Methods for Convex Optimization and Statistical Learning

by

Paul Grigas

B.S., Cornell University (2011)

Submitted to the Sloan School of Management in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

© Massachusetts Institute of Technology 2016. All rights reserved.
Abstract

We present several contributions at the interface of first-order methods for convex optimization and problems in statistical machine learning. In the first part of this thesis, we present new results for the Frank-Wolfe method, with a particular focus on: (i) novel computational guarantees that apply for any step-size sequence, (ii) a novel adjustment to the basic algorithm to better account for warm-start information, and (iii) extensions of the computational guarantees that hold in the presence of approximate subproblem and/or gradient computations. In the second part of the thesis, we present a unifying framework for interpreting “greedy” first-order methods — namely Frank-Wolfe and greedy coordinate descent — as instantiations of the dual averaging method of Nesterov, and we discuss the implications thereof.

In the third part of the thesis, we present an extension of the Frank-Wolfe method that is designed to induce near-optimal low-rank solutions for nuclear norm regularized matrix completion and, for more general problems, induces near-optimal “well-structured” solutions. We establish computational guarantees that trade off efficiency in computing near-optimal solutions with upper bounds on the rank of iterates. We then present extensive computational results that show significant computational advantages over existing related approaches, in terms of delivering low rank and low run-time to compute a target optimality gap.

In the fourth part of the thesis, we analyze boosting algorithms in linear regression from the perspective of modern first-order methods in convex optimization. We show that classic boosting algorithms in linear regression can be viewed as subgradient descent to minimize the maximum absolute correlation between features and residuals. We also propose a slightly modified boosting algorithm that yields an algorithm for the Lasso, and that computes the Lasso path. Our perspective leads to first-ever comprehensive computational guarantees for all of these boosting algorithms, which provide a precise theoretical description of the amount of data-fidelity and regularization imparted by running a boosting algorithm, for any dataset. In the fifth and final part of the thesis, we present several related results in the
contexts of boosting algorithms for logistic regression and the AdaBoost algorithm.

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

First and foremost, I would like to thank my thesis advisor, mentor, colleague, and friend, Rob Freund. Rob has been a constant and incredible source of support and guidance during my time at MIT, and I deeply appreciate the considerable amount of time and effort that he has put forth while working with me. Rob has always been available to bounce ideas back and forth with, and at the same time he has always encouraged me to pursue my own ideas and to gain new and valuable experiences. Rob’s excitement for research, his deep knowledge and intuition about optimization, and his scholarly values and integrity have had a profound and lasting impact on me. I am grateful to have had such an overwhelmingly positive experience while working with Rob, and as I launch my own academic career I am driven to be as encouraging and generous with my future students as Rob was with me.

Secondly, I would like to thank Rahul Mazumder, who I have had the pleasure of collaborating with for several years and who served as one of my committee members. I met Rahul during the second year of my PhD and he has played an integral role in the development of my interests in statistics and machine learning. It is simply a joy to work with Rahul; his passion for research is infectious and the unique perspective he provides is very valuable. I am very grateful for his guidance and sage advice along the way as well.

I would also like to thank several other people who have had a significant influence on the work appearing in this thesis as well as my overall research interests. Thanks to Stefanie Jegelka for serving on my thesis committee and for providing valuable feedback on the work appearing in this thesis. I have greatly enjoyed every conversation with Stefanie and I am grateful for her thoughts on this work. I am very fortunate for the opportunity to collaborate with Yurii Nesterov. In a sense, Yurii was the unofficial fourth member of my committee and I am grateful for his support and the impact that he has had on my research. Each visit with Yurii has felt like the research equivalent of a 100-yard dash, and I tend to leave with many questions but a strong motivation to seek answers. I had a very interesting and exciting experience interning at Yahoo! Labs, and a big thanks goes to Zheng Wen and KC
Lee for mentoring me while I was there and for being great collaborators.

There are many faculty members at MIT from whom I have learned a lot, both inside and outside the classroom. Thanks to Itai Ashlagi, Dimitris Bertsimas, Erik Brynjolfsson, Vivek Farias, David Gamarnik, Patrick Jaillet, Retsef Levi, Jim Orlin, Asu Ozdaglar, Özalp Özer, Georgia Perakis, Andreas Schulz, David Simchi-Levi, John Tsitsiklis, and Juan Pablo Vielma for many interesting lectures and conversations. Special thanks also to Dimitris and Patrick for their commitment to the students of the ORC in their roles as Co-directors. I would also like to thank several faculty members at Cornell – Huseyin Topaloglu, Robert Bland, David Williamson, and David Shmoys – whose encouragement and advice helped lead me to pursue a PhD. Thanks to Laura Rose, Andrew Carvalho, and David Merrill, the ORC/Sloan staff members who have helped me along the way. I am also fortunate to have been supported by an NSF Graduate Research Fellowship for three years.

I owe a tremendous amount of gratitude to my friends who have supported me and have been there to unwind (and sometimes commiserate) with me. I feel very privileged to have been part of a community as wonderful as the Operations Research Center. The level of camaraderie and collegiality among students is truly stellar. To my close friends from the ORC (past and present) – Ross Anderson, Ali Aouad, Chaithanya Bandi, Andre Calmon, Maxime Cohen, Iain Dunning, Adam Elmachtoub, David Fagnan, Vishal Gupta, Swati Gupta, Kris Johnson, Nathan Kallus, Angie King, Andrew Li, Will Ma, Velibor Misić, Zach Owen, Allison O’Hair, Yaron Shaposhnik, John Silberholz, Joel Tay, Leon Valdes, He Wang, and Nataly Youssef – and to the many others who I have shared good times with, thank you for making this journey so memorable and fun. Special thanks to my senior officemates – Adam and Maxime – for motivating me and showing me what the finish line looks like. In addition to being a great friend, Adam has also become a great collaborator. Outside of the ORC, I am fortunate to have good friends from various stages of my life. Thanks to Adam, Paul, Andy R., Emily, Sean, Colleen, Andy P., Dan, Erica, Kerry, and Peter for many years of friendship, and special thanks to Britta whose friendship and support has always provided so much for me.
Finally, I would like to thank my family for their endless love and support. My parents have both always encouraged me to pursue my dreams. My mother, Jessica, has provided infinite, unconditional love throughout my entire life. My father, Paul, has taught me many valuable lessons. My brother, Jon, is also a lifelong best friend. My grandparents, Terry and Nick, have always supported me too. My grandfather, Nick, in particular has always been a role model and instilled in me the value of academics. All of my achievements are due to the unconditional love and support of my family, and I dedicate this thesis to them.
Contents

1 Introduction 19

2 New Analysis and Results for the Frank-Wolfe Method 25

2.1 Introduction ............................................. 25

2.1.1 Notation ............................................. 28

2.2 The Frank-Wolfe Method ................................ 28

2.3 Computational Guarantees for Specific Step-size Sequences ............... 36

2.3.1 A Well-studied Step-size Sequence .................. 37

2.3.2 Simple Averaging .................................... 38

2.3.3 Constant Step-size .................................. 38

2.3.4 Extensions using Line-Searches ..................... 39

2.4 Computational Guarantees for a Warm Start ......................... 40

2.4.1 A Dynamic Version of the Warm-Start Step-size Strategy .......... 42

2.5 Analysis of the Frank-Wolfe Method with Inexact Gradient Computations and/or Subproblem Solutions .......................... 46

2.5.1 Frank-Wolfe Method with Inexact Linear Optimization Subproblem Solutions ........................................... 47
3 Dual Averaging as a Unifying Viewpoint of Greedy First-Order Methods

3.1 Introduction

3.2 Greedy Coordinate Descent and Dual Averaging

3.2.1 Greedy Coordinate Descent as Dual Averaging/Mirror Descent

3.3 Frank-Wolfe as a Special Case of Dual Averaging

3.3.1 A New Complexity Analysis for Dual Averaging

4 An Extended Frank-Wolfe Method with “In-Face” Directions, and its Application to Low-Rank Matrix Completion

4.1 Introduction

4.1.1 Organization/Results

4.1.2 Notation

4.2 Frank-Wolfe Method, Away Steps, and In-Face Steps

4.2.1 Away Steps

4.2.2 An “In-Face” Extended Frank-Wolfe Method

4.3 Solving Matrix Completion Problems using the In-Face Extended Frank-Wolfe Method

4.3.1 Characterization of faces of the nuclear norm ball

4.3.2 Linear optimization subproblem solution for regular Frank-Wolfe step

4.3.3 Strategies and computation of the in-face direction $D^k$

4.3.4 Computing the maximal step-size $\alpha_{\text{stop}}^k$ in Step (2.)
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.5 Initial values, step-sizes, and computational guarantees</td>
<td>106</td>
</tr>
<tr>
<td>4.3.6 Efficiently Updating the Thin SVD of $Z^k$</td>
<td>107</td>
</tr>
<tr>
<td>4.3.7 Rank accounting</td>
<td>109</td>
</tr>
<tr>
<td>4.4 Computational Experiments and Results</td>
<td>110</td>
</tr>
<tr>
<td>5 A New Perspective on Boosting in Linear Regression</td>
<td>119</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>119</td>
</tr>
<tr>
<td>5.1.1 Contributions</td>
<td>126</td>
</tr>
<tr>
<td>5.2 LS-Boost($\varepsilon$): Computational Guarantees and Statistical Implications</td>
<td>128</td>
</tr>
<tr>
<td>5.2.1 Computational Guarantees and Intuition</td>
<td>128</td>
</tr>
<tr>
<td>5.2.2 Statistical Insights from the Computational Guarantees</td>
<td>132</td>
</tr>
<tr>
<td>5.3 Boosting Algorithms as Subgradient Descent</td>
<td>136</td>
</tr>
<tr>
<td>5.3.1 Brief Review of Subgradient Descent</td>
<td>136</td>
</tr>
<tr>
<td>5.3.2 A Subgradient Descent Framework for Boosting</td>
<td>138</td>
</tr>
<tr>
<td>5.3.3 Deriving and Interpreting Computational Guarantees for FS$_\varepsilon$</td>
<td>141</td>
</tr>
<tr>
<td>5.4 Boosting and LASSO</td>
<td>144</td>
</tr>
<tr>
<td>5.4.1 R-FS$_{\varepsilon,\delta}$: Computational Guarantees and their Implications</td>
<td>147</td>
</tr>
<tr>
<td>5.5 Statistical properties of boosting algorithms: an empirical study</td>
<td>152</td>
</tr>
<tr>
<td>6 A New Perspective on Boosting in Logistic Regression and AdaBoost</td>
<td>153</td>
</tr>
<tr>
<td>6.1 Review of Mirror Descent for Nonsmooth Convex Optimization</td>
<td>153</td>
</tr>
<tr>
<td>6.2 Greedy Methods for Logistic Regression</td>
<td>159</td>
</tr>
<tr>
<td>6.2.1 The Boosting Setup</td>
<td>166</td>
</tr>
<tr>
<td>6.2.2 Regularized Logistic Loss Minimization and its Properties</td>
<td>168</td>
</tr>
</tbody>
</table>
6.3 AdaBoost and Computational Guarantees for the Margin and the Log-Exponential Loss ......................................................... 170

6.3.1 Regularized Exponential Loss Minimization and its Properties ........ 183

A Supplement to Chapter 2 .......................................................... 187

A.1 Frank-Wolfe Bounds in the Presence of Minmax Duality Structure .... 187

A.2 Smoothness and the Curvature Constant $C_{h,q}$ ............................. 188

B Supplement to Chapter 4 .......................................................... 189

B.1 Remainder of the Proof of Theorem 4.1 ...................................... 189

C Supplement to Chapter 5 .......................................................... 193

C.1 Additional Details for Section 5.2 ............................................ 193

C.1.1 Properties of Convex Quadratic Functions ............................. 193

C.1.2 Proof of Theorem 5.1 ......................................................... 195

C.2 Additional Details for Section 5.3 ............................................ 199

C.2.1 An Elementary Sequence Process Result, and a Proof of Proposition 5.1 199

C.2.2 Proof of Theorem 5.2 ......................................................... 201

C.3 Additional Details for Section 5.4 ............................................ 203

C.3.1 Duality Between Regularized Correlation Minimization and the LASSO 203

C.3.2 Proof of Proposition 5.3 ......................................................... 207

C.3.3 Proof of Theorem 5.3 ......................................................... 207

D Supplement to Chapter 6 .......................................................... 213

D.1 Useful Properties Regarding LogitBoost .................................... 213
List of Figures

3-1 Algorithmic equivalencies .................................................................................. 74

4-1 The facial structure of a polytope is not affine invariant ................................... 94

4-2 Plots of rank and relative optimality gap versus run time ................................. 114

5-1 Evolution of LS-BOOST(\(\varepsilon\)) and FS\(\varepsilon\) .................................................. 123

5-2 Coefficient profiles for different algorithms ..................................................... 125

5-3 Figure showing shrinkage/training error profiles for LS-BOOST(\(\varepsilon\)), FS\(\varepsilon\) and LASSO ................................................................. 133

5-4 Profiles of \(\ell_1\) shrinkage for LS-BOOST(\(\varepsilon\)) ............................................... 134

5-5 Profiles of \(\ell_1\) shrinkage for FS\(\varepsilon\) ................................................................. 142

5-6 Evolution of the R-FS\(\varepsilon,\delta\) algorithm .......................................................... 148

5-7 Coefficient profiles for R-FS\(\varepsilon,\delta\) ................................................................. 151
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Properties of first-order methods</td>
<td>55</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of different methods on small-scale examples</td>
<td>115</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison of different methods on medium-large scale examples</td>
<td>116</td>
</tr>
<tr>
<td>4.3</td>
<td>Comparison of different methods on the MovieLens10M dataset</td>
<td>116</td>
</tr>
<tr>
<td>4.4</td>
<td>Diagnostic table showing properties of different methods</td>
<td>118</td>
</tr>
<tr>
<td>6.1</td>
<td>Computational bound guarantees for LogitBoost</td>
<td>167</td>
</tr>
<tr>
<td>6.2</td>
<td>Computational bound guarantees for AdaBoost</td>
<td>176</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

During the last 15 years, there has been a substantial amount of development and an ever-increasing amount of confluence in two important areas: first-order methods in convex optimization, and methodology for a wide array of problems in statistical machine learning. At the same time, these developments have been further catalyzed by the virtual explosion in the amount of available data in a wide array of application domains, i.e., the “big data” era.

For many applications in statistical machine learning – such as high-dimensional regression, classification, and matrix completion – first-order methods are appealing for a number of reasons. First, first-order methods offer a very low iteration iteration cost that, in general, scales much better with the dimension of the problem than that of interior-point or other polynomial-time methods. Second, in the setting of statistical learning, where optimization problems depend on data that is noisy or otherwise limited, it is not necessary and often not sensible to require very high-accuracy solutions. Thus the weaker rates of convergence of first-order methods are typically satisfactory for such applications. Finally, the relatively “simple” nature of the computations involved in many first-order methods make them amenable to enhancements – such as parallelization or distributed architectures – that lead to truly scalable implementations.

In this thesis, we present several contributions at the interface of first-order methods for
convex optimization and problems in statistical machine learning. Among several themes in this thesis, we are interested in: (i) gaining a better understanding of existing first-order methods through the development of novel computational guarantees and by establishing novel connections, (ii) developing new methods that, in many situations, better promote structured solutions, and (iii) gaining a better understanding of and developing new statistical algorithms through their novel connections with some first-order methods.

The results in chapters 2, 4, and 5 are also included in the papers [34], [35], and [36], respectively. Some results from these papers have been omitted or only briefly commented on in this thesis. In the remainder of this chapter, we provide a very brief summary for each of the remaining chapters of this thesis.

**Chapter 2: New Analysis and Results for the Frank-Wolfe Method.** The Frank-Wolfe method for smooth convex optimization problems has received substantial renewed interest in recent years due in part to its applicability in several areas of machine learning and statistics. In this chapter, we present new results for the Frank-Wolfe method, with a particular focus on: (i) novel computational guarantees that apply for any step-size sequence, (ii) a novel adjustment to the basic algorithm to better account for warm-start information, and (iii) extensions of the computational guarantees that hold in the presence of approximate subproblem and/or gradient computations.

**Chapter 3: Dual Averaging as a Unifying Viewpoint of Greedy First-Order Methods.** Especially in huge-scale applications, greedy first-order methods offer relatively low computational requirements per iteration in addition to other appealing computational properties. In this chapter, we present precise equivalences between the dual averaging method for non-smooth optimization and two greedy first-order methods — (i) greedy coordinate descent, and (ii) the Frank-Wolfe method. The dual averaging method requires specifying two sequences: (a) the “step-size” sequence \( \{\alpha_k\} \) which defines the linear model based on subgradients of the objective function, and (b) the “regularization” sequence \( \{\beta_k\} \) which de-
fines the relative importance of the linear model \textit{vis-à-vis} the prox function. It turns out that greedy coordinate descent corresponds precisely to a well-known specification of these two sequences – namely, that corresponding to the Mirror Descent method. On the other hand, Frank-Wolfe implies an entirely new specification for these two sequences – one that places a much larger amount of weight on the \( \{ \beta_k \} \) sequence as compared to Mirror Descent. We present novel computational guarantees for this new specification of the step-size and regularization sequences.

