size and avoid an expensive line search.

6.7.2 Use of Hessian

Popular boosting libraries such as XGBoost [27] and TFBT [88] compute the Hessian and perform a Newton boosting step instead of gradient boosting. Since the Newton step may not be well defined (e.g. if the Hessian is degenerate), an additional euclidean regularizer is also added. This has been shown to improve performance and reduce the need for a line-search for the $\eta$ parameter sequence [101, 97]. For LogitBoost (i.e. when $l(x)$ is the logistic loss), [101] demonstrate that trust-region Newton’s method can indeed significantly improve the convergence. Leveraging similar results in second-order methods for convex optimization (e.g. [81, 51]) and adapting accelerated second-order methods [74] would be an interesting direction for the future work.

6.7.3 Out-of-sample Performance

Throughout this work we focus only on minimizing the empirical training loss $L(f)$ (see Formula (6.2)). In reality what we really care about is the out-of-sample error of our resulting ensemble $f^M(x)$. A number of regularization tricks such as i) early stopping [116], ii) pruning [27, 88], iii) smaller step-sizes [88], iv) dropout [88] etc. are usually employed in practice to prevent over-fitting and improve generalization. Since AGBM requires much fewer iterations to achieve the same training loss than GBM, it outputs a much sparser set of learners. We believe this is partially the reason for its better out-of-sample performance. However a joint theoretical study of the out-of-
sample error along with the empirical error $L_n(f)$ is much needed. It would also shed light on the effectiveness of the numerous ad-hoc regularization techniques currently employed.

6.8 Conclusion

In this chapter, we proposed a novel Accelerated Gradient Boosting Machine (AGBM), proved its rate of convergence and introduced a computationally inexpensive practical variant of AGBM that takes advantage of strong convexity of loss function and achieves linear convergence. Finally we demonstrated with a number of numerical experiments the effectiveness of AGBM over conventional GBM in obtaining a model with good training and/or testing data fidelity.
Appendix A

Growth Conditions

A.1 Growth Constant $G$ and the Modulus of Weak Sharp Minima

The optimal solution set $\text{Opt}$ of (4.1) is called a set of weak sharp minima with modulus $\alpha$ if it holds that:

$$f(x) \geq f^* + \alpha \cdot \text{Dist}(x, \text{Opt}) \quad \text{for all } x \in Q.$$  \hfill (A.1)

This concept was first developed by Polyak [85] when $\text{Opt}$ is a singleton, and generalized by Burke and Ferris [23] to include the possibility of multiple optima. The modulus of weak sharp minima has been a useful tool in sensitivity analysis [25, 50], convergence analysis for certain problem classes [24, 23], linear regularity and error bounds [20, 21, 22], perturbation properties of nonlinear optimization [95, 96, 15], as well as in the finite termination of certain algorithms [86], [32], and [24].

Comparing (A.1) to (2.5), we see that the modulus $\alpha$ of weak sharp minima is a close cousin of the growth constant $G$. Indeed, if we were to loosen the restriction that $f_{\text{slb}}$ be a strict lower bound and instead allow it to take the value $f_{\text{slb}} = f^*$ in the definition of $G$ in (2.3), then we would obtain precisely that $G = \alpha^{-1}$. However,
the notion of $f_{slb}$ being a strict lower bound is fundamental for the results herein.

Note that (A.1) specifies the exact local growth of $f(\cdot)$ away from the set of optimal solutions. And although as defined in (A.1) the weak sharp minima is a global property, due to convexity it is essentially a local property and indeed its usefulness derives from the local nature of the weak sharp minima in a neighborhood of the optimal solution set. This is in contrast to the growth constant $G$ as defined in (2.3), which by its nature is a global property as illustrated in the constructions in Figure 2-1. Last of all we point out that while one can easily have $\alpha = 0$ for weak sharp minima (just let $f(x)$ be a differentiable convex function whose optimum is attained in the relative interior of $Q$), Theorem 2.1.1 shows that $G$ is finite for all reasonably-behaved convex functions.

A.2 Proof of Theorem 2.1.1

Proof of Theorem 2.1.1: Let us fix an optimal solution $x^* \in \text{Opt}$, and define $\delta := \max_{v \in E_{E}} \|v - x^*\|$ and define $\bar{G} := \max\{\frac{\delta}{\epsilon}, \frac{\delta}{f^* - f_{slb}}\}$. We will prove that for any $x \in Q$, the following inequality holds:

$$\text{Dist}(x, \text{Opt}) \leq \bar{G} (f(x) - f_{slb}) \ ,$$

(A.2)

which then implies that $G \leq \bar{G}$ is finite. We consider two cases as follows:

Case (i): $x \in \text{Opt}_{\epsilon}$. In this case we have $x = v + s$ where $v \in E_{E}$ and $s \in S$. Since $s$ is in the recession cone of $\text{Opt}_{\epsilon}$ it holds that $x^* + s \in \text{Opt}$, whereby

$$\text{Dist}(x, \text{Opt}) \leq \|x - (x^* + s)\| = \|v - x^*\| \leq \delta \ ,$$

(A.3)

and therefore

$$f(x) - f_{slb} \geq f^* - f_{slb} \geq \frac{(f^* - f_{slb})(\text{Dist}(x, \text{Opt}))}{\delta} \geq \bar{G}^{-1}\text{Dist}(x, \text{Opt}) \ ,$$

194
which shows (A.2) in this case.

Case (ii): $x \notin \text{Opt}$. Let $x^1$ be the projection of $x$ onto Opt and let $x^2$ be the point on the line segment from $x^1$ to $x$ that satisfies $f(x^2) = f^* + \varepsilon$. (Existence of $x^2$ is guaranteed by continuity of $f(\cdot)$.) Then

$$f(x) - f_{\text{slb}} \geq f(x) - f^* \geq (f(x^2) - f^*) \frac{\|x - x^1\|}{\|x^2 - x^1\|} \geq \frac{\varepsilon\|x - x^1\|}{\delta} = \frac{\varepsilon \text{Dist}(x, \text{Opt})}{\delta} \geq \tilde{G}^{-1} \text{Dist}(x, \text{Opt}),$$

where the second inequality is from the convexity of $f(\cdot)$ which implies the chordal inequality $\frac{f(x) - f^*}{\|x - x^1\|} \geq \frac{f(x^2) - f^*}{\|x^2 - x^1\|}$, and the third inequality uses $\|x^2 - x^1\| = \text{Dist}(x^2, \text{Opt}) \leq \delta$ (from (A.3)). The last equality above uses the fact that $\text{Dist}(x, \text{Opt}) = \|x - x^1\|$. This proves (A.2) in this case. \qed
Appendix B

Relative Conditions

B.1 Solving the subproblem (4.7) when \( h(x) \) is a convex function of \( \|x\|_2^2 \) and \( Q \) has simple constraints

We consider the following subproblem:

\[
\min_{x \in Q} \langle c, x \rangle + h(x), \tag{B.1}
\]

where \( h(x) = g(\|x\|_2^2) \) and \( g(\cdot) \) is a (univariate) closed convex function of \( \|x\|_2^2 \). Let \( y := \|x\|_2^2 \) and define \( D := \{\|x\|_2^2 : x \in Q\} \subset \mathbb{R} \), which is the domain of \( g(\cdot) \). Let \( g^*(\cdot) \) denote the conjugate function of \( g(\cdot) \), namely

\[
g^*(t) := \sup_{y \in D} \{ty - g(y)\},
\]

whose domain we denote by \( D^* \). Since \( g(\cdot) \) is a convex function, we know from conjugacy theory [7] that \( g(y) = \sup_{t \in D^*} \{ty - g^*(t)\} \). Therefore (B.1) becomes

\[
\min_{x \in Q} \{\langle c, x \rangle + g(\|x\|_2^2)\} = \min_{x \in Q} \{\sup_{t \in D^*} \{\langle c, x \rangle + t\|x\|_2^2 - g^*(t)\}\}
\]

\[
= \sup_{t \in D^*} \{-g^*(t) + \min_{x \in Q} \{\langle c, x \rangle + t\|x\|_2^2\}\},
\]
where the second equality above holds whenever the min and the sup operators can
be exchanged (which is akin to strong duality). Notice that \( \min_{x \in Q} \{ \langle c, x \rangle + t \| x \|_2^2 \} \) is a Euclidean projection problem. Therefore the subproblem (4.7) becomes a 1-
dimensional concave maximization problem if the Euclidean projection problem can
be easily solved and one can conveniently form and work with the univariate convex
conjugate function \( g^*(\cdot) \).