Chapter 4: \textbf{An Extended Frank-Wolfe Method with “In-Face” Directions, and its Application to Low-Rank Matrix Completion.} Recently, the problem of \textit{matrix completion}, where one observes a small fraction of (noisy) entries of a matrix and is asked to estimate the remaining missing entries, has received substantial attention by a variety of researchers due to its diverse applications, perhaps most notably in the area of recommender systems and in particular to the Netflix Challenge. One often assumes that the underlying true data matrix (before possibly being contaminated by noise) has low-rank structure. Fundamental contributions in this area have demonstrated that, in many circumstances, a low-rank estimate that fits the data well may be extracted by solving a surrogate convex optimization problem based on nuclear norm regularization. Nevertheless, there are still gaps in understanding the capabilities of algorithms to actually deliver high-quality, low-rank solutions.

We present an extension of the Frank-Wolfe method that is designed to induce near-optimal low-rank solutions for nuclear norm regularized matrix completion and, for more general problems, induces near-optimal “well-structured” solutions. Our extended Frank-Wolfe method is based on a new approach to generating “in-face” directions at each iteration. Due to a precise characterization of the facial structure of the nuclear norm ball \cite{91}, our extension is better suited to deliver near-optimal low-rank solutions in the context of matrix completion. We establish general computational guarantees for the method and, in the case of matrix completion, the guarantees trade off efficiency in computing near-optimal solutions.
with upper bounds on the rank of iterates. We present extensive computational results on both real and artificial datasets that demonstrate that our extended Frank-Wolfe method (in different versions) shows significant computational advantages over existing related approaches, in terms of delivering low rank and low run time to compute a target optimality gap.

Chapter 5: A New Perspective on Boosting in Linear Regression. Boosting is a very popular and successful paradigm in supervised statistical learning (in particular, linear regression problems) for combining several “weak” learners into a powerful and predictive model. Boosting is often applied in the high-dimensional setting where explicit regularization schemes such as the Lasso are also very popular and powerful statistical techniques. Up until this point there has been no theoretical work investigating how boosting methods in linear regression tradeoff between data-fidelity and regularization in a manner akin to the Lasso. Furthermore, although there has been some theoretical work exploring the connections between boosting and the Lasso, there has been no work demonstrating how boosting and the Lasso might be related in a general setting without restrictive assumptions on the given dataset.

We show that several boosting algorithms in linear regression can be seen as instances of the classical subgradient descent method. Our framework suggests a simple modification to the well-known boosting algorithms that yields new boosting algorithms that are closely related to the Lasso. For all of the boosting algorithms studied, we use the tools of modern first-order methods to derive the first-ever comprehensive computational guarantees that apply for any dataset and that show how the amount of data-fidelity and regularization are controlled directly by algorithmic parameters (and thus how they tradeoff one another).

Chapter 6: A New Perspective on Boosting in Logistic Regression and AdaBoost. In this chapter, we consider boosting in the context of binary classification problems. We consider two boosting algorithms: (i) LogitBoost, which is naturally tied to the logistic loss
function, and (ii) AdaBoost, which is naturally tied to the exponential loss function. We interpret both of these algorithms in two ways – as greedy coordinate descent applied to minimize the logistic/log-exponential loss functions, and as mirror descent applied to related auxiliary problems. (The fact that we can interpret each algorithm in two ways is part of a more general phenomena explored in Chapter 3.) Motivated by the notion that logistic regression is most appropriate when the data is not linearly separable, we introduce a novel condition number to measure the degree of non-separability of a given dataset. We combine our new condition number with standard greedy coordinate descent analysis to yield new computational guarantees for the LogitBoost algorithm. We also define a related condition number for the log-exponential loss function, and we utilize guarantees from both greedy coordinate descent and mirror descent to inform the convergence behavior of AdaBoost.
Chapter 2

New Analysis and Results for the Frank-Wolfe Method

2.1 Introduction

Our interest in this chapter is the Frank-Wolfe method for convex optimization, which is also referred to as the conditional gradient method. The original Frank-Wolfe method, developed for smooth convex optimization on a polytope, dates back to Frank and Wolfe [33], and was generalized to the more general smooth convex objective function over a bounded convex feasible region thereafter, see for example Demyanov and Rubinov [23], Dunn and Harshbarger [30], Dunn [28], [29], also Levitin and Polyak [63] and Polyak [80]. More recently there has been renewed interest in the Frank-Wolfe method due to some of its properties that we will shortly discuss, see for example Clarkson [21], Hazan [52], Jaggi [54], Giesen et al. [46], and most recently Harchaoui et al. [49], Lan [62] and Temlyakov [93]. The Frank-Wolfe method is premised on being able to easily solve (at each iteration) linear optimization problems over the feasible region of interest. This is in contrast to other first-order methods, such as the accelerated methods of Nesterov [74, 73], which are premised on being able to easily solve (at each iteration) certain projection problems defined by a strongly convex
prox function. In many applications, solving a linear optimization subproblem is much simpler than solving the relevant projection subproblem. Moreover, in many applications the solutions to the linear optimization subproblems are often highly structured and exhibit particular sparsity and/or low-rank properties, which the Frank-Wolfe method is able to take advantage of as follows. The Frank-Wolfe method solves one subproblem at each iteration and produces a sequence of feasible solutions that are each a convex combination of all previous subproblem solutions, for which one can derive an $O(\frac{1}{k})$ rate of convergence for appropriately chosen step-sizes. Due to the structure of the subproblem solutions and the fact that iterates are convex combinations of subproblem solutions, the feasible solutions returned by the Frank-Wolfe method are also typically very highly-structured. For example, when the feasible region is the unit simplex $\Delta_n := \{\lambda \in \mathbb{R}^n : e^T\lambda = 1, \lambda \geq 0\}$ and the linear optimization oracle always returns an extreme point, then the Frank-Wolfe method has the following sparsity property: the solution that the method produces at iteration $k$ has at most $k$ non-zero entries. (This observation generalizes to the matrix optimization setting: if the feasible region is a ball induced by the nuclear norm, then at iteration $k$ the rank of the matrix produced by the method is at most $k$.) In many applications, such structural properties are highly desirable, and in such cases the Frank-Wolfe method may be more attractive than the faster accelerated methods, even though the Frank-Wolfe method has a slower rate of convergence.

The first set of contributions in this chapter concern computational guarantees for arbitrary step-size sequences. In Section 2.2 we present a new complexity analysis of the Frank-Wolfe method wherein we derive an exact functional dependence of the complexity bound at iteration $k$ as a function of the step-size sequence $\{\tilde{\alpha}_k\}$. We derive bounds on the deviation from the optimal objective function value (and on the duality gap in the presence of minmax structure), and on the so-called FW gaps, which may be interpreted as specially structured duality gaps. In Section 2.3 we use the technical theorems developed in Section 2.2 to derive computational guarantees for a variety of simple step-size rules including the well-studied step-size rule $\tilde{\alpha}_k := \frac{2}{k+2}$, simple averaging, and constant step-sizes. Our analysis
retains the well-known optimal $O\left(\frac{1}{k}\right)$ rate (optimal for linear optimization oracle-based methods [62], following also from [54]) when the step-size is either given by the rule $\alpha_k := \frac{2}{k+2}$ or is determined by a line-search. We also derive an $O\left(\frac{\ln(k)}{k}\right)$ rate for both the case when the step-size is given by simple averaging and in the case when the step-size is simply a suitably chosen constant.

The second set of contributions in this chapter concern “warm-start” step-size rules and associated computational guarantees that reflect the quality of the given initial iterate. The $O\left(\frac{1}{k}\right)$ computational guarantees associated with the step-size sequence $\bar{\alpha}_k := \frac{2}{k+2}$ are independent of quality of the initial iterate. This is good if the objective function value of the initial iterate is very far from the optimal value, as the computational guarantee is independent of the poor quality of the initial iterate. But if the objective function value of the initial iterate is moderately close to the optimal value, one would want the Frank-Wolfe method, with an appropriate step-size sequence, to have computational guarantees that reflect the closeness to optimality of the initial objective function value. In Section 2.4, we introduce a modification of the $\bar{\alpha}_k := \frac{2}{k+2}$ step-size rule that incorporates the quality of the initial iterate. Our new step-size rule maintains the $O\left(\frac{1}{k}\right)$ complexity bound but now the bound is enhanced by the quality of the initial iterate. We also introduce a dynamic version of this warm start step-size rule, which dynamically incorporates all new bound information at each iteration. For the dynamic step-size rule, we also derive a $O\left(\frac{1}{k}\right)$ complexity bound that depends naturally on all of the bound information obtained throughout the course of the algorithm.

The third set of contributions concern computational guarantees in the presence of approximate computation of gradients and linear optimization subproblem solutions. In Section 2.5, we first consider a variation of the Frank-Wolfe method where the linear optimization subproblem at iteration $k$ is solved approximately to an (additive) absolute accuracy of $\delta_k$. We show that, independent of the choice of step-size sequence $\{\bar{\alpha}_k\}$, the Frank-Wolfe method does not suffer from an accumulation of errors in the presence of approximate subproblem solutions. We extend the “technical” complexity theorems of Section 2.2 which
imply, for instance, that when an optimal step-size such as \( \bar{\alpha}_k := \frac{2}{k+2} \) is used and the \( \{\delta_k\} \) accuracy sequence is a constant \( \delta \), then a solution with accuracy \( O(\frac{1}{k} + \delta) \) can be achieved in \( k \) iterations. We next examine variations of the Frank-Wolfe method where exact gradient computations are replaced with inexact gradient computations, under two different models of inexact gradient computations. We show that all of the complexity results under the previously examined approximate subproblem solution case (including, for instance, the non-accumulation of errors) directly apply to the case where exact gradient computations are replaced with the \( \delta \)-oracle approximate gradient model introduced by d’Aspremont \[22\]. We also examine replacing exact gradient computations with the \( (\delta, L) \)-oracle model introduced by Devolder et al. \[25\]. In this case the Frank-Wolfe method suffers from an accumulation of errors under essentially any step-size sequence \( \{\bar{\alpha}_k\} \). These results provide some insight into the inherent tradeoffs faced in choosing among several first-order methods.

2.1.1 Notation

Let \( E \) be a finite-dimensional real vector space with dual vector space \( E^* \). For a given \( s \in E^* \) and a given \( \lambda \in E \), let \( s^T\lambda \) denote the evaluation of the linear functional \( s \) at \( \lambda \). For a norm \( \| \cdot \| \) on \( E \), let \( B(c, r) = \{ \lambda \in E : \|\lambda - c\| \leq r \} \). The dual norm \( \| \cdot \|^* \) on the space \( E^* \) is defined by \( \|s\|^* := \max_{\lambda \in B(0,1)} \{ s^T\lambda \} \) for a given \( s \in E^* \). The notation \( \tilde{v} \leftarrow \arg \max_{v \in S} \{ f(v) \} \) denotes assigning \( \tilde{v} \) to be any optimal solution of the problem \( \max_{v \in S} \{ f(v) \} \).

2.2 The Frank-Wolfe Method

We recall the Frank-Wolfe method for convex optimization, see Frank and Wolfe \[33\], also Demyanov and Rubinov \[23\], Levitin and Polyak \[63\], and Polyak \[80\], stated here for maximization problems:

\[
\max_{\lambda} \quad h(\lambda) \\
\text{s.t.} \quad \lambda \in Q ,
\]
where $Q \subset E$ is convex and compact, and $h(\cdot) : Q \to \mathbb{R}$ is concave and differentiable on $Q$. Let $h^*$ denote the optimal objective function value of (2.1). The basic Frank-Wolfe method is presented in Method 2.1, where the main computational requirement at each iteration is to solve a linear optimization problem over $Q$ in step (2.) of the method. The step-size $\bar{\alpha}_k$ in step (4.) could be chosen by inexact or exact line-search, or by a pre-determined or dynamically determined step-size sequence $\{\bar{\alpha}_k\}$. Also note that the version of the Frank-Wolfe method in Method 2.1 does not allow a (full) step-size $\bar{\alpha}_k = 1$, the reasons for which will become apparent below.

**Method 2.1** Frank-Wolfe Method for maximizing $h(\lambda)$

Initialize at $\lambda_1 \in Q$, (optional) initial upper bound $B_0$, $k \leftarrow 1$.

At iteration $k$:

1. Compute $\nabla h(\lambda_k)$.
2. Compute $\bar{\lambda}_k \leftarrow \arg \max_{\lambda \in Q} \{h(\lambda_k) + \nabla h(\lambda_k)^T(\lambda - \lambda_k)\}$.
   
   $B_k^w \leftarrow h(\lambda_k) + \nabla h(\lambda_k)^T(\bar{\lambda}_k - \lambda_k)$.
   
   $G_k \leftarrow \nabla h(\lambda_k)^T(\bar{\lambda}_k - \lambda_k)$.

3. (Optional: compute other upper bound $B_k^o$), update best bound $B_k \leftarrow \min\{B_{k-1}^w, B_k^w, B_k^o\}$.

4. Set $\lambda_{k+1} \leftarrow \lambda_k + \bar{\alpha}_k(\bar{\lambda}_k - \lambda_k)$, where $\bar{\alpha}_k \in [0, 1)$.

As a consequence of solving the linear optimization problem in step (2.) of the method, one conveniently obtains the following upper bound on the optimal value $h^*$ of (2.1):

$$B_k^w := h(\lambda_k) + \nabla h(\lambda_k)^T(\bar{\lambda}_k - \lambda_k),$$

and it follows from the fact that the linearization of $h(\cdot)$ at $\lambda_k$ dominates $h(\cdot)$ that $B_k^w$ is a valid upper bound on $h^*$. We also study the quantity $G_k$:

$$G_k := B_k^w - h(\lambda_k) = \nabla h(\lambda_k)^T(\bar{\lambda}_k - \lambda_k),$$

which we refer to as the “FW gap” at iteration $k$ for convenience. Note that $G_k \geq h^* - h(\lambda_k) \geq 0$. The use of the upper bound $B_k^w$ dates to the original 1956 paper of Frank and Wolfe [33]. As early as 1970, Demyanov and Rubinov [23] used the FW gap quantities extensively in
their convergence proofs of the Frank-Wolfe method, and perhaps this quantity was used even earlier. In certain contexts, $G_k$ is an important quantity by itself, see for example Hearn [53], Khachiyan [57] and Giesen et al. [46]. Indeed, Hearn [53] studies basic properties of the FW gaps independent of their use in any algorithmic schemes. For results concerning upper bound guarantees on $G_k$ for specific and general problems see Khachiyan [57], Clarkson [21], Hazan [52], Jaggi [54], Giesen et al. [46], and Harchaoui et al. [49]. Both $B^w_k$ and $G_k$ are computed directly from the solution of the linear optimization problem in step (2.) and are recorded therein for convenience.

In some of our analysis of the Frank-Wolfe method, the computational guarantees will depend on the quality of upper bounds on $h^*$. In addition to the Wolfe bound $B^w_k$, step (3.) allows for an “optional other upper bound $B^o_k$ ” that also might be computed at iteration $k$. Sometimes there is structural knowledge of an upper bound as a consequence of a dual problem associated with (2.1), as when $h(\cdot)$ is conveyed with minmax structure, namely:

$$h(\lambda) = \min_{x \in P} \phi(x, \lambda), \quad (2.4)$$

where $P$ is a closed convex set and $\phi(\cdot, \cdot) : P \times Q \to \mathbb{R}$ is a continuous function that is convex in the first variable $x$ and concave in the second variable $\lambda$. In this case define the convex function $f(\cdot) : P \to \mathbb{R}$ given by $f(x) := \max_{\lambda \in Q} \phi(x, \lambda)$ and consider the following duality paired problems:

$$(\text{Primal}): \min_{x \in P} f(x) \quad \text{and} \quad (\text{Dual}): \max_{\lambda \in Q} h(\lambda), \quad (2.5)$$

where the dual problem corresponds to our problem of interest (2.1). Weak duality holds, namely $h(\lambda) \leq h^* \leq f(x)$ for all $x \in P, \lambda \in Q$. At any iterate $\lambda_k \in Q$ of the Frank-Wolfe method one can construct a “minmax” upper bound on $h^*$ by considering the variable $x$ in that structure:

$$B^m_k := f(x_k) := \max_{\lambda \in Q} \{\phi(x_k, \lambda)\} \quad \text{where} \quad x_k \in \arg\min_{x \in P} \{\phi(x, \lambda_k)\}, \quad (2.6)$$
and it follows from weak duality that \( B^o_k := B^m_k \) is a valid upper bound for all \( k \). Notice that \( x_k \) defined above is the “optimal response” to \( \lambda_k \) in a minmax sense and hence is a natural choice of duality-paired variable associated with the variable \( \lambda_k \). Under certain regularity conditions, for instance when \( h(\cdot) \) is globally differentiable on \( E \), one can show that \( B^m_k \) is at least as tight a bound as Wolfe’s bound, namely \( B^m_k \leq B^w_k \) for all \( k \) (see Proposition A.1), and therefore the FW gap \( G_k \) conveniently bounds this minmax duality gap: \( B^m_k - h(\lambda_k) \leq B^w_k - h(\lambda_k) = G_k \).

(Indeed, in the minmax setting notice that the optimal response \( x_k \) in (2.6) is a function of the current iterate \( \lambda_k \) and hence \( f(x_k) - h(\lambda_k) = B^m_k - h(\lambda_k) \) is not just any duality gap but rather is determined completely by the current iterate \( \lambda_k \). This special feature of the duality gap \( B^m_k - h(\lambda_k) \) is exploited in the application of the Frank-Wolfe method to rounding of polytopes [57], parametric optimization on the spectrahedron [46], and perhaps elsewhere as well, where bounds on the FW gap \( G_k \) are used to bound \( B^m_k - h(\lambda_k) \) directly.)

We also mention that in some applications there might be exact knowledge of the optimal value \( h^* \), such as in certain linear regression and/or machine learning applications where one knows \textit{a priori} that the optimal value of the loss function is zero. In these situations one can set \( B^o_k \leftarrow h^* \).

Towards stating and proving complexity bounds for the Frank-Wolfe method, we use the following curvature constant \( C_{h,Q} \), which is defined to be the minimal value of \( C \) satisfying:

\[
h(\lambda + \alpha(\bar{\lambda} - \lambda)) \geq h(\lambda) + \nabla h(\lambda)^T(\alpha(\bar{\lambda} - \lambda)) - \frac{1}{2}C\alpha^2 \quad \text{for all } \lambda, \bar{\lambda} \in Q \text{ and all } \alpha \in [0, 1]. \tag{2.7}
\]

(This notion of curvature was introduced by Clarkson [21] and extended in Jaggi [54].) For any choice of norm \( \| \cdot \| \) on \( E \), let \( \text{Diam}_Q \) denote the diameter of \( Q \) measured with the norm \( \| \cdot \| \), namely \( \text{Diam}_Q := \max_{\lambda, \bar{\lambda} \in Q} \{ \| \lambda - \bar{\lambda} \| \} \) and let \( L_{h,Q} \) be the Lipschitz constant for \( \nabla h(\cdot) \) on \( Q \), namely \( L_{h,Q} \) is the smallest constant \( L \) for which it holds that:

\[
\| \nabla h(\lambda) - \nabla h(\bar{\lambda}) \| \leq L\| \lambda - \bar{\lambda} \| \quad \text{for all } \lambda, \bar{\lambda} \in Q .
\]
It is straightforward to show that \( C_{h,Q} \) is bounded above by the more classical metrics \( \text{Diam}_Q \) and \( L_{h,Q} \), namely

\[
C_{h,Q} \leq L_{h,Q}(\text{Diam}_Q)^2, \tag{2.8}
\]

see [54]; we present a short proof of this inequality in Proposition A.2 for completeness. In contrast to other (proximal) first-order methods, the Frank-Wolfe method does not depend on a choice of norm. The norm invariant definition of \( C_{h,Q} \) and the fact that (2.8) holds for any norm are therefore particularly appealing properties of \( C_{h,Q} \) as a behavioral measure for the Frank-Wolfe method.

As a prelude to stating our main technical results, we define the following two auxiliary sequences, where \( \alpha_k \) and \( \beta_k \) are functions of the first \( k \) step-size sequence values, \( \bar{\alpha}_1, \ldots, \bar{\alpha}_k \), from the Frank-Wolfe method:

\[
\beta_k = \frac{1}{k-1} \prod_{j=1}^{k-1} (1 - \bar{\alpha}_j), \quad \alpha_k = \frac{\beta_k \bar{\alpha}_k}{1 - \bar{\alpha}_k}, \quad k \geq 1 . \tag{2.9}
\]

(Here and in what follows we use the conventions: \( \prod_{j=1}^{0} \cdot = 1 \) and \( \sum_{i=1}^{0} \cdot = 0 \).)

The following two theorems are our main technical constructs that will be used to develop the results herein. The first theorem concerns optimality gap bounds.

**Theorem 2.1.** Consider the iterate sequences of the Frank-Wolfe method (Method 2.1) \( \{\lambda_k\} \) and \( \{\tilde{\lambda}_k\} \) and the sequence of upper bounds \( \{B_k\} \) on \( h^* \), using the step-size sequence \( \{\bar{\alpha}_k\} \). For the auxiliary sequences \( \{\alpha_k\} \) and \( \{\beta_k\} \) given by (2.9), and for any \( k \geq 0 \), the following inequality holds:

\[
B_k - h(\lambda_{k+1}) \leq B_k - h(\lambda_1) + \frac{1}{2} C_{h,Q} \sum_{i=1}^{k} \frac{\alpha_i^2}{\beta_{i+1}} . \tag{2.10}
\]

(The summation expression in the rightmost term above appears also in the bound given for the dual averaging method of Nesterov [75]. Indeed, this is no coincidence as the sequences \( \{\alpha_k\} \) and \( \{\beta_k\} \) given by (2.9) arise precisely from a connection between the Frank-Wolfe
method and the dual averaging method. If we define \( s_k := \lambda_0 + \sum_{i=0}^{k-1} \alpha_i \tilde{\lambda}_i \), then one can interpret the sequence \( \{s_k\} \) as the sequence of dual variables in a particular instance of the dual averaging method. This connection underlies the proof of Theorem 2.1 and the careful reader will notice the similarities between the proof of Theorem 2.1 and the proof of Theorem 1 in [75]. For this reason we will henceforth refer to the sequences (2.9) as the “dual averages” sequences associated with \( \{\tilde{\alpha}_k\} \).

The second theorem concerns the FW gap values \( G_k \) from step (2.) in particular.

**Theorem 2.2.** Consider the iterate sequences of the Frank-Wolfe method (Method 2.1) \( \{\lambda_k\} \) and \( \{\tilde{\lambda}_k\} \), the sequence of upper bounds \( \{B_k\} \) on \( h^* \), and the sequence of FW gaps \( \{G_k\} \) from step (2.), using the step-size sequence \( \{\tilde{\alpha}_k\} \). For the auxiliary sequences \( \{\alpha_k\} \) and \( \{\beta_k\} \) given by (2.9), and for any \( \ell \geq 0 \) and \( k \geq \ell + 1 \), the following inequality holds:

\[
\min_{i \in \{\ell+1, \ldots, k\}} G_i \leq \frac{1}{\sum_{i=\ell+1}^{k} \tilde{\alpha}_i} \left[ B_{\ell} - h(\lambda_1) \right] + \frac{1}{2} C_{h,Q} \sum_{i=1}^{\ell} \frac{\alpha_i^2}{\beta_{i+1}} + \frac{1}{2} C_{h,Q} \sum_{i=\ell+1}^{k} \frac{\alpha_i^2}{\beta_{i+1}} \sum_{i=\ell+1}^{k} \tilde{\alpha}_i. \tag{2.11}
\]

Theorems 2.1 and 2.2 can be applied to yield specific complexity results for any specific step-size sequence \( \{\tilde{\alpha}_k\} \) (satisfying the mild assumption that \( \tilde{\alpha}_k < 1 \)) through the use of the implied \( \{\alpha_k\} \) and \( \{\beta_k\} \) dual averages sequences. This is shown for several useful step-size sequences in the next section.

**Proof of Theorem 2.1** We will show the slightly more general result for \( k \geq 0 \):

\[
\min \{B, B_k\} - h(\lambda_{k+1}) \leq \frac{B - h(\lambda_1)}{\beta_{k+1}} + \frac{1}{2} C_{h,Q} \sum_{i=1}^{k} \frac{\alpha_i^2}{\beta_{i+1}} \quad \text{for any } B, \tag{2.12}
\]

from which (2.10) follows by substituting \( B = B_k \) above.

For \( k = 0 \) the result follows trivially since \( \beta_1 = 1 \) and the summation term on the right side of (2.12) is zero by the conventions for null products and summations stated earlier. For \( k \geq 1 \), we begin by observing that the following equalities hold for the dual averages
sequences (2.9):

\[ \beta_{i+1} - \beta_i = \bar{\alpha}_i \beta_{i+1} = \alpha_i \quad \text{and} \quad \beta_{i+1} \bar{\alpha}_i^2 = \frac{\alpha_i^2}{\beta_{i+1}} \quad \text{for} \ i \geq 1 , \quad (2.13) \]

and

\[ 1 + \sum_{i=1}^{k} \alpha_i = \beta_{k+1} \quad \text{for} \ k \geq 1 . \quad (2.14) \]

We then have for \( i \geq 1 \):

\[
\beta_{i+1} h(\lambda_{i+1}) \geq \beta_{i+1} \left[ h(\lambda_i) + \nabla h(\lambda_i)^T (\bar{\lambda}_i - \lambda_i) \bar{\alpha}_i - \frac{1}{2} \bar{\alpha}_i^2 C_{h,Q} \right] \\
= \beta_i h(\lambda_i) + (\beta_{i+1} - \beta_i) h(\lambda_i) + \beta_{i+1} \bar{\alpha}_i \nabla h(\lambda_i)^T (\bar{\lambda}_i - \lambda_i) - \frac{1}{2} \beta_{i+1} \bar{\alpha}_i^2 C_{h,Q} \\
= \beta_i h(\lambda_i) + \alpha_i h(\lambda_i) + \alpha_i \nabla h(\lambda_i)^T (\bar{\lambda}_i - \lambda_i) - \frac{1}{2} \beta_{i+1} \bar{\alpha}_i^2 C_{h,Q} \\
= \beta_i h(\lambda_i) + \alpha_i \left[ h(\lambda_i) + \nabla h(\lambda_i)^T (\bar{\lambda}_i - \lambda_i) \right] - \frac{1}{2} \beta_{i+1} \bar{\alpha}_i^2 C_{h,Q} \\
= \beta_i h(\lambda_i) + \alpha_i B_i w - \frac{1}{2} \beta_{i+1} \bar{\alpha}_i^2 C_{h,Q} .
\]

The inequality in the first line above follows from the definition of \( C_{h,Q} \) in (2.7) and \( \lambda_{i+1} - \lambda_i = \bar{\alpha}_i (\bar{\lambda}_i - \lambda_i) \). The second equality above uses the identities (2.13), and the fourth equality uses the definition of the Wolfe upper bound (2.2). Rearranging and summing the above over \( i \), it follows that for any scalar \( B \):

\[
B + \sum_{i=1}^{k} \alpha_i B_i w \leq B + \beta_{k+1} h(\lambda_{k+1}) - \beta_1 h(\lambda_1) + \frac{1}{2} \sum_{i=1}^{k} \alpha_i^2 \beta_{i+1} C_{h,Q} . \quad (2.15)
\]
Therefore
\[
\min\{B, B_k\} \beta_{k+1} = \min\{B, B_k\} \left(1 + \sum_{i=1}^{k} \alpha_i\right)
\leq B + \sum_{i=1}^{k} \alpha_i B_i^w
\leq B + \beta_{k+1} h(\lambda_{k+1}) - h(\lambda_1) + \frac{1}{2} \sum_{i=1}^{k} \frac{\alpha_i^2}{\beta_{i+1}} C_{h,Q}
\]
where the first equality above uses identity (2.14), the first inequality uses the fact that \(B_k \leq B_i^w\) for \(i \leq k\), and the second inequality uses (2.15) and the fact that \(\beta_1 = 1\). The result then follows by dividing by \(\beta_{k+1}\) and rearranging terms.

**Proof of Theorem 2.2** For \(i \geq 1\) we have:
\[
h(\lambda_{i+1}) \geq h(\lambda_i) + \nabla h(\lambda_i)^T (\tilde{\lambda}_i - \lambda_i) \tilde{\alpha}_i - \frac{1}{2} \tilde{\alpha}_i^2 C_{h,Q}
\]
\[
= h(\lambda_i) + \tilde{\alpha}_i G_i - \frac{1}{2} \tilde{\alpha}_i^2 C_{h,Q}
\]
(2.16)
where the inequality follows from the definition of the curvature constant in (2.7), and the equality follows from the definition of the FW gap in (2.3). Summing the above over \(i \in \{\ell+1, \ldots, k\}\) and rearranging yields:
\[
\sum_{i=\ell+1}^{k} \tilde{\alpha}_i G_i \leq h(\lambda_{k+1}) - h(\lambda_{\ell+1}) + \sum_{i=\ell+1}^{k} \frac{1}{2} \tilde{\alpha}_i^2 C_{h,Q}
\]
(2.17)
Combining (2.17) with Theorem 2.1 we obtain:
\[
\sum_{i=\ell+1}^{k} \tilde{\alpha}_i G_i \leq h(\lambda_{k+1}) - B_\ell + \frac{B_\ell - h(\lambda_1)}{\beta_{\ell+1}} + \frac{1}{2} C_{h,Q} \sum_{i=1}^{\ell} \frac{\alpha_i^2}{\beta_{i+1}} + \sum_{i=\ell+1}^{k} \frac{1}{2} \tilde{\alpha}_i^2 C_{h,Q}
\]
and since $B_\ell \geq h^* \geq h(\lambda_{k+1})$ we obtain:

\[
\left( \min_{i \in \{\ell+1, \ldots, k\}} G_i \right) \left( \sum_{i=\ell+1}^{k} \tilde{\alpha}_i \right) \leq \sum_{i=\ell+1}^{k} \tilde{\alpha}_i G_i \leq \frac{B_\ell - h(\lambda_1)}{\beta_{\ell+1}} + \frac{1}{2} C_{h,Q} \sum_{i=1}^{\ell} \frac{\alpha_i^2}{\beta_{i+1}} + \sum_{i=\ell+1}^{k} \frac{1}{2} \tilde{\alpha}_i^2 C_{h,Q},
\]

and dividing by $\sum_{i=\ell+1}^{k} \tilde{\alpha}_i$ yields the result.

2.3 Computational Guarantees for Specific Step-size Sequences

Herein we use Theorems 2.1 and 2.2 to derive computational guarantees for a variety of specific step-size sequences.

It will be useful to consider a version of the Frank-Wolfe method wherein there is a single “pre-start” step. In this case we are given some $\lambda_0 \in Q$ and some upper bound $B_{-1}$ on $h^*$ (one can use $B_{-1} = +\infty$ if no information is available) and we proceed like any other iteration except that in step (4.) we set $\lambda_1 \leftarrow \tilde{\lambda}_0$, which is equivalent to setting $\tilde{\alpha}_0 := 1$. We have the following property for the pre-start step.

**Proposition 2.1.** Let $\lambda_1$ and $B_0$ be computed by the pre-start step. Then $B_0 - h(\lambda_1) \leq \frac{1}{2} C_{h,Q}$.

**Proof.** We have $\lambda_1 = \tilde{\lambda}_0$ and $B_0 \leq B_0^w$, whereby from the definition of $C_{h,Q}$ using $\alpha = 1$ we have:

\[
h(\lambda_1) = h(\tilde{\lambda}_0) \geq h(\lambda_0) + \nabla h(\lambda_0)^T (\tilde{\lambda}_0 - \lambda_0) - \frac{1}{2} C_{h,Q} = B_0^w - \frac{1}{2} C_{h,Q} \geq B_0 - \frac{1}{2} C_{h,Q},
\]

and the result follows by rearranging terms.

The proofs of the bounds presented in the following subsections are omitted for brevity; they are included in the full paper [34] and they all follow the same strategy, mentioned previously, of deriving the implied dual averages sequences (2.9) and then substituting these
sequence values into Theorems 2.1 and 2.2 (and perhaps using some elementary facts about sequences and series).

2.3.1 A Well-studied Step-size Sequence

Suppose we initiate the Frank-Wolfe method with the pre-start step from a given value \( \lambda_0 \in Q \) (which by definition assigns the step-size \( \bar{\alpha}_0 = 1 \) as discussed earlier), and then use the step-size \( \bar{\alpha}_i = 2/(i + 2) \) for \( i \geq 1 \). This can be written equivalently as:

\[
\bar{\alpha}_i = \frac{2}{i + 2} \quad \text{for} \ i \geq 0 .
\] (2.18)

Computational guarantees for this sequence appeared in Clarkson [21], Hazan [52] (with a corrected proof in Giesen et al. [46]), and Jaggi [54]. In unpublished correspondence with one of the authors in 2007, Nemirovski [71] presented a short inductive proof of convergence of the Frank-Wolfe method using this step-size rule.