### B.2 Extension to Composite Optimization

Here we discuss some details of the extension of the ideas and results of this paper
to composite optimization as described in Section 3.3.3, using the definitions \( \tilde{f}(\cdot) := f(\cdot) + P(\cdot) \), and \( \tilde{h}(\cdot) = Lh(\cdot) + P(\cdot) \) as defined in Section 3.3.3. Note that \( \tilde{f}(\cdot) \) and \( \tilde{h}(\cdot) \) are not necessarily differentiable on \( Q \) since they include the function \( P(\cdot) \). However, we can use the equivalent condition from (a-ii) of Proposition 3.1.1 to define relative
smoothness. Let us now show how convergence results for the Primal Gradient Scheme
still hold in this more general setting using an extension of the proof of Theorem 3.3.1.

Let \( g_P(x) \in \partial P(x) \) be a specific subgradient of \( P(\cdot) \) at \( x \), and we will use the same
subgradient of \( P(\cdot) \) at \( x \) when constructing a subgradient of \( \tilde{f}(\cdot) \) and/or \( \tilde{h}(\cdot) \), namely
\( g_f(x) := \nabla f(x) + g_P(x) \) and \( g_h(x) := L \nabla h(x) + g_P(x) \). Then Algorithm 6 has the
following update:

\[
\begin{align*}
    x^{i+1} &= \arg\min_{x \in Q} \{ \tilde{f}(x^i) + \langle g_f(x^i), x - x^i \rangle + D_{\tilde{h}}(x, x^i) \} \\
    &= \arg\min_{x \in Q} \{ \tilde{f}(x^i) + \langle \nabla f(x^i) + g_P(x^i), x - x^i \rangle + D_{Lh}(x, x^i) + P(x) - P(x^i) - \langle g_P(x^i), x - x^i \rangle \} \\
    &= \arg\min_{x \in Q} \{ f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + LD_h(x, x^i) + P(x) \},
\end{align*}
\]

(B.2)

where in the third equality above the term involving \( g_P(x^i) \) arising in \( \partial \tilde{f}(x^i) \) cancels
out the corresponding term involving \( g_P(x^i) \) arising in \( \partial \tilde{h}(x^i) \) as part of the expansion
of \( D_{\tilde{h}}(x, x^i) \). There is therefore no actual need to compute \( g_P(x^i) \in \partial P(x^i) \) in the
update. Indeed, this update (B.2) corresponds exactly to the update in the NoLips
algorithm [8] (up to the step-size) and the PGA-B algorithm in [118] (up to the

198
step-size) for composite optimization.

The proof of the computational guarantee in Theorem 3.3.1 can be generalized directly to the composite optimization setting as follows. Let us denote

\[ s_t(x) := \tilde{f}(x^i) + \langle g_f(x^i), x - x^i \rangle + D_h(x, x^i) = f(x^i) + \langle \nabla f(x^i), x - x^i \rangle + LD_h(x, x^i) + P(x). \]

Notice that \( x^{i+1} = \arg \min_{x \in Q} s_t(x) \); therefore from the first-order optimality conditions there is a subgradient \( g_{s_t}(x^{i+1}) \in \partial s_t(x^{i+1}) \) for which \( \langle g_{s_t}(x^{i+1}), x - x^{i+1} \rangle \geq 0 \) for all \( x \in Q \). From the additivity property of subgradients, we can write \( g_{s_t}(x^{i+1}) = \nabla f(x^i) + L \nabla h(x^{i+1}) - L \nabla h(x^i) + \bar{g} \) for some \( \bar{g} \in \partial P(x^{i+1}) \), and let us assign \( g_P(x^{i+1}) := \bar{g} = g_{s_t}(x^{i+1}) - \nabla f(x^i) - L \nabla h(x^{i+1}) + L \nabla h(x^i) \), which then is used to define the subgradient \( g_f(x^{i+1}), g_h(x^{i+1}) \), and the Bregman distance \( D_h(x, x^{i+1}) \) in the proof.