**Bound 2.1.** Under the step-size sequence (2.18), the following inequalities hold for all \( k \geq 1 \):

\[
B_k - h(\lambda_{k+1}) \leq \frac{2C_{h,Q}}{k + 4}
\] (2.19)

and

\[
\min_{i \in \{1, \ldots, k\}} G_i \leq \frac{4.5C_{h,Q}}{k} .
\] (2.20)

The bound (2.19) is a very minor improvement over that in Hazan [52], Giesen et al. [46], Jaggi [54], and Harchaoui et al. [49], as the denominator is additively larger by 1 (after accounting for the pre-start step and the different indexing conventions). The bound (2.20) is a modification of the original bound in Jaggi [54], and is also a slight improvement of the bound in Harchaoui et al. [49] inasmuch as the denominator is additively larger by 1 and the bound is valid for all \( k \geq 1 \).
2.3.2 Simple Averaging

Consider the following step-size sequence:

$$\bar{\alpha}_i = \frac{1}{i + 1} \quad \text{for } i \geq 0 ,$$  \hspace{1cm} (2.21)

where, as with the step-size sequence \(2.18\), we write \(\bar{\alpha}_0 = 1\) to indicate the use of the pre-start step. It follows from a simple inductive argument that, under the step-size sequence \(2.21\), \(\lambda_{k+1}\) is the simple average of \(\lambda_0, \lambda_1, \ldots, \lambda_k\), i.e., we have

$$\lambda_{k+1} = \frac{1}{k + 1} \sum_{i=0}^{k} \lambda_i \quad \text{for all } k \geq 0 .$$

**Bound 2.2.** Under the step-size sequence \(2.21\), the following inequality holds for all \(k \geq 0\):

$$B_k - h(\lambda_{k+1}) \leq \frac{1}{2} C_{h,Q} (1 + \ln(k + 1)) \frac{1}{k + 1} ,$$  \hspace{1cm} (2.22)

and the following inequality holds for all \(k \geq 2\):

$$\min_{i \in \{1, \ldots, k\}} G_i \leq \frac{3}{4} C_{h,Q} \frac{2.3 + 2 \ln(k)}{k - 1} .$$  \hspace{1cm} (2.23)

2.3.3 Constant Step-size

Given \(\bar{\alpha} \in (0, 1)\), consider using the following constant step-size rule:

$$\bar{\alpha}_i = \bar{\alpha} \quad \text{for } i \geq 1 .$$  \hspace{1cm} (2.24)

**Bound 2.3.** Under the step-size sequence \(2.24\), the following inequality holds for all \(k \geq 1\):

$$B_k - h(\lambda_{k+1}) \leq (B_k - h(\lambda_1)) (1 - \bar{\alpha})^k + \frac{1}{2} C_{h,Q} \left[ \bar{\alpha} - \bar{\alpha} (1 - \bar{\alpha})^k \right] .$$  \hspace{1cm} (2.25)
If the pre-start step is used, then:

\[
B_k - h(\lambda_{k+1}) \leq \frac{1}{2} C_{h,Q} \left[ (1 - \bar{\alpha})^{k+1} + \bar{\alpha} \right].
\] (2.26)

If we decide \textit{a priori} to run the Frank-Wolfe method for \( k \) iterations after the pre-start step, then we can optimize the bound (2.26) with respect to \( \bar{\alpha} \). The optimized value of \( \bar{\alpha} \) in the bound (2.26) is easily derived to be:

\[
\bar{\alpha}^* = 1 - \frac{1}{\sqrt{k} + 1}.
\] (2.27)

With \( \bar{\alpha} \) determined by (2.27), we obtain a simplified bound from (2.26) and also a guarantee for the FW Gap sequence \( \{G_i\} \) if the method is continued with the same constant step-size (2.27) for an additional \( k + 1 \) iterations.

\textbf{Bound 2.4.} \textit{If we use the pre-start step and the constant step-size sequence (2.27) for all iterations, then after \( k \) iterations the following inequality holds:}

\[
B_k - h(\lambda_{k+1}) \leq \frac{1}{2} C_{h,Q} \frac{(1 + \ln(k + 1))}{k}.
\] (2.28)

\textit{Furthermore, after \( 2k + 1 \) iterations the following inequality holds:}

\[
\min_{i \in \{1, \ldots, 2k+1\}} G_i \leq \frac{1}{2} C_{h,Q} \frac{(1 + 2 \ln(k + 1))}{k}.
\] (2.29)

It is curious to note that the bounds (2.22) and (2.28) are almost identical, although (2.28) requires fixing \textit{a priori} the number of iterations \( k \).

\textbf{2.3.4 Extensions using Line-Searches}

The original method of Frank and Wolfe [33] used a line-search to determine the next iterate \( \lambda_{k+1} \) by assigning \( \hat{\alpha}_k \leftarrow \arg \max_{\alpha \in [0,1]} \{ h(\lambda_k + \alpha(\bar{\lambda}_k - \lambda_k)) \} \) and \( \lambda_{k+1} \leftarrow \lambda_k + \hat{\alpha}_k(\bar{\lambda}_k - \lambda_k) \). When
$h(\cdot)$ is a quadratic and the dimension of the space $E$ of variables $\lambda$ is not huge, an exact line-search is easy to compute analytically, otherwise an inexact line-search can be used. It is a straightforward extension of Theorem 2.1 to show that if an exact line-search is utilized at every iteration, then the bound (2.10) holds for any choice of step-size sequence $\{\hat{\alpha}_k\}$, and not just the sequence $\{\hat{\alpha}_k\}$ of line-search step-sizes. In particular, the $O\left(\frac{1}{k}\right)$ computational guarantee (2.19) holds, as does (2.22) and (2.25), as well as the bound (2.32) to be developed in Section 2.4.

2.4 Computational Guarantees for a Warm Start

In the framework of this chapter, the well-studied step-size sequence (2.18) and associated computational guarantees (Bound 2.1) corresponds to running the Frank-Wolfe method initiated with the pre-start step from the initial point $\lambda_0$. One feature of the main computational guarantees as presented in the bounds (2.19) and (2.20) is their insensitivity to the quality of the initial point $\lambda_0$. This is good if $h(\lambda_0)$ is very far from the optimal value $h^*$, as the poor quality of the initial point does not affect the computational guarantee. But if $h(\lambda_0)$ is moderately close to the optimal value, one would want the Frank-Wolfe method, with an appropriate step-size sequence, to have computational guarantees that reflect the closeness to optimality of the initial objective function value $h(\lambda_0)$. Let us see how this can be done.

We will consider starting the Frank-Wolfe method without the pre-start step, started at an initial point $\lambda_1$, and let $C_1$ be a given estimate of the curvature constant $C_{h,Q}$. Consider the following step-size sequence:

$$\hat{\alpha}_i = \frac{2}{\frac{2C_1}{B_1-h(\lambda_1)} + i + 1} \quad \text{for } i \geq 1.$$  \hfill (2.30)

Comparing (2.30) to the well-studied step-size rule (2.18), one can think of the above step-size rule as acting “as if” the Frank-Wolfe method had run for $\frac{2C_1}{B_1-h(\lambda_1)}$ iterations before arriving at $\lambda_1$. The next result presents a computational guarantee associated with this.
step-size rule.

**Bound 2.5.** Under the step-size sequence $[2.30]$, the following inequality holds for all $k \geq 1$:

$$B_k - h(\lambda_{k+1}) \leq 2 \max\{C_1, C_{h,Q}\} \frac{2C_1}{B_1 - h(\lambda_1)} + \frac{k}{C_1}.$$  \hspace{1cm} (2.31)

Notice that in the case when $C_1 = C_{h,Q}$, the bound in (2.31) simplifies conveniently to:

$$B_k - h(\lambda_{k+1}) \leq \frac{2C_{h,Q}}{B_1 - h(\lambda_1)} + \frac{k}{C_{h,Q}}.$$  \hspace{1cm} (2.32)

Also, as a function of the estimate $C_1$ of the curvature constant, it is easily verified that the bound in (2.31) is optimized at $C_1 = C_{h,Q}$.

We remark that the bound (2.31) (or (2.32)) is small to the extent that the initial bound gap $B_1 - h(\lambda_1)$ is small, as one would want. However, to the extent that $B_1 - h(\lambda_1)$ is small, the incremental decrease in the bound due to an additional iteration is less. In other words, while the bound (2.31) is nicely sensitive to the initial bound gap, there is no longer rapid decrease in the bound in the early iterations. It is as if the algorithm had already run for \left(\frac{2C_1}{B_1 - h(\lambda_1)}\right) iterations to arrive at the initial iterate $\lambda_1$, with a corresponding dampening in the marginal value of each iteration after then. This is a structural feature of the Frank-Wolfe method that is different from first-order methods that use prox functions and/or projections.

**Proof of Bound 2.5.** Define $s = \frac{2C_1}{B_1 - h(\lambda_1)}$, whereby $\bar{\alpha}_i = \frac{2}{s + i + 1}$ for $i \geq 1$. It then is straightforward to show that the dual averages sequences (2.9) are for $i \geq 1$: 

$$\beta_i = \prod_{j=1}^{i-1} (1 - \bar{\alpha}_j)^{-1} = \prod_{j=1}^{i-1} \frac{s + j + 1}{s + j - 1} = \frac{(s + i - 1)(s + i)}{s(s + 1)},$$

and

$$\alpha_i = \frac{\beta_i \bar{\alpha}_i}{1 - \bar{\alpha}_i} = \frac{2(s + i)(s + i - 1)(s + i + 1)}{s(s + 1)(s + i + 1)(s + i - 1)} = \frac{2(s + i)}{s(s + 1)}. $$
Furthermore, we have:

\[
\begin{align*}
\sum_{i=1}^{k} \frac{\alpha_i^2}{\beta_{i+1}} &= \sum_{i=1}^{k} \frac{4(s+i)^2(s)(s+1)}{s^2(s+1)^2(s+i)(s+i+1)} = \sum_{i=1}^{k} \frac{4(s+i)}{s(s+1)(s+i+1)} \leq \frac{4k}{s(s+1)}. \quad (2.33)
\end{align*}
\]

Utilizing Theorem 2.1 and (2.33), we have for \( k \geq 1 \):

\[
\begin{align*}
B_k - h(\lambda_{k+1}) &\leq \frac{B_k - h(\lambda_1)}{\beta_{k+1}} + \frac{1}{2} C_{h,Q} \sum_{i=1}^{k} \frac{\alpha_i^2}{\beta_{i+1}} \\
&\leq \frac{s(s+1)}{(s+k)(s+k+1)} \left( B_1 - h(\lambda_1) + \frac{C_{h,Q}}{2} \cdot \frac{4k}{s(s+1)} \right) \\
&= \frac{s(s+1)}{(s+k)(s+k+1)} \left( \frac{2C_1}{s} + \frac{2kC_{h,Q}}{s(s+1)} \right) \\
&\leq \frac{2 \max\{C_1, C_{h,Q}\}}{(s+k)(s+k+1)} (s + 1 + k) \\
&= \frac{2 \max\{C_1, C_{h,Q}\}}{s+k},
\end{align*}
\]

which completes the proof. \( \Box \)

### 2.4.1 A Dynamic Version of the Warm-Start Step-size Strategy

The step-size sequence (2.30) determines all step-sizes for the Frank-Wolfe method based on two pieces of information at the initial point \( \lambda_1 \): *(i)* the initial bound gap \( B_1 - h(\lambda_1) \), and *(ii)* the given estimate \( C_1 \) of the curvature constant. The step-size sequence (2.30) is a static warm-start strategy in that all step-sizes are determined by information that is available or computed at the first iterate. Let us see how we can improve the computational guarantee by treating every iterate as if it were the initial iterate, and hence dynamically determine
the step-size sequence as a function of accumulated information about the bound gap and the curvature constant.

At the start of a given iteration $k$ of the Frank-Wolfe method, we have the iterate value $\lambda_k \in Q$ and an upper bound $B_{k-1}$ on $h^*$ from the previous iteration. We also will now assume that we have an estimate $C_{k-1}$ of the curvature constant from the previous iteration as well. Steps (2.) and (3.) of the Frank-Wolfe method then perform the computation of $\tilde{\lambda}_k, B_k$ and $G_k$. Instead of using a pre-set formula for the step-size $\bar{\alpha}_k$, we will determine the value of $\bar{\alpha}_k$ based on the current bound gap $B_k - h(\lambda_k)$ as well as on a new estimate $C_k$ of the curvature constant. (We will shortly discuss how $C_k$ is computed.) Assuming $C_k$ has been computed, and mimicking the structure of the static warm-start step-size rule (2.30), we compute the current step-size as follows:

$$\bar{\alpha}_k := \frac{2}{2C_k B_k - h(\lambda_k)} + 2,$$  \hspace{1cm} (2.34)

where we note that $\bar{\alpha}_k$ depends explicitly on the value of $C_k$. Comparing $\bar{\alpha}_k$ in (2.34) with (2.18), we interpret $\frac{2}{2C_k B_k - h(\lambda_k)}$ to be “as if” the current iteration $k$ was preceded by $\frac{2}{2C_k B_k - h(\lambda_k)}$ iterations of the Frank-Wolfe method using the standard step-size (2.18). This interpretation is also in concert with that of the static warm-start step-size rule (2.30).

We now discuss how we propose to compute the new estimate $C_k$ of the curvature constant $C_{h,Q}$ at iteration $k$. Because $C_k$ will be only an estimate of $C_{h,Q}$, we will need to require that $C_k$ (and the step-size $\bar{\alpha}_k$ (2.34) that depends explicitly on $C_k$) satisfy:

$$h(\lambda_k + \bar{\alpha}_k (\tilde{\lambda}_k - \lambda_k)) \geq h(\lambda_k) + \bar{\alpha}_k (B_k - h(\lambda_k)) - \frac{1}{2} C_k \bar{\alpha}_k^2.$$  \hspace{1cm} (2.35)

In order to find a value $C_k \geq C_{k-1}$ for which (2.35) is satisfied, we first test if $C_k := C_{k-1}$ satisfies (2.35), and if so we set $C_k \leftarrow C_{k-1}$. If not, one can perform a standard doubling strategy, testing values $C_k \leftarrow 2C_{k-1}, 4C_{k-1}, 8C_{k-1}, \ldots$, until (2.35) is satisfied. Since (2.35) will be satisfied whenever $C_k \geq C_{h,Q}$ from the definition of $C_{h,Q}$ in (2.7) and the inequality
$B_k - h(\lambda_k) \leq B_k^w - h(\lambda_k) = \nabla h(\lambda_k)^T (\tilde{\lambda}_k - \lambda_k)$, it follows that the doubling strategy will guarantee $C_k \leq \max\{C_0, 2C_{h,Q}\}$. Of course, if an upper bound $\bar{C} \geq C_{h,Q}$ is known, then $C_k \leftarrow \bar{C}$ is a valid assignment for all $k \geq 1$. Moreover, the structure of $h(\cdot)$ may be sufficiently simple so that a value of $C_k \geq C_{k-1}$ satisfying (2.35) can be determined analytically via closed-form calculation, as is the case if $h(\cdot)$ is a quadratic function for example. 

The formal description of the Frank-Wolfe method with dynamic step-size strategy is presented in Method 2.2.

**Method 2.2** Frank-Wolfe Method with Dynamic Step-sizes for maximizing $h(\lambda)$

Initialize at $\lambda_1 \in Q$, initial estimate $C_0$ of $C_{h,Q}$, (optional) initial upper bound $B_0$, $k \leftarrow 1$ .

At iteration $k$:
1. Compute $\nabla h(\lambda_k)$ .
2. Compute $\tilde{\lambda}_k \leftarrow \arg \max_{\lambda \in Q} \{h(\lambda_k) + \nabla h(\lambda_k)^T (\lambda - \lambda_k)\}$.

$$
B_k^w \leftarrow h(\lambda_k) + \nabla h(\lambda_k)^T (\tilde{\lambda}_k - \lambda_k).
$$

$$
G_k \leftarrow \nabla h(\lambda_k)^T (\tilde{\lambda}_k - \lambda_k).
$$

3. (Optional: compute other upper bound $B_k^o$), update best bound $B_k \leftarrow \min\{B_{k-1}, B_k^w, B_k^o\}$.
4. Compute $C_k$ for which the following conditions hold:
   (i) $C_k \geq C_{k-1}$ , and
   (ii) $h(\lambda_k + \tilde{\alpha}_k(\tilde{\lambda}_k - \lambda_k)) \geq h(\lambda_k) + \tilde{\alpha}_k (B_k - h(\lambda_k)) - \frac{1}{2}C_k \tilde{\alpha}_k^2$, where $\tilde{\alpha}_k := \frac{2}{2C_k B_k - h(\lambda_k) + k}$. 
5. Set $\lambda_{k+1} \leftarrow \lambda_k + \tilde{\alpha}_k(\tilde{\lambda}_k - \lambda_k)$ .

We have the following computational guarantees for the Frank-Wolfe method with dynamic step-sizes (Method 2.2):

**Bound 2.6.** The iterates of the Frank-Wolfe method with dynamic step-sizes (Method 2.2) satisfy the following for any $k \geq 1$:

$$
B_k - h(\lambda_k) \leq \min_{\ell \in \{1, \ldots, k\}} \left\{ \frac{2C_k}{2C_k B_k - h(\lambda_k) + k} \right\}.
$$

(2.36)

Furthermore, if the doubling strategy is used to update the estimates $\{C_k\}$ of $C_{h,Q}$, it holds that $C_k \leq \max\{C_0, 2C_{h,Q}\}$. 

44
Notice that (2.36) naturally generalizes the static warm-start bound (2.31) (or (2.32)) to this more general dynamic case. Consider, for simplicity, the case where $C_k = C_{h,Q}$ is the known curvature constant. In this case, (2.36) says that we may apply the bound (2.32) with any $\ell \in \{1, \ldots, k\}$ as the starting iteration. That is, the computational guarantee for the dynamic case is at least as good as the computational guarantee for the static warm-start step-size (2.30) initialized at any iteration $\ell \in \{1, \ldots, k\}$.

Proof of Bound 2.6: Let $i \geq 1$. For convenience define $A_i = \frac{2C_i}{B_i - h(\lambda_i)}$, and in this notation (2.34) is $\bar{\alpha}_i = \frac{2}{A_i + 2}$. Applying (ii) in step (4.) of Method 2.2 we have:

$$B_{i+1} - h(\lambda_{i+1}) \leq B_i - h(\lambda_i) - \bar{\alpha}_i(B_i - h(\lambda_i)) + \frac{1}{2}\bar{\alpha}_i^2C_i$$

$$\leq B_i - h(\lambda_i) - \bar{\alpha}_i(B_i - h(\lambda_i)) + \frac{1}{2}\bar{\alpha}_i^2C_i$$

$$= (B_i - h(\lambda_i))(1 - \bar{\alpha}_i) + \frac{1}{2}\bar{\alpha}_i^2C_i$$

$$= 2C_i \left( \frac{A_i}{A_i + 2} \right) + \frac{2C_i}{(A_i + 2)^2}$$

$$= 2C_i \left( \frac{A_i + 3}{(A_i + 2)^2} \right)$$

$$< \frac{2C_i}{A_i + 1},$$

where the last inequality follows from the fact that $(a + 2)^2 > a^2 + 4a + 3 = (a + 1)(a + 3)$ for $a \geq 0$. Therefore

$$A_{i+1} = \frac{2C_{i+1}}{B_{i+1} - h(\lambda_{i+1})} = \frac{C_{i+1}}{C_i} \left( \frac{2C_i}{B_{i+1} - h(\lambda_{i+1})} \right) > \frac{C_{i+1}}{C_i} (A_i + 1). \quad (2.37)$$

We now show by reverse induction that for any $\ell \in \{1, \ldots, k\}$ the following inequality is
true:

\[ A_k \geq \frac{C_k}{C_{\ell}} A_\ell + k - \ell \quad (2.38) \]

Clearly (2.38) holds for \( \ell = k \), so let us suppose (2.38) holds for some \( \ell + 1 \in \{2, \ldots, k\} \). Then

\[
A_k \geq \frac{C_k}{C_{\ell+1}} A_{\ell+1} + k - \ell - 1
\]

\[
> \frac{C_k}{C_{\ell+1}} \left( \frac{C_{\ell+1}}{C_\ell} A_\ell + 1 \right) + k - \ell - 1
\]

\[
\geq \frac{C_k}{C_\ell} A_\ell + k - \ell,
\]

where the first inequality is the induction hypothesis, the second inequality uses (2.37), and the third inequality uses the monotonicity of the \( \{C_k\} \) sequence. This proves (2.38). Now for any \( \ell \in \{1, \ldots, k\} \) we have from (2.38) that:

\[
B_k - h(\lambda_k) = \frac{2C_k}{A_k} \leq \frac{2C_k}{C_{\ell}} A_{\ell} + k - \ell = \frac{2C_k}{B_{\ell - h(\lambda_\ell)} + k - \ell},
\]

proving the result.

2.5 Analysis of the Frank-Wolfe Method with Inexact Gradient Computations and/or Subproblem Solutions

In this section we present and analyze extensions of the Frank-Wolfe method in the presence of inexact computation of gradients and/or subproblem solutions. We first consider the case when the linear optimization subproblem is solved approximately.
2.5.1 Frank-Wolfe Method with Inexact Linear Optimization Subproblem Solutions

Here we consider the case when the linear optimization subproblem is solved approximately, which arises especially in optimization over matrix variables. For example, consider instances of (2.1) where $Q$ is the spectrahedron of symmetric matrices, namely $Q = \{ \Lambda \in S^{n \times n} : \Lambda \succeq 0, I \cdot \Lambda = 1 \}$, where $S^{n \times n}$ is the space of symmetric matrices of order $n$, "$\succeq$" is the Löwner ordering thereon, and "$\cdot \cdot \cdot$" denotes the trace inner product. For these instances solving the linear optimization subproblem corresponds to computing the leading eigenvector of a symmetric matrix, whose solution when $n \gg 0$ is typically computed inexactly using iterative methods.

For $\delta \geq 0$ an (absolute) $\delta$-approximate solution to the linear optimization subproblem
\[
\max_{\lambda \in Q} \{ c^T \lambda \}
\]
is a vector $\tilde{\lambda} \in Q$ satisfying:
\[
c^T \tilde{\lambda} \geq \max_{\lambda \in Q} \{ c^T \lambda \} - \delta ,
\]
and we use the notation $\tilde{\lambda} \leftarrow \text{approx}(\delta)_{\lambda \in Q} \{ c^T \lambda \}$ to denote assigning to $\tilde{\lambda}$ any such $\delta$-approximate solution. The same additive linear optimization subproblem approximation model is considered in Dunn and Harshbarger [30] and Jaggi [54], and a multiplicative linear optimization subproblem approximation model is considered in Lacoste-Julien et al. [61]; a related approximation model is used in connection with a greedy coordinate descent method in Dudík et al. [27]. In Method [2.3] we present a version of the Frank-Wolfe algorithm that uses approximate linear optimization subproblem solutions. Note that Method [2.3] allows for the approximation quality $\delta = \delta_k$ to be a function of the iteration index $k$. Note also that the definition of the Wolfe upper bound $B^w_k$ and the FW gap $G_k$ in step (2.) are amended from the original Frank-Wolfe algorithm (Method [2.1]) by an additional term $\delta_k$. It follows from (2.39) that:
\[
B^w_k = h(\lambda_k) + \nabla h(\lambda_k)^T(\tilde{\lambda}_k - \lambda_k) + \delta_k \geq \max_{\lambda \in Q} \{ h(\lambda_k) + \nabla h(\lambda_k)^T(\lambda - \lambda_k) \} \geq h^* ,
\]
which shows that $B^w_k$ is a valid upper bound on $h^*$, with similar properties for $G_k$. The following two theorems extend Theorem 2.1 and Theorem 2.2 to the case of approximate subproblem solutions. Analogous to the case of exact subproblem solutions, these two theorems can easily be used to derive suitable bounds for specific step-sizes rules such as those in Sections 2.3 and 2.4.

**Method 2.3** Frank-Wolfe Method with Approximate Subproblem Solutions

Initialize at $\lambda_1 \in Q$, (optional) initial upper bound $B_0$, $k \leftarrow 1$.

At iteration $k$:
1. Compute $\nabla h(\lambda_k)$.
2. Compute $\tilde{\lambda}_k \leftarrow \text{approx}(\delta_k)\lambda \in Q\{h(\lambda_k) + \nabla h(\tilde{\lambda}_k)\} \cdot \\
B^w_k \leftarrow h(\lambda_k) + \nabla h(\tilde{\lambda}_k) - \lambda_k + \delta_k$. \\
$G_k \leftarrow \nabla h(\lambda_k)\tilde{\lambda}_k - \lambda_k + \delta_k$.
3. (Optional: compute other upper bound $B^o_k$), update best bound $B_k \leftarrow \min\{B_{k-1}, B^w_k, B^o_k\}$.
4. Set $\lambda_{k+1} \leftarrow \lambda_k + \bar{\alpha}_k(\tilde{\lambda}_k - \lambda_k)$, where $\bar{\alpha}_k \in [0, 1)$.

**Theorem 2.3**. Consider the iterate sequences of the Frank-Wolfe method with approximate subproblem solutions (Method 2.3) $\{\lambda_k\}$ and $\{\tilde{\lambda}_k\}$ and the sequence of upper bounds $\{B_k\}$ on $h^*$, using the step-size sequence $\{\bar{\alpha}_k\}$. For the auxiliary sequences $\{\alpha_k\}$ and $\{\beta_k\}$ given by (2.9), and for any $k \geq 0$, the following inequality holds:

$$B_k - h(\lambda_{k+1}) \leq \frac{B_k - h(\lambda_k)}{\beta_{k+1}} + \frac{1}{2}C h_{\cdot Q} \sum_{i=1}^{k} \frac{\alpha^2_i}{\beta_i} + \frac{\alpha^2_i}{\beta_{k+1}} + \sum_{i=1}^{k} \alpha_i \delta_i . \quad (2.40)$$

**Theorem 2.4**. Consider the iterate sequences of the Frank-Wolfe method with approximate subproblem solutions (Method 2.3) $\{\lambda_k\}$ and $\{\tilde{\lambda}_k\}$, the sequence of upper bounds $\{B_k\}$ on $h^*$, and the sequence of FW gaps $\{G_k\}$ from step (2.), using the step-size sequence $\{\bar{\alpha}_k\}$. For the auxiliary sequences $\{\alpha_k\}$ and $\{\beta_k\}$ given by (2.9), and for any $\ell \geq 0$ and $k \geq \ell + 1$, the
following inequality holds:

\[
\min_{i \in \{\ell+1, \ldots, k\}} G_i \leq \frac{1}{\sum_{i=\ell+1}^k \bar{\alpha}_i} \left[ B_\ell - h(\lambda_1) - \frac{1}{2} C_{h, Q} \sum_{i=1}^\ell \frac{\alpha_i^2}{\beta_{i+1}} + \frac{\sum_{i=1}^\ell \alpha_i \delta_i}{\sum_{i=\ell+1}^k \bar{\alpha}_i} \right] + \frac{1}{2} C_{h, Q} \sum_{i=\ell+1}^k \bar{\alpha}_i \delta_i + \frac{\sum_{i=\ell+1}^k \bar{\alpha}_i \delta_i}{\sum_{i=\ell+1}^k \bar{\alpha}_i} \,.
\] (2.41)

**Remark 2.1.** The pre-start step can also be generalized to the case of approximate solution of the linear optimization subproblem. Let \( \lambda_1 \) and \( B_0 \) be computed by the pre-start step with a \( \delta = \delta_0 \)-approximate subproblem solution. Then Proposition 2.1 generalizes to:

\[
B_0 - h(\lambda_1) \leq \frac{1}{2} C_{h, Q} + \delta_0 ,
\]

and hence if the pre-start step is used (2.40) implies that:

\[
B_k - h(\lambda_{k+1}) \leq \frac{1}{2} C_{h, Q} \sum_{i=0}^k \frac{\alpha_i^2}{\beta_{i+1}} + \frac{\sum_{i=0}^k \alpha_i \delta_i}{\beta_{k+1}} ,
\] (2.42)

where \( \alpha_0 := 1 \).

Let us now discuss implications of Theorems 2.3 and 2.4, and Remark 2.1. Observe that the bounds on the right-hand sides of (2.40) and (2.41) are composed of the exact terms which appear on the right-hand sides of (2.10) and (2.11), plus additional terms involving the solution accuracy sequence \( \delta_1, \ldots, \delta_k \). It follows from (2.14) that these latter terms are particular convex combinations of the \( \delta_i \) values and zero, and in (2.42) the last term is a convex combination of the \( \delta_i \) values, whereby they are trivially bounded above by \( \max\{\delta_1, \ldots, \delta_k\} \). When \( \delta_i := \delta \) is a constant, then this bound is simply \( \delta \), and we see that the errors due to the approximate computation of linear optimization subproblem solutions do not accumulate, independent of the choice of step-size sequence \( \{\bar{\alpha}_k\} \). In other words, Theorem 2.3 implies that if we are able to solve the linear optimization subproblems to
an accuracy of \( \delta \), then the Frank-Wolfe method can solve (2.1) to an accuracy of \( \delta \) plus a function of the step-size sequence \( \{\bar{\alpha}_k\} \), the latter of which can be made to go to zero at an appropriate rate depending on the choice of step-sizes. Similar observations hold for the terms depending on \( \delta_1, \ldots, \delta_k \) that appear on the right-hand side of (2.41).

Note that Jaggi [54] considers the case where \( \delta_i := \frac{1}{2} \delta \bar{\alpha}_i C_{h,Q} \) (for some fixed \( \delta \geq 0 \)) and \( \bar{\alpha}_i := \frac{2}{i+2} \) for \( i \geq 0 \) (or \( \bar{\alpha}_i \) is determined by a line-search), and shows that in this case Method 2.3 achieves \( O \left( \frac{1}{k} \right) \) convergence in terms of both the optimality gap and the FW gaps. These results can be recovered as a particular instantiation of Theorems 2.3 and 2.4 using similar logic as in the proof of Bound 2.1.

**Proof of Theorem 2.3** First recall the identities (2.13) and (2.14) for the dual averages sequences (2.9). Following the proof of Theorem 2.1, we then have for \( i \geq 1 \):

\[
\beta_{i+1} h(\lambda_{i+1}) \geq \beta_{i+1} \left[ h(\lambda_i) + \nabla h(\lambda_i)^T (\tilde{\lambda}_i - \lambda_i) \bar{\alpha}_i - \frac{1}{2} \bar{\alpha}_i^2 C_{h,Q} \right] \\
= \beta_i h(\lambda_i) + (\beta_{i+1} - \beta_i) h(\lambda_i) + \beta_{i+1} \bar{\alpha}_i \nabla h(\lambda_i)^T (\tilde{\lambda}_i - \lambda_i) - \frac{1}{2} \bar{\alpha}_i C_{h,Q} \\
= \beta_i h(\lambda_i) + \alpha_i \left[ h(\lambda_i) + \nabla h(\lambda_i)^T (\tilde{\lambda}_i - \lambda_i) \right] - \frac{1}{2} \frac{\alpha_i^2}{\beta_{i+1}} C_{h,Q} \\
= \beta_i h(\lambda_i) + \alpha_i B_i - \alpha_i \delta_i - \frac{1}{2} \frac{\alpha_i^2}{\beta_{i+1}} C_{h,Q} ,
\]

where the third equality above uses the definition of the Wolfe upper bound (2.2) in Method 2.3. The rest of the proof follows exactly as in the proof of Theorem 2.1.

**Proof of Theorem 2.4** For \( i \geq 1 \) we have:

\[
h(\lambda_{i+1}) \geq h(\lambda_i) + \nabla h(\lambda_i)^T (\tilde{\lambda}_i - \lambda_i) \bar{\alpha}_i - \frac{1}{2} \bar{\alpha}_i^2 C_{h,Q} \\
= h(\lambda_i) + \bar{\alpha}_i G_i - \bar{\alpha}_i \delta_i - \frac{1}{2} \bar{\alpha}_i^2 C_{h,Q} ,
\]

where the equality above follows from the definition of the FW gap in Method 2.3. Summing
the above over $i \in \{\ell + 1, \ldots, k\}$ and rearranging yields:

$$
\sum_{i=\ell+1}^{k} \bar{\alpha}_i G_i \leq h(\lambda_{k+1}) - h(\lambda_{\ell+1}) + \sum_{i=\ell+1}^{k} \frac{1}{2} \bar{\alpha}_i^2 C_{h,Q} + \sum_{i=\ell+1}^{k} \bar{\alpha}_i \delta_i .
$$

(2.43)

The rest of the proof follows by combining (2.43) with Theorem 2.3 and proceeding as in the proof of Theorem 2.2.

\[\square\]

2.5.2 Frank-Wolfe Method with Inexact Gradient Computations

We now consider a version of the Frank-Wolfe method where the exact gradient computation is replaced with the computation of an approximate gradient, as was explored in Section 3 of Jaggi [54]. We analyze two different models of approximate gradients and derive computational guarantees for each model. We first consider the $\delta$-oracle model of d’Aspremont [22], which was developed in the context of accelerated first-order methods. For $\delta \geq 0$, a $\delta$-oracle is a (possibly non-unique) mapping $g_\delta(\cdot) : Q \rightarrow E^*$ that satisfies:

$$
\| (\nabla h(\bar{\lambda}) - g_\delta(\bar{\lambda}))^T (\lambda - \bar{\lambda}) \| \leq \delta \quad \text{for all } \lambda, \bar{\lambda} \in Q .
$$

(2.44)

Note that the definition of the $\delta$-oracle does not consider inexact computation of function values. Depending on the choice of step-size sequence $\{\bar{\alpha}_k\}$, this assumption is acceptable as the Frank-Wolfe method may or may not need to compute function values. (The warm-start step-size rule (2.34) requires computing function values, as does the computation of the upper bounds $\{B_k^\nu\}$, in which case a definition analogous to (2.44) for function values can be utilized.)

The next proposition states the following: suppose one solves for the exact solution of the linear optimization subproblem using the $\delta$-oracle instead of the exact gradient. Then the absolute suboptimality of the computed solution in terms of the exact gradient is at most $2\delta$.

**Proposition 2.2.** For any $\bar{\lambda} \in Q$ and any $\delta \geq 0$, if $\bar{\lambda} \in \arg\max_{\lambda \in Q} \{g_\delta(\bar{\lambda})^T \lambda\}$, then $\bar{\lambda}$ is a
2\(\delta\)-approximate solution to the linear optimization subproblem \[
\max_{\lambda \in Q} \{ \nabla h(\bar{\lambda})^T \lambda \}.
\]

Proof. Let \(\hat{\lambda} \in \arg\max_{\lambda \in Q} \{ \nabla h(\bar{\lambda})^T \lambda \}\). Then we have:

\[
\nabla h(\bar{\lambda})^T (\tilde{\lambda} - \bar{\lambda}) \geq g_\delta(\bar{\lambda})^T (\tilde{\lambda} - \bar{\lambda}) - \delta
\geq g_\delta(\bar{\lambda})^T (\tilde{\lambda} - \bar{\lambda}) - \delta
\geq \nabla h(\bar{\lambda})^T (\tilde{\lambda} - \bar{\lambda}) - 2\delta
= \max_{\lambda \in Q} \{ \nabla h(\bar{\lambda})^T \lambda \} - \nabla h(\bar{\lambda})^T \bar{\lambda} - 2\delta,
\]

where the first and third inequalities use (2.44), the second inequality follows since \(\tilde{\lambda} \in \arg\max_{\lambda \in Q} \{ g_\delta(\bar{\lambda})^T \lambda \}\), and the final equality follows since \(\hat{\lambda} \in \arg\max_{\lambda \in Q} \{ \nabla h(\bar{\lambda})^T \lambda \}\). Rearranging terms then yields the result.

Now consider a version of the Frank-Wolfe method where the computation of \(\nabla h(\lambda_k)\) at step (1.) is replaced with the computation of \(g_\delta(\lambda_k)\). Then Proposition 2.2 implies that such a version can be viewed simply as a special case of the version of the Frank-Wolfe method with approximate subproblem solutions (Method 2.3) of Section 2.5.1 with \(\delta_k\) replaced by \(2\delta_k\). Thus, we may readily apply Theorems 2.3 and 2.4 and Proposition 2.1 to this case. In particular, similar to the results in [22] regarding error non-accumulation for an accelerated first-order method, the results herein imply that there is no accumulation of errors for a version of the Frank-Wolfe method that computes approximate gradients with a \(\delta\)-oracle at each iteration. Furthermore, it is a simple extension to consider a version of the Frank-Wolfe method that computes both (i) approximate gradients with a \(\delta\)-oracle, and (ii) approximate linear optimization subproblem solutions.