Recall that the Primal Gradient Scheme does not rely on the choice of subgradient of \( P(x^{i+1}) \), thus the choice of \( g_P(x^{i+1}) \) is only used in the proof and it is well-defined.

Utilizing the above method for specifying the subgradients of \( P(\cdot) \) at each of the iterates \( x^i \) of the Primal Gradient Scheme, we can prove the following more specialized form of the Three Point Property which we can use in the proof of Theorem 3.3.1 for the setting composite optimization.

**Lemma B.2.1.** For any \( x \in Q \), we have for any \( i \geq 0 \),

\[
\begin{align*}
    f(x^i) + \langle g_f(x^i), x^{i+1} - x^i \rangle + D_h(x^{i+1}, x^i) &\leq f(x^i) + \langle g_f(x^i), x - x^i \rangle + D_h(x, x^i) - D_h(x, x^{i+1}). \\
\end{align*}
\]

(B.3)

**Proof:** Notice that \( s_t(x) - \bar{h}(x) = f(x^i) + \langle \nabla f(x^i) - L \nabla h(x^i), x - x^i \rangle - Lh(x^i) \)
and so is a linear function of $x$, whereby it holds that

$$ (s_i(x) - \bar{h}(x)) - (s_i(x_{i+1}) - \bar{h}(x_{i+1})) = \langle \nabla (s_i - \bar{h})(x_{i+1}), x - x_{i+1} \rangle $$

$$ = \langle g_{s_i}(x_{i+1}), x - x_{i+1} \rangle - \langle g_{\bar{h}}(x_{i+1}), x - x_{i+1} \rangle $$

$$ \geq -\langle g_{\bar{h}}(x_{i+1}), x - x_{i+1} \rangle $$

where the inequality follows from the choice of $g_{s_i}(x_{i+1})$. Rearranging the above and recalling the definition of $s_i(x)$ then completes the proof.

The proof of Theorem 3.3.1 in the setting of composite optimization follows directly by replacing $h(\cdot)$, $\nabla h(\cdot)$, $f(\cdot)$ and $\nabla f(\cdot)$ by $\bar{h}(\cdot)$, $g_{\bar{h}}(\cdot)$, $\bar{f}(\cdot)$ and $g_{\bar{f}}(\cdot)$, respectively, and utilizing (B.3) to deduce the second inequality in (3.28).

### B.3 Criteria for choosing the reference function $h(\cdot)$

One natural question is how can we choose $h(\cdot)$ in order to lower the value of the bound in Theorem 3.3.1? Let us consider the simple case when $f(\cdot)$ is twice differentiable and is not strongly convex, namely $\mu = 0$, and $f(\cdot)$ attains its optimum at some point $x^*$. Then the convergence bound (4.4.3) can be re-written as:

$$ f(x^k) - f(x^*) \leq \frac{1}{k} D_{Lh}(x^*, x^0) $$

$$ = \frac{1}{k} D_f(x^*, x^0) + \frac{1}{k} \left( \int_0^1 \int_0^t (x^* - x^0)^T [\nabla^2 (Lh - f)(x^0 + s(x^* - x^0))] (x^* - x^0) \, ds \, dt \right) $$

where $\nabla^2 (Lh - f)(y)$ is the Hessian of the "gap function" $Lh(\cdot) - f(\cdot)$ at the point $y \in Q$. Notice that the first term above is fixed independent of the choice of $h(\cdot)$ and $L$. It follows from Proposition 3.1.1 that if $f(\cdot)$ is $L$-smooth relative to $h(\cdot)$ then $\nabla^2 (Lh - f)(y) \succeq 0$ for any $y \in \text{int } Q$, whereby the second term above is always nonnegative. Since we do not know $x^*$ in most cases, in order to make the bound smaller we want the Hessian $\nabla^2 (Lh - f)(y)$ to be smaller for all $y \in \text{int } Q$. 