Inexact Gradient Computation Model via the \((\delta, L)\)-oracle

The premise (2.44) underlying the \(\delta\)-oracle is quite strong and can be restrictive in many cases. For this reason among others, Devolder et al. [25] introduce the less restrictive \((\delta, L)\)-
oracle model. For scalars $\delta, L \geq 0$, the $(\delta, L)$-oracle is defined as a (possibly non-unique) mapping $Q \to \mathbb{R} \times E^*$ that maps $\bar{\lambda} \to (h_{(\delta,L)}(\bar{\lambda}), g_{(\delta,L)}(\bar{\lambda}))$ which satisfy:

\begin{align}
  h(\lambda) &\leq h_{(\delta,L)}(\bar{\lambda}) + g_{(\delta,L)}(\bar{\lambda})^T(\lambda - \bar{\lambda}), \quad \text{and} \quad (2.45) \\
  h(\lambda) &\geq h_{(\delta,L)}(\bar{\lambda}) + g_{(\delta,L)}(\bar{\lambda})^T(\lambda - \bar{\lambda}) - \frac{L}{2}\|\lambda - \bar{\lambda}\|^2 - \delta \quad \text{for all } \lambda, \bar{\lambda} \in Q, \tag{2.46}
\end{align}

where $\| \cdot \|$ is a choice of norm on $E$. Note that in contrast to the $\delta$-oracle model, the $(\delta, L)$-oracle model does not assume that the function $h(\cdot)$ is smooth or even concave – it simply assumes that there is an oracle returning the pair $(h_{(\delta,L)}(\bar{\lambda}), g_{(\delta,L)}(\bar{\lambda}))$ satisfying (2.45) and (2.46).

In Method 2.4 we present a version of the Frank-Wolfe method that utilizes the $(\delta, L)$-oracle. Note that we allow the parameters $\delta$ and $L$ of the $(\delta, L)$-oracle to be a function of the iteration index $k$. Inequality (2.45) in the definition of the $(\delta, L)$-oracle immediately implies that $B_k^w \geq h^*$. We now state the main technical complexity bound for Method 2.4, in terms of the sequence of bound gaps $\{B_k - h(\lambda_{k+1})\}$. Recall from Section 2.2 the definition $\text{Diam}_Q := \max_{\lambda, \tilde{\lambda} \in Q}\{\|\lambda - \tilde{\lambda}\|\}$, where the norm $\| \cdot \|$ is the norm used in the definition of the $(\delta, L)$-oracle (2.46).

**Method 2.4 Frank-Wolfe Method With $(\delta, L)$-Oracle**

Initialize at $\lambda_1 \in Q$, (optional) initial upper bound $B_0$. $k \leftarrow 1$.

At iteration $k$:

1. Compute $h_k \leftarrow h_{(\delta_k, L_k)}(\lambda_k)$, $g_k \leftarrow g_{(\delta_k, L_k)}(\lambda_k)$.
2. Compute $\tilde{\lambda}_k \leftarrow \arg\max_{\lambda \in Q}\{h_k + g_k^T(\lambda - \lambda_k)\}$.

$$
B_k^w \leftarrow h_k + g_k^T(\tilde{\lambda}_k - \lambda_k).
$$

3. (Optional: compute other upper bound $B_k^o$), update best bound $B_k \leftarrow \min\{B_{k-1}^w, B_k^o\}$.
4. Set $\lambda_{k+1} \leftarrow \lambda_k + \bar{\alpha}_k(\tilde{\lambda}_k - \lambda_k)$, where $\bar{\alpha}_k \in [0, 1)$.

**Theorem 2.5.** Consider the iterate sequences of the Frank-Wolfe method with the $(\delta, L)$-oracle (Method 2.4) $\{\lambda_k\}$ and $\{\tilde{\lambda}_k\}$ and the sequence of upper bounds $\{B_k\}$ on $h^*$, using the step-size sequence $\{\tilde{\alpha}_k\}$. For the auxiliary sequences $\{\alpha_k\}$ and $\{\beta_k\}$ given by (2.9), and for
any $k \geq 0$, the following inequality holds:

$$B_k - h(\lambda_{k+1}) \leq \frac{B_k - h(\lambda_1)}{\beta_{k+1}} + \frac{1}{2} \text{Diam}_\mathcal{Q}^2 \sum_{i=1}^{k} L_{i} \frac{\alpha_i^2}{\beta_{i+1}} + \frac{1}{\beta_{k+1}} \sum_{i=1}^{k} \beta_{i+1} \delta_i .$$  \hfill (2.47)

As with Theorem 2.3, observe that the terms on the right-hand side of (2.47) are composed of the exact terms which appear on the right-hand side of (2.10), plus an additional term that is a function of $\delta_1, \ldots, \delta_k$. Unfortunately, Theorem 2.5 implies an accumulation of errors for Method 2.4 under essentially any choice of step-size sequence $\{\tilde{\alpha}_k\}$. Indeed, suppose that $\beta_i = O(i^\gamma)$ for some $\gamma \geq 0$, then $\sum_{i=1}^{k} \beta_{i+1} = O(k^{\gamma+1})$, and in the constant case where $\delta_i := \delta$, we have $\sum_{i=1}^{k} \beta_{i+1} \delta_i = O(k\delta)$. Therefore in order to achieve an $O\left(\frac{1}{k}\right)$ rate of convergence (for example with the step-size sequence (2.18)) we need $\delta = O\left(\frac{1}{k^2}\right)$. This negative result nevertheless contributes to the understanding of the merits and demerits of different first-order methods as follows. Note that in [25] it is shown that the “classical” gradient methods (both primal and dual), which require solving a proximal projection problem at each iteration, achieve an $O\left(\frac{1}{k} + \delta\right)$ accuracy under the $(\delta, L)$-oracle model for constant $(\delta, L)$. On the other hand, it is also shown in [25] that all accelerated first-order methods (which also solve proximal projection problems at each iteration) generically achieve an $O\left(\frac{1}{k^2} + k\delta\right)$ accuracy and thus suffer from an accumulation of errors under the $(\delta, L)$-oracle model. As discussed in the Introduction herein, the Frank-Wolfe method offers two possible advantages over these proximal methods: (i) the possibility that solving the linear optimization subproblem is easier than the projection-type problem in an iteration of a proximal method, and/or (ii) the possibility of greater structure (sparsity, low rank) of the iterates. In Table 2.1 we summarize the cogent properties of these three methods (or classes of methods) under exact gradient computation as well as with the $(\delta, L)$-oracle model. As can be seen from the table in Table 2.1 no single method dominates in the three categories of properties shown in the table; thus there are inherent tradeoffs among these methods/classes.
<table>
<thead>
<tr>
<th>Method/Class</th>
<th>Type of Subproblem</th>
<th>Accuracy with Exact Gradients</th>
<th>Accuracy with $(\delta, L)$-oracle</th>
<th>Special Structure of Iterates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank-Wolfe</td>
<td>Linear Optimization</td>
<td>$O(1/k)$</td>
<td>$O(1/k) + O(\delta k)$</td>
<td>Yes</td>
</tr>
<tr>
<td>Classical Gradient</td>
<td>Prox Projection</td>
<td>$O(1/k)$</td>
<td>$O(1/k) + O(\delta)$</td>
<td>No</td>
</tr>
<tr>
<td>Accelerated</td>
<td>Prox Projection</td>
<td>$O(1/k^2)$</td>
<td>$O(1/k^2) + O(\delta k)$</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.1: Properties of three (classes of) first-order methods after $k$ iterations.

**Proof of Theorem 2.5:** Note that (2.45) and (2.46) with $\bar{\lambda} = \lambda$ imply that:

$$h(\lambda) \leq h(\delta, L)(\lambda) \leq h(\lambda) + \delta \quad \text{for all } \lambda \in Q . \quad (2.48)$$

Recall properties (2.13) and (2.14) of the dual averages sequences (2.9). Following the proof of Theorem 2.1, we then have for $i \geq 1$:

$$\beta_{i+1} h(\lambda_{i+1}) \geq \beta_i h_i + g_i^T (\tilde{\lambda}_i - \lambda_i) \bar{\alpha}_i - \frac{1}{2} \bar{\alpha}_i^2 L_i \text{Diam}_Q^2 - \delta_i$$

$$= \beta_i h_i + (\beta_{i+1} - \beta_i) h_i + \beta_{i+1} \bar{\alpha}_i g_i^T (\tilde{\lambda}_i - \lambda_i) - \frac{1}{2} \beta_{i+1} \bar{\alpha}_i^2 L_i \text{Diam}_Q^2 - \beta_{i+1} \delta_i$$

$$= \beta_i h_i + \alpha_i \left[ h_i + g_i^T (\tilde{\lambda}_i - \lambda_i) \right] - \frac{1}{2} \frac{\alpha_i^2}{\beta_{i+1}} L_i \text{Diam}_Q^2 - \beta_{i+1} \delta_i$$

$$\geq \beta_i h(\lambda_i) + \alpha_i B_i^w - \frac{1}{2} \frac{\alpha_i^2}{\beta_{i+1}} L_i \text{Diam}_Q^2 - \beta_{i+1} \delta_i ,$$

where the first inequality uses (2.46), and the second inequality uses (2.48) and the definition of the Wolfe upper bound in Method 2.4. The rest of the proof follows as in the proof of Theorem 2.1. 

**2.6 Summary/Conclusions**

In this chapter we have developed new analysis and results for the Frank-Wolfe method. Virtually all of our results are consequences and applications of Theorems 2.1 and 2.2 which present computational guarantees for optimality gaps (Theorem 2.1) and the “FW gaps” (Theorem 2.2) for arbitrary step-size sequences \{\bar{\alpha}_k\} of the Frank-Wolfe method. These
technical theorems are applied to yield computational guarantees for the well-studied step-size rule $\bar{\alpha}_k := \frac{2}{k+2}$ (Section 2.3.1), simple averaging (Section 2.3.2), and constant step-size rules (Section 2.3.3). The second set of contributions in the chapter concern “warm start” step-size rules and computational guarantees that reflect the quality of the given initial iterate (Section 2.4) as well as the accumulated information about the optimality gap and the curvature constant over a sequence of iterations (Section 2.4.1). The third set of contributions concerns computational guarantees in the presence of an approximate solution of the linear optimization subproblem (Section 2.5.1) and approximate computation of gradients (Section 2.5.2).
Chapter 3

Dual Averaging as a Unifying Viewpoint of Greedy First-Order Methods

3.1 Introduction

In this chapter, we consider two greedy first-order methods, the greedy coordinate descent method and the Frank-Wolfe method. Greedy first-order methods are attractive for many large-scale/high-dimensional problems in statistics and machine learning applications due to the fact that the algorithms themselves are “structure-enhancing.” For some examples of and exposition regarding this important feature of these algorithms, see the introductions of Chapters 2 and 4 (regarding the Frank-Wolfe method) and Chapter 5 (regarding the greedy coordinate descent method) herein. In this chapter, we demonstrate that both algorithms are in fact specific instantiations of one “master” algorithm — the method of dual averaging — applied to specific auxiliary problems.

Notation  Let $e_j$ denote the $j^{th}$ unit vector in $\mathbb{R}^n$, and let $e = (1, \ldots, 1)$. Let $\| \cdot \|_q$ denote the $\ell_q$-norm for $q \in [1, \infty]$ with unit ball $B_q$, and let $\|v\|_0$ denote the number of non-zero coefficients of the vector $v$. For $A \in \mathbb{R}^{m \times n}$, let $\|A\|_{q_1, q_2} := \max_{x: \|x\|_{q_1} \leq 1} \|Ax\|_{q_2}$ be the operator
norm. For a scalar \( \alpha \), \( \text{sgn}(\alpha) \) denotes the sign of \( \alpha \), and \( \alpha^+ \), \( \alpha^- \) denote the positive and negative parts of \( \alpha \), respectively. For a given norm \( \| \cdot \| \) on \( \mathbb{R}^n \), \( \| \cdot \|_* \) denotes the dual norm defined by \( \| s \|_* = \max_{x : \| x \| \leq 1} s^T x \). Let \( \partial f(\cdot) \) denote the subdifferential operator of a convex function \( f(\cdot) \). The notation \( "\hat{v} \leftarrow \arg \max_{v \in S} f(v)" \) denotes assigning \( \hat{v} \) to be any optimal solution of the problem \( \max_{v \in S} \{ f(v) \} \). For a convex set \( P \) let \( \Pi_P(\cdot) \) denote the Euclidean projection operator onto \( P \), namely \( \Pi_P(\bar{x}) := \arg \min_{x \in P} \| x - \bar{x} \|^2 \).

Let us begin this chapter by considering the following two non-smooth convex optimization problems:

\[
\min_{x \in P} f(x) ,
\]

and

\[
\min_{x \in P} \phi(x) := f(x) + d(x) .
\]

Here \( P \) is a closed convex set in a finite dimensional Euclidean space \( E \), \( f(\cdot) : P \to \mathbb{R} \) is a (generally) nonsmooth and \( L_f \)-Lipschitz convex function with computable subgradients, and \( d(\cdot) : P \to \mathbb{R} \) is a \( \sigma \)-strongly convex function. We refer to \( d(\cdot) \) as the prox function, and we assume that the auxiliary problem \( \min_{x \in P} \{ \langle c, x \rangle + d(x) \} \) is efficiently solvable for all \( c \in E^* \). Let \( x^e := \arg \min_{x \in P} d(x) \) denote the prox center and assume (without loss of generality) that \( d(x^e) = 0 \).

Algorithm 3.1 is the dual averaging method of Nesterov [76], slightly generalized to allow for arbitrary starting values in both the primal and dual sequences. Since Nesterov’s paper [76], there has been a substantial amount of interest in the dual averaging method, due at least in part to its convenient primal-dual structure and the greater flexibility that arises from having two control sequences, \( \{ \alpha_k \} \) and \( \{ \beta_k \} \).

Until otherwise stated, assume for simplicity that \( s_0 = 0 \) and \( x_0 = x^e = \arg \min_{x \in P} d(x) \). Let us define \( A_k := \sum_{i=0}^{k-1} \alpha_i \) for \( k \geq 0 \). Observe that the dual averaging method is producing a
Algorithm 3.1 Dual Averaging Method

Initialize at $x_0 \in P$ and $s_0 \in E^*$, choose $\beta_0 > 0$, and set $k \leftarrow 0$.

At iteration $k$:
1. Compute $g_k \in \partial f(x_k)$.
2. Choose $\alpha_k \geq 0$ and set
   $$s_{k+1} \leftarrow s_k + \alpha_k g_k .$$
3. Choose $\beta_{k+1} \geq \beta_k$ and set
   $$x_{k+1} \leftarrow \arg \min_{x \in P} \{\langle s_{k+1}, x \rangle + \beta_{k+1} d(x) \} .$$

sequence of linear functions $\{ \ell_k(\cdot) \}$ defined by:

$$\ell_k(x) := \sum_{i=0}^{k-1} \alpha_i [f(x_i) + \langle g_i, x - x_i \rangle] \quad k \geq 0 ,$$

and note that $\ell_k(\cdot)$ lower bounds $A_k f(\cdot)$, i.e.,

$$\ell_k(x) \leq A_k f(x) \quad \text{for all } x \in P . \quad (3.3)$$

Define $\mu_k := \beta_k / A_k$ and note that we can write $x_k$ as:

$$x_k = \arg \min_{x \in P} \left\{ \frac{1}{\mu_k} \ell_k(x) + \mu_k d(x) \right\} .$$

Thus $x_k$ may be interpreted as the solution to a linear model of the objective $f(\cdot)$ plus a regularization term intended to induce stability in the iterate sequence $\{x_k\}$. Herein, we consider two regimes for the values of $\mu_k$:

- $\mu_k \to 0$: in this case, the importance of the prox function is vanishing as $k \to \infty$ and we expect the iterates $\{x_k\}$ to approach a solution of (3.1). This is the regime primarily considered in [76] (wherein a host of computational guarantees are derived for problem (3.1) and several variants of (3.1)), and it turns out the interpretation of greedy coordinate descent developed herein falls under this regime as well.
• $\mu_k = 1$ for all $k \geq 1$: in this case, the importance of the prox function is kept constant, and we expect the iterates $\{x_k\}$ to approach a solution of (3.2). Note also that $\mu_k = 1$ means that $\beta_k = \sum_{i=0}^{k-1} \alpha_i$. It turns out that the interpretation of the Frank-Wolfe method as dual averaging corresponds to this regime. As far as we know, this regime of $\{\alpha_k\}$ and $\{\beta_k\}$ values has not been considered directly in the dual averaging literature before. However, sequences with the same structure have appeared in the analysis of the Frank-Wolfe method and its variants, see [78] and Chapter 2 herein.

3.2 Greedy Coordinate Descent and Dual Averaging

Consider the following unconstrained convex optimization problem:

$$F^* := \min_{\lambda \in \mathbb{R}^n} F(\lambda), \quad (3.4)$$

where $F(\cdot): \mathbb{R}^n \rightarrow \mathbb{R}$ is a smooth convex function. That is, we assume that the gradient of $F(\cdot)$ is Lipschitz continuous with respect to the $\ell_1$-norm:

$$\|\nabla F(\lambda) - \nabla F(\lambda')\|_\infty \leq L_F \|\lambda - \lambda'\|_1 \quad \text{for all } \lambda, \lambda' \in \mathbb{R}^n. \quad (3.5)$$

The greedy coordinate descent method is part of the general class of coordinate descent methods for solving (3.4), which have the defining property of updating one coordinate at each iteration. This property is particularly appealing in the high-dimensional setting ($n \gg 0$) since updating one coordinate at a time is particularly simple and amenable to sparse representations of the variables. The greedy coordinate descent method determines the coordinate to be updated in a “greedy” fashion by selecting the coordinate $j$ that has the largest magnitude in the gradient. The determination of the coordinate direction is then followed by a step of length $\alpha$ in either the positive or negative direction of this coordinate, depending on the sign of the coordinate of the gradient. The formal statement of the greedy coordinate descent method to solve (3.4) is presented in Algorithm 3.2.
Algorithm 3.2 Greedy Coordinate Descent Method

Initialize at $\lambda^0 \in \mathbb{R}^n, k = 0$

At iteration $k$:

1. Compute:
   \[ j_k \in \arg \max_{j \in \{1, \ldots, n\}} |\nabla F(\lambda^k)_j| \]

2. Choose $\alpha_k \geq 0$ and set:
   \[ \lambda_{jk}^{k+1} \leftarrow \lambda_{jk}^k - \alpha_k \text{sgn}(\nabla F(\lambda^k)_{jk}) \]
   \[ \lambda_j^{k+1} \leftarrow \lambda_j^k \text{ for } j \neq j_k \]

There is no general computational guarantee associated with the greedy coordinate descent method without additional assumptions such as bounded optimal solutions, strong convexity of the function $F(\cdot)$, and/or maximum distances of the starting points and/or the iterates from the set of optimal solutions (or the set of near-optimal solutions). Other variants of coordinate descent methods use a non-greedy strategy in the choice of the coordinate to be updated, such as randomized ordering or cyclic ordering of coordinates. We refer interested readers to Nesterov [77], Richtárik and Takáč [85], and Beck and Tetruashvili [6] for recent work on coordinate descent methods.

We now present a set of computational guarantees for the greedy coordinate descent method. Let $\mathcal{S}_0$ denote the set of all $\lambda$ whose objective value is at most $F(\lambda^0)$, namely $\mathcal{S}_0 := \{ \lambda \in \mathbb{R}^n : F(\lambda) \leq F(\lambda^0) \}$, and let $\mathcal{S}^*$ denote the set of optimal solutions of (3.4), i.e., $\mathcal{S}^* := \{ \lambda \in \mathbb{R}^n : F(\lambda) = F^* \}$. Let $\text{Dist}_0$ denote the largest distance of points in $\mathcal{S}_0$ to the set of optimal solutions $\mathcal{S}^*$, whose formal definition is:

\[ \text{Dist}_0 := \max_{\lambda \in \mathcal{S}_0} \min_{\lambda^* \in \mathcal{S}^*} \| \lambda - \lambda^* \|_1. \quad (3.6) \]

The following computational guarantee for greedy coordinate descent is an amended and extended version of results in Beck and Tetruashvili [6].
Theorem 3.1. (Computational Guarantees for Greedy Coordinate Descent) Let \( \{\lambda^k\} \) be generated according to the Greedy Coordinate Descent method (Algorithm 3.2) using the step-size sequence \( \{\alpha_k\} \). Recall that \( L_F \) denotes the Lipschitz constant of the gradient of \( F(\cdot) \) with respect to the \( \ell_1 \) norm. Then for all \( k \geq 0 \) it holds that:

\[
\min_{i \in \{0, \ldots, k\}} \| \nabla F(\lambda^i) \|_\infty \leq \frac{F(x^0) - F^* + \frac{L_F}{2} \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}. \tag{3.7}
\]

If the step-sizes are chosen using the rule:

\[
\alpha_k = \frac{\| \nabla F(\lambda^k) \|_\infty}{L_F} \quad \text{for all } k \geq 0 , \tag{3.8}
\]

then for each \( k \geq 0 \), the following inequality holds:

\[
\min_{i \in \{0, \ldots, k\}} \| \nabla F(\lambda^i) \|_\infty \leq \sqrt{\frac{2L_F(F(\lambda^0) - F^*)}{k + 1}} , \tag{3.9}
\]

and if \( \text{Dist}_0 \) is finite, then also:

\[
F(\lambda^k) - F^* \leq \frac{1}{F(\lambda^0) - F^* + \frac{L_F}{2} (\text{Dist}_0)^2} \leq \frac{2L_F(\text{Dist}_0)^2}{k} . \tag{3.10}
\]

Proof. Since \( F(\cdot) \) satisfies (3.5), it follows easily from the fundamental theorem of calculus (also Proposition [A.2] in Appendix A) that:

\[
F(\hat{\lambda}) \leq F(\lambda) + \nabla F(\lambda)^T (\hat{\lambda} - \lambda) + \frac{L_F}{2} \| \hat{\lambda} - \lambda \|_1^2 \quad \text{for all } \lambda, \hat{\lambda} . \tag{3.11}
\]

Applying (3.11) to the iterates of the Greedy Coordinate Descent method yields the following
for each $i \geq 0$:

\[
F(\lambda^{i+1}) \leq F(\lambda^i) + \nabla F(\lambda^i)^T (\lambda^{i+1} - \lambda^i) + \frac{L_F}{2} \|\lambda^{i+1} - \lambda^i\|_1
\]

\[
= F(\lambda^i) - \alpha_i |\nabla F(\lambda^i)_{j|} + \frac{L_F}{2} \alpha_i^2
\]

\[
= F(\lambda^i) - \alpha_i |\nabla F(\lambda^i)||_\infty + \frac{L_F}{2} \alpha_i^2 . \tag{3.12}
\]

Summing the above for $i = 0, \ldots, k$ yields:

\[
F^* \leq F(\lambda^{k+1}) \leq F(\lambda^0) - \sum_{i=0}^{k} \alpha_i |\nabla F(\lambda^i)||_\infty + \frac{L_F}{2} \sum_{i=0}^{k} \alpha_i^2 . \tag{3.13}
\]

Next notice that

\[
\sum_{i=0}^{k} \alpha_i |\nabla F(\lambda^i)||_\infty \geq \left(\sum_{i=0}^{k} \alpha_i\right) \left(\min_{i \in \{0, \ldots, k\}} |\nabla F(\lambda^i)||_\infty\right) ,
\]

and substituting this inequality above and rearranging yields (3.7).

Now suppose we use the step-sizes (3.8). Substituting (3.8) in (3.12) yields:

\[
F(\lambda^{i+1}) \leq F(\lambda^i) - \frac{1}{2L_F} |\nabla F(\lambda^i)||_\infty^2 , \tag{3.14}
\]

which shows in particular that the values $F(\lambda^i)$ are decreasing and in particular $F(\lambda^i) \leq F(\lambda^0)$. Substituting the step-sizes (3.8) in (3.13) yields after rearranging:

\[
\sum_{i=0}^{k} |\nabla F(\lambda^i)||_\infty^2 \leq 2L_F (F(\lambda^0) - F(\lambda^{k+1})) \leq 2L_F (F(\lambda^0) - F^*) .
\]

Therefore:

\[
(k + 1) \left(\min_{i \in \{0, \ldots, k\}} |\nabla F(\lambda^i)||_\infty\right)^2 \leq \sum_{i=0}^{k} |\nabla F(\lambda^i)||_\infty^2 \leq 2L_F (F(\lambda^0) - F^*) ,
\]
and rearranging yields \( (3.9) \). Now suppose as well that \( \text{Dist}_0 \) is finite, and let \( \lambda^i \) be an iterate of the greedy coordinate descent method. Then from the monotonicity property of the method mentioned earlier in the proof, we have \( F(\lambda^i) \leq F(\lambda^0) \) and therefore \( \lambda^i \in S_0 \). Therefore there exists \( \lambda^* \in S^* \) for which \( \|\lambda^i - \lambda^*\|_1 \leq \text{Dist}_0 \), and from the gradient inequality for the convex function \( F(\cdot) \) it holds that

\[
F^* = F(\lambda^*) \geq F(\lambda^i) + \nabla F(\lambda^i)^T (\lambda^* - \lambda^i)
\]

\[
\geq F(\lambda^i) - \|\nabla F(\lambda^i)\|_{\infty} \|\lambda^* - \lambda^i\|_1
\]

\[
\geq F(\lambda^i) - \|\nabla F(\lambda^i)\|_{\infty} \text{Dist}_0,
\]

and rearranging the above yields \( \|\nabla F(\lambda^i)\|_{\infty} \geq \frac{F(\lambda^i) - F^*}{\text{Dist}_0} \). Substituting this inequality into \( (3.14) \) and subtracting \( F^* \) from both sides yields:

\[
F(\lambda^{i+1}) - F^* \leq F(\lambda^i) - F^* - \frac{(F(\lambda^i) - F^*)^2}{2L_F \text{Dist}_0^2}.
\]

Define \( a_i := F(\lambda^i) - F^* \), and it follows that the nonnegative series \( \{a_i\} \) satisfies \( a_{i+1} \leq a_i - \frac{a_i^2}{2L_F \text{Dist}_0^2} \). A standard induction argument (see for example Lemma 3.5 of [6]) then establishes that

\[
a_k \leq \frac{1}{a_0} + \frac{k}{2L_F \text{Dist}_0^2},
\]

which is precisely \( (3.10) \).

\[\square\]

**Remark 3.1.** If an exact line-search is used instead of the step-size rule \( (3.8) \), then bounds \( (3.9) \) and \( (3.10) \) remain valid. This follows easily from the structure of the above proof.

It also should be noted that while the guarantees \( (3.7) \) and \( (3.8) \) follow directly without building up any new machinery, they are entirely motivated by the equivalence that we are about to establish.
3.2.1 Greedy Coordinate Descent as Dual Averaging/Mirror Descent

In order to establish that the greedy coordinate descent method is a specific instance of the dual averaging method, let us assume that the objective function is given in a particular composite form, namely
\[ F(\lambda) := G(-A\lambda) \]
for all \( \lambda \in \mathbb{R}^{m \times n} \) is a given linear operator and \( G(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R} \) is a differentiable convex function. Thus, problem (3.4) can be written as:
\[ F^* := \min_{\lambda \in \mathbb{R}^n} F(\lambda) := G(-A\lambda) . \quad (3.15) \]

We further assume that \( G(\cdot) \) has Lipschitz continuous gradient:
\[ \|\nabla G(y) - \nabla G(y')\|_{\sharp*} \leq L_G \|y - y'\|_{\sharp*} \quad \text{for all } y, y' \in \mathbb{R}^m , \quad (3.16) \]
where \( \| \cdot \|_{\sharp} \) is a fixed norm in \( \mathbb{R}^m \) with dual norm \( \| \cdot \|_{\sharp*} \). Note that (3.16) immediately implies that \( F(\cdot) \) satisfies (3.5) with \( L_F \leq L_G \|A\|_{1,\sharp*} \), i.e., it holds that:
\[ \|\nabla F(\lambda) - \nabla F(\lambda')\|_{\infty} \leq L_G \|A\|_{1,\sharp*} \|\lambda - \lambda'\|_{1} \quad \text{for all } \lambda, \lambda' \in \mathbb{R}^n . \]

Herein, we demonstrate that Algorithm 3.2 may be viewed as a specific instantiation of dual averaging. To demonstrate this, we need to specify five objects for dual averaging: (i) the prox function \( d(\cdot) \), (ii) the feasible region \( P \), (iii) the non-smooth function \( f(\cdot) \), (iv) the sequence \( \{\alpha_k\} \), and (v) the sequence \( \{\beta_k\} \). Towards this end, consider the convex conjugate of \( G(\cdot) \), defined by:
\[ G^*(x) := \max_{y \in \mathbb{R}^m} \{x^Ty - G(y)\} . \quad (3.17) \]

Now we specify the five dual averaging objects as follows.

1. Let \( d(\cdot) \) be \( G^*(\cdot) \).

2. Let \( P \) be the effective domain of \( d(\cdot) \), \( P := \text{dom}(d(\cdot)) = \{x : d(x) < \infty\} \).
3. Let \( f(\cdot) \) be defined by \( f(x) := \|A^T x\|_\infty \).

4. Let \( \{\alpha_k\} \) be the same as the \( \{\alpha_k\} \) sequence in Algorithm 3.2.

5. Let \( \{\beta_k\} \) be defined by \( \beta_k := 1 \) for all \( k \geq 0 \).

Let us make a few remarks.

1. The strong convexity of \( d(\cdot) \) follows from the well known duality result between smoothness and strong convexity – namely a convex function is globally smooth if and only if its conjugate is strongly convex \cite{101}. It follows that \( d(\cdot) \) has strong convexity constant \( \sigma := \frac{1}{L_G} \).

2. Note that (3.17) implies that:

\[
G(y) = \max_{x \in P} \{y^T x - d(x)\} \quad \text{for all } y \in \mathbb{R}^m,
\]

and therefore

\[
F(\lambda) = G(-A\lambda) = \max_{x \in P} \{-x^T A\lambda - d(x)\} \quad \text{for all } \lambda \in \mathbb{R}^n. \tag{3.18}
\]

3. Given (3.18), Danskin’s Theorem gives the following rule for computing gradients:

\[
\bar{x} = \arg \max_{x \in P} \{-x^T A\lambda - d(x)\} \implies \nabla F(\lambda) = -A^T \bar{x}. \tag{3.19}
\]

Note that \( \bar{x} \) defined above always exists for any \( \lambda \in \mathbb{R}^n \), thus for any \( \lambda \in \mathbb{R}^n \), we have \( \nabla F(\lambda) = -A^T \bar{x} \) where \( \bar{x} = \arg \max_{x \in P} \{-x^T A\lambda - d(x)\} \). That is, whenever we compute a gradient \( \nabla F(\lambda) \) we are always at least implicitly solving the subproblem \( \arg \max_{x \in P} \{-x^T A\lambda - d(x)\} \) even though we may not be explicitly doing so.

Algorithm 3.2 can now be re-written equivalently as follows.
Algorithm 3.3 Greedy Coordinate Descent Method in Conjugate Form

Initialize at $\lambda^0 \in \mathbb{R}^n, k = 0$

At iteration $k$:
1. Compute:
   
   \[
   x^k = \arg\max_{x \in P} \{-x^T A \lambda^k - d(x)\}
   \]

   \[
   j_k \in \arg\max_{j \in \{1, \ldots, n\}} |(A^T x^k)_j|
   \]

2. Choose $\alpha_k \geq 0$ and set:
   \[
   \lambda^k_{j_k} \leftarrow \lambda^k_{j_k} + \alpha_k \text{sgn}((A^T x^k)_{j_k})
   \]

   \[
   \lambda^k_j \leftarrow \lambda^k_j \quad \text{for} \quad j \neq j_k
   \]

Greedy Coordinate Descent is Dual Averaging. We now show that greedy coordinate descent is an instance of dual averaging. To do so we need to define some auxiliary variables for Algorithm 3.3:

\[
\tilde{\lambda}^k = \text{sgn}((A^T x^k)_{j_k}) e_{j_k}, \quad g^k = A \tilde{\lambda}^k,
\]

and we need the following initial conditions for the dual averaging method:

\[
s^0 = A \lambda^0, \quad x^0 = \arg\min_{x \in P} \{x^T A \lambda^0 + d(x)\}.
\]

Furthermore, let $\{s^k\}$ be the sequence with $s^{k+1} = s^k + \alpha_k g^k$ for all $k \geq 0$. We are now ready to give the equivalence result.

Theorem 3.2. The greedy coordinate descent method (Algorithms 3.2, 3.3) is an instance of the dual averaging method with prox function $d(\cdot)$ applied to solve (3.2) initialized at (3.21). That is, the auxiliary sequences (3.20) satisfy:

1. $g^k \in \partial f(x^k)$ for all $k \geq 0$

2. $x^k = \arg\min_{x \in P} \{(s^k)^T x + d(x)\}$ for all $k \geq 0$.  

67
Proof. Observe that
\[ f(x) = \| A^T x \|_\infty = \max_{\lambda \in B_1} \{ x^T A \lambda \}, \]
and therefore
\[ j_k \in \arg\max_{j \in \{1, \ldots, n\}} |(A^T x_k)_j| \Rightarrow \tilde{\lambda}^k \in \arg\max_{\lambda \in B_1} \{ x^T A \lambda \} \Rightarrow g^k \in \partial f(x^k). \]

Now observe that step (2.) of Algorithm 3.3 is equivalent to the update \( \lambda^{k+1} \leftarrow \lambda^k + \alpha_k \tilde{\lambda}^k \), hence we have:
\[
 s^k = A\lambda^0 + \sum_{i=0}^{k-1} \alpha_i A\tilde{\lambda}^i = A \left( \lambda^0 + \sum_{i=0}^{k-1} \alpha_i \tilde{\lambda}^i \right) = A\lambda^k.
\]
And therefore:
\[
 x^k = \arg\max_{x \in P} \{ -x^T A\lambda^k - d(x) \}
 = \arg\min_{x \in P} \{ x^T A\lambda^k + d(x) \}
 = \arg\min_{x \in P} \{ (s^k)^T x + d(x) \}.
\]

We now make a few more remarks.