200
There is a trade-off between how small the Hessian $\nabla^2 (Lh - f)(y)$ is and how hard it will be to solve the subproblem (4.7). If we choose $Lh(\cdot) = f(\cdot)$, the Hessian of the gap function is 0, but solving the subproblem (4.7) is as hard as solving the original problem (4.1). On the other hand, in standard gradient descent we use $h(\cdot) = \frac{1}{2} \| \cdot \|_2$ in which case the subproblem (4.7) can be easily solved, while the Hessian of the gap function can be huge—thus implying a poorer convergence bound. There are a number of ways to try to manage this trade-off. For example, in gradient descent with preconditioning we can use $h(\cdot) = \frac{1}{2} \| \cdot \|_B$ := $\sqrt{\langle \cdot, B \cdot \rangle}$, where $B$ is a computationally-friendly positive definite matrix—typically a diagonal matrix. The criteria for designing $B$ usually involves (i) ensuring that solving equations with $B$ is easy (so that the subproblem (4.7) can be easily solved), and (ii) $B$ is “close to” the Hessian of $f(\cdot)$ (so that the Hessian of the gap function is small).

### B.4 Finite Radius Bound for SVM

Here we derive an upper bound on the norm of an optimal solution of the SVM problem (4.26).

**Proposition B.4.1.** The optimal solution to the SVM problem (4.26) lies in the ball $B_2(0, R)$ for $R = \min \left\{ \frac{1}{\alpha \lambda} \sum_{i=1}^{n} \|w_i\|_2, \sqrt{2/\lambda} \right\}$.

**Proof:** For convenience define $A_i := y_i w_i$ for $i = 1, \ldots, n$. Then we can re-write the SVM problem as the following constrained optimization problem:

$$\min_{s, x} \quad \frac{1}{n} e^T s + \frac{1}{2} \|x\|_2^2$$

subject to

$$s + A x \geq e$$

and

$$s \geq 0.$$  

Let $\pi$ and $\beta$ be the multipliers on the inequality constraints above. Then the KKT
conditions imply, among other things, that the optimal solution $x^*$ must satisfy:

$$\pi^* + \beta^* = \frac{1}{n} e$$

$$\lambda x^* = A^T \pi^*$$

where $\pi^* \geq 0$ and $\beta^* \geq 0$. Define $\bar{\pi}^* = n \pi^*$. Then $0 \leq \bar{\pi}^* \leq e$ and

$$\lambda \|x^*\|_2 = \|A^T \pi^*\|_2 = \frac{1}{n} \|A^T \pi^*\|_2 \leq \frac{1}{n} \sum_{i=1}^{n} \|A_i\|_2 = \frac{1}{n} \sum_{i=1}^{n} \|w_i\|_2,$$

which proves the first term in the definition of $R$. Also, we have $\frac{1}{2} \|x^*\|_2^2 \leq f(x^*) \leq f(0) = 1$, thus $\|x^*\|_2 \leq \sqrt{2/\lambda}$. Therefore $\|x^*\|_2 \leq \min \left\{ \frac{1}{\sqrt{n} \lambda} \sum_{i=1}^{n} \|w_i\|_2, \sqrt{2/\lambda} \right\}$, which finishes the proof.
Bibliography


algorithms as gradient descent. In Advances in neural information processing 

[68] Indraneel Mukherjee, Cynthia Rudin, and Robert E Schapire. The rate of con-
2347, 2013.

[69] A. Nedić and S. Lee. On stochastic subgradient mirror-descent algorithm with 

[70] Sahand Negahban and Martin J Wainwright. Joint support recovery under high-
dimensional scaling: Benefits and perils of $\ell_{1,\infty}$-regularization. In Proceed-
ings of the 21st International Conference on Neural Information Processing Systems, 

[71] Arkadi S Nemirovsky and David B Yudin. Problem Complexity and Method 


[74] Yu Nesterov. Accelerating the cubic regularization of newtonâ€™s method on 

[75] Yurii Nesterov. A method of solving a convex programming problem with 
convergence rate $O(1/k^2)$. In Soviet Mathematics Doklady, volume 27, pages 

[76] Yurii Nesterov. Introductory lectures on convex optimization: a basic course. 

[77] Yurii Nesterov. Introductory lectures on convex optimization: A basic course, 


[79] Yurii Nesterov. Smoothing technique and its applications in semidefinite opti-

[80] Yurii Nesterov. Gradient methods for minimizing composite functions. Mathe-