1. It can be shown that dual averaging (with \( \beta_k := 1 \) as here) and the method of mirror descent [72, 5] (also, see Section 6.1 of Chapter 6 herein for a brief review) are exactly equivalent under the condition that the iterates satisfy \( x^k \in \text{relint}(P) \) for all \( k \geq 0 \), which is satisfied for example when \( P = \mathbb{R}^m \) or when \( d(\cdot) \) is an entropy like function (i.e. \( d(\cdot) \) is essentially smooth).

2. The mirror descent specification of the \( \{\alpha_k\} \) and \( \{\beta_k\} \) sequences implies that we are in the regime where \( \mu_k \to 0 \), hence the relevant problem for dual averaging is (3.1), which in this instance is:
\[
 \min_{x \in P} \| A^T x \|_\infty.
\]
Consequently, the results in [76] imply computational guarantees for the above problem – these guarantees are essentially the same as (3.7). Note that the above problem is not the dual of (3.4); rather, it is an auxiliary problem that is fundamentally tied to the greedy coordinate descent algorithm.

3. This result can easily be generalized to steepest descent with other norms, i.e., we may replace \( \| \cdot \|_1 \) with an arbitrary norm in step (1.) of Algorithm 3.2 and replace \( \| \cdot \|_\infty \) in the definition of \( f(\cdot) \) with the dual norm.

4. Recall that \( \{ \nabla F(\lambda) : \lambda \in \mathbb{R}^n \} \subseteq \{ -A^T x : x \in P \} \). Thus Theorem 3.2 is essentially saying that greedy coordinate descent is working towards minimizing \( \| \nabla F(\lambda) \|_\infty \). This is consistent with the computational guarantees in Theorem 3.1.

3.3 Frank-Wolfe as a Special Case of Dual Averaging

The second part of this chapter is concerned with the Frank-Wolfe method. Here, we are interested in the following problem:

\[
\max_{\lambda \in Q} h(\lambda) := g(A\lambda) ,
\]  

(3.22)

where \( Q \subseteq \mathbb{R}^n \) is convex and compact, \( A \in \mathbb{R}^{m \times n} \) and \( g(\cdot) : Q \to \mathbb{R} \) is concave and differentiable. As in the previous section, we assume that \( g(\cdot) \) is \( L_g \)-globally smooth, which also implies that \( h(\cdot) \) is smooth as well. It can easily be shown that \( h(\cdot) \) admits the following representation:

\[
h(\lambda) = \min_{x \in P} \{ x^T A\lambda + d(x) \} \text{ for all } \lambda \in Q ,
\]  

(3.23)

where \( d(\cdot) \) is the conjugate of \( g(\cdot) \) and is therefore \( \sigma \)-strongly convex on \( P \) (the effective domain of \( d(\cdot) \)) with strong convexity constant \( \sigma := \frac{1}{L_g} \). Given the representation (3.23), we
can write down a dual of (3.22) as:

$$\min_{x \in P} \max_{\lambda \in Q} \{x^T A\lambda + d(x)\} \equiv \min_{x \in P} f(x) + d(x) \quad (3.24)$$

where \(f(\cdot) : P \rightarrow \mathbb{R}\) is defined by \(f(x) := \max_{\lambda \in Q} \{x^T A\lambda\}\). Notice that (3.24) is precisely in the format of (3.2).

We consider the same basic Frank-Wolfe method as in Chapter 2, i.e., Algorithm 2.1. Let us write down what the Frank-Wolfe method looks like using (3.23) as a basic object for performing computations. Suppose that we are at the \(k\)th iteration of the Frank-Wolfe method. Then our current iterate is \(\lambda_k\), and, by assumption, we can at least implicitly compute a point \(x_k \in P\) such that \(x_k \in \arg\min_{x \in P} \{x^T A\lambda_k + d(x)\}\) and \(\nabla h(\lambda_k) = A^T x_k\). Therefore, step (2.) of the Frank-Wolfe method (Algorithm 2.1) is computing:

\[
\tilde{\lambda}_k \in \arg\max_{\lambda \in Q} \{h(\lambda_k) + \nabla h(\lambda_k)^T (\lambda - \lambda_k)\} = \arg\max_{\lambda \in Q} \{x_k^T A\lambda\} ,
\]

Thus, step (2.) of the Frank-Wolfe method is simply computing the function value \(f(x_k)\) and the subgradient \(A\tilde{\lambda}_k \in \partial f(x_k)\). Using the above observations about the Frank-Wolfe iterations, we can rewrite the Frank-Wolfe method and optional pre-start procedure in “minmax form” below.

**Algorithm 3.4 Frank-Wolfe Method in Max Form**

Initialize at \(\lambda_1 \in Q\), (optional) initial upper bound \(B_0\), \(k \leftarrow 1\).

At iteration \(k\):

1. Compute \(\nabla h(\lambda_k) = A^T x_k\), \(x_k \in \arg\min_{x \in P} \{x^T A\lambda_k + d(x)\}\).
2. Compute \(\tilde{\lambda}_k \leftarrow \arg\max_{\lambda \in Q} \{x_k^T A\lambda\}\).
3. (Optional: Compute upper bound \(B_k\) at iteration \(k\).)
4. Set \(\lambda_{k+1} \leftarrow \lambda_k + \bar{\alpha}_k (\tilde{\lambda}_k - \lambda_k)\), where \(\bar{\alpha}_k \in [0, 1]\).

Now we are almost ready to show that the Frank-Wolfe method to maximize \(h(\cdot)\) is actually an instance of the generalized dual averaging method (Algorithm 3.1), for a particular initialization \(x_0\) and \(s_0\) of the dual averaging method. The final step involves “unwinding”
Procedure 3.5 Pre-start Step of Frank-Wolfe method given \( \lambda_0 \in Q \)

1. Compute \( \nabla h(\lambda_0) = A^T x_0 , \) \( x_0 \in \arg \min_{x \in P} \{ x^T A\lambda_0 + d(x) \} . \)
2. Compute \( \tilde{\lambda}_0 \leftarrow \arg \max_{\lambda \in Q} \{ x_0^T A\lambda \} . \)
3. (Optional: Compute upper bound \( B_0 \).)
4. Set \( \lambda_1 \leftarrow \tilde{\lambda}_0 . \)

step (4.) of the Frank-Wolfe method, which involves taking a convex combination, into the “additive form” of the dual averaging method. To accomplish this, we define the following sequence of \( \{ \alpha_k \} \) and \( \{ \beta_k \} \) values constructed simply as a function of the \( \{ \bar{\alpha}_k \} \) step-size sequence values from the Frank-Wolfe method:

\[
\beta_k = \frac{1}{\prod_{j=1}^{k-1} (1 - \bar{\alpha}_j)} , \quad \alpha_k = \frac{\beta_k \bar{\alpha}_k}{1 - \bar{\alpha}_k} , \quad k \geq 1 .
\]

(3.25)

(Here and in what follows we use the conventions: \( \prod_{j=1}^{0} = 1 \) and \( \sum_{i=1}^{0} = 0 \).) We define the following initial values:

\[
\beta_0 = 1 , \quad s_0 := A\lambda_1 , \quad \alpha_0 = 0 , \quad x_0 \text{ arbitrary}, \quad \tilde{\lambda}_0 := \arg \max_{\lambda \in Q} \{ x_0^T A\lambda \} . \quad (3.26)
\]

We will also be interested in the following modification of (3.26) that will be used to analyze Frank-Wolfe with a pre-start step (Procedure 3.5):

\[
\beta_0 = 1 , \quad s_0 := 0 , \quad \alpha_0 = 1 , \quad x_0 := \arg \min_{x \in P} \{ x^T A\lambda_0 + d(x) \} . \quad (3.27)
\]

We have the following equivalence result.

**Theorem 3.3.** Let \( \{ \lambda_k \} , \{ x_k \} , \) and \( \{ \tilde{\lambda}_k \} , \) \( k \geq 1 \), be the iterate sequences of the Frank-Wolfe Method written in minmax form (Algorithm 3.4) for solving the problem (3.22). Consider the sequences \( \{ \alpha_k \} \) and \( \{ \beta_k \} \) given in (3.25) and consider the initialization (3.26) if there is no pre-start step or (3.27) if the pre-start step (Procedure 3.5) is used. For each \( k \geq 0 \),
define
\[ g_k := A\tilde{\lambda}_k \text{ and } s_{k+1} \leftarrow s_k + \alpha_k g_k = s_0 + \sum_{i=0}^{k} \alpha_i g_i. \]

Then the Frank-Wolfe iterates correspond to iterates of the dual averaging method. In particular,

1. \( g_k \in \partial f(x_k) \) for all \( k \geq 0 \).
2. \( x_k \in \arg \min_{x \in P} \{s_k^T x + \beta_k d(x)\} \) for all \( k \geq 1 \).

Proof. We first do some arithmetic. Regardless of whether a pre-start step is used in the Frank-Wolfe method, it follows from Step (4.) of Algorithm 3.4 that for all \( k \geq 0 \), \( \lambda_{k+1} \) is a convex combination of \( \lambda_1 \) and \( \tilde{\lambda}_1, \ldots, \tilde{\lambda}_k \). Indeed, cumulatively applying the formula in Step (4.) of Algorithm 3.4 yields:

\[ \lambda_{k+1} = \left[ \prod_{j=1}^{k} (1 - \bar{\alpha}_j) \right] \lambda_1 + \sum_{i=1}^{k} \left[ \prod_{j=i+1}^{k} (1 - \bar{\alpha}_j) \right] \bar{\alpha}_i \tilde{\lambda}_i, \text{ for } k \geq 0. \]

Using the values of \( \{\beta_k\} \) given in (3.25), we can re-write the above as:

\[ \lambda_{k+1} = \frac{1}{\beta_{k+1}} \lambda_1 + \sum_{i=1}^{k} \frac{\beta_i}{\beta_{k+1}} \frac{\bar{\alpha}_i}{1 - \bar{\alpha}_i} \tilde{\lambda}_i, \text{ for } k \geq 0, \]

and then using the values of \( \{\alpha_k\} \) in (3.25) yields

\[ \lambda_{k+1} = \frac{1}{\beta_{k+1}} \lambda_1 + \sum_{i=1}^{k} \alpha_i \tilde{\lambda}_i \text{ for } k \geq 0. \tag{3.28} \]

It is also easy to establish that

\[ \beta_{k+1} = 1 + \sum_{i=1}^{k} \alpha_i \text{ for } k \geq 0. \tag{3.29} \]

To validate (3.29), note that it is true for \( k = 0 \) since in particular \( \beta_1 = 1 \). Assuming (3.29)
holds for some $k$, then arithmetic substitution establishes:

$$1 + \sum_{i=1}^{k+1} \alpha_i = \beta_{k+1} + \alpha_{k+1} = \beta_{k+1} + \frac{\beta_{k+1} \bar{\alpha}_{k+1}}{1 - \bar{\alpha}_{k+1}} = \frac{\beta_{k+1}}{1 - \bar{\alpha}_{k+1}} = \beta_{k+2},$$

completing the inductive step.

We now prove (1.). Consider any $x \in P$. Since $f(x) = \max_{\lambda \in Q} \{x^T A \lambda\}$, it follows easily that if $\tilde{\lambda} \in \arg \max_{\lambda \in Q} \{x^T A \lambda\}$, then $g := A \tilde{\lambda} \in \partial f(x)$. Therefore from Step (2.) of Algorithm 3.4 it holds that $g_k := A \tilde{\lambda}_k \in \partial f(x_k)$ for $k \geq 1$, and it holds as well for $k = 0$ from Step (2.) of Procedure 3.5 if the pre-start step is used. If the pre-start step is not used, then for $k = 0$ we have $g_0 := A \tilde{\lambda}_0 \in \partial f(x_0)$ from the starting values (3.26).

To prove (2.), pre-multiply (3.28) by $A$ and re-index, which yields for $k \geq 1$ that

$$A \lambda_k = \frac{1}{\beta_k} \left[ A \lambda_1 + \sum_{i=1}^{k-1} \alpha_i A \lambda_i \right].$$

If no pre-start step is used, using (3.26) we have for $k \geq 1$ that:

$$A \lambda_k = \frac{1}{\beta_k} \left[ s_0 + \sum_{i=0}^{k-1} \alpha_i g_i \right] = \frac{1}{\beta_k} s_k,$$

since $s_k = s_0 + \sum_{i=0}^{k-1} \alpha_i g_i$, for $k \geq 1$. If the pre-start step is used, then using (3.27) and $\lambda_1 = \tilde{\lambda}_0$ (from Step (4.) of Procedure 3.5) we arrive similarly at:

$$A \lambda_k = \frac{1}{\beta_k} \left[ s_0 + \sum_{i=0}^{k-1} \alpha_i g_i \right] = \frac{1}{\beta_k} s_k \text{ for } k \geq 1.$$
Therefore Step (1.) of Algorithm 3.4 yields for $k \geq 1$ that:

$$x_k = \arg \min_{x \in P} \{x^T A \lambda_k + d(x)\}$$

$$= \arg \min_{x \in P} \{\frac{1}{\beta_k} s_k^T x + d(x)\}$$

$$= \arg \min_{x \in P} \{s_k^T x + \beta_k d(x)\},$$

which proves (2.).

---

**Discussion of various equivalences between Frank-Wolfe and other methods.** In [3], Bach shows an equivalence between Frank-Wolfe and the method of mirror descent applied to problem (3.2) (under certain assumptions). Note that the mirror descent method applied to (3.2) involves computing (sub)gradients of both $f(\cdot)$ and $d(\cdot)$. Note also that the results in [3] only apply under the key assumption that the prox function $d(\cdot)$ is essentially smooth, i.e., $\|
abla d(x)\|_* \to \infty$ as $x \to \partial P$. Thus, given the results in [3] and the results presented herein, we have the following diagram (a dashed line indicates that an equivalence is only true, strictly speaking, if the prox function $d(\cdot)$ is essentially smooth).

![Equivalencies of Frank-Wolfe, dual averaging, and mirror descent.](image)

**Figure 3-1:** Equivalencies of Frank-Wolfe, dual averaging, and mirror descent.
### 3.3.1 A New Complexity Analysis for Dual Averaging

The equivalence presented in Section 3.3 suggests a novel regime for the dual averaging sequences – namely sequences that satisfy \( \beta_k = \sum_{i=0}^{k-1} \alpha_i \). At the same time, it turns out that the computational guarantees presented in Chapter 2 apply not only for the primal Frank-Wolfe problem (3.22), but also yield results for the dual problem on the right hand side of (3.24). Thus, we are left with the following two questions: (i) can one derive complexity results for the dual problem on the right hand side of (3.24) via a direct analysis of dual averaging?, and (ii) can such results be stated and derived in a form that allows for more general non-smooth convex functions \( f(\cdot) \) (i.e., more general than functions of the form \( f(x) = \max_{\lambda \in Q} \{ x^T A \lambda \} \) considered previously in Section 3.3)? Herein, we answer both of these questions in the affirmative.

Let us return to the set up regarding problems (3.1) and (3.2) introduced in Section 3.1. Assume for simplicity that \( s_0 = 0 \) and \( x_0 = x^c = \arg\min_{x \in P} d(x) \). Recall some earlier definitions, namely \( A_k := \sum_{i=0}^{k-1} \alpha_i \) for \( k \geq 0 \) and the sequence of linear functions \( \{ \ell_k(\cdot) \} \) defined by:

\[
\ell_k(x) := \sum_{i=0}^{k-1} \alpha_i [f(x_i) + \langle g_i, x - x_i \rangle] \quad k \geq 0,
\]

and recall that \( \ell_k(\cdot) \) lower bounds \( A_k f(\cdot) \), i.e.,

\[
\ell_k(x) \leq A_k f(x) \quad \text{for all } x \in P. \tag{3.30}
\]

Also define the following functions:

\[
\Psi_k(x) := \ell_k(x) + \beta_k d(x),
\]

define \( \Psi_k^* := \min_{x \in P} \Psi_k(x) \) and observe that by definition \( x_k = \arg\min_{x \in P} \Psi_k(x) \).

Throughout this section, we assume that \( \mu_k = 1 \) for all \( k \geq 1 \), i.e., we have \( A_k = \beta_k \) for
all $k \geq 1$. Now since $A_k = \beta_k$, we have that $\Psi_k(\cdot)$ provides a lower bound on $A_k \phi(\cdot)$, i.e.,

$$
\Psi_k(x) = \ell_k(x) + \beta_k d(x) = \ell_k(x) + A_k d(x) \leq A_k (f(x) + d(x)) = A_k \phi(x),
$$

(3.31)

for all $k \geq 1$. We aim to show the following property:

$$
A_k \phi(\hat{x}_k) \leq \Psi_k^* + B_k \quad \text{for } k \geq 1,
$$

(3.32)

where $\hat{x}_k$ is a certain point computed from the iterates $x_0, \ldots, x_k$ and $B_k$ is a remainder term. Combining (3.31) and (3.32) yields:

$$
\phi(\hat{x}_k) - \phi(x) \leq \frac{B_k}{A_k} \quad \text{for all } x \in P \text{ and } k \geq 1.
$$

We assume that the subgradients of $f(\cdot)$ have bounded variation with constant $M_f$:

$$
\|g_x - g_y\|_* \leq M_f \quad \text{for all } x, y \in P, \ g_x \in \partial f(x), \ g_y \in \partial f(y).
$$

(3.33)

Note that assuming that $f(\cdot)$ has uniformly bounded subgradients, i.e,

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

(3.34)

is stronger than (3.33) since (3.34) implies (3.33) with $M_f \leq 2L_f$. On the other hand, $M_f$ may be significantly smaller than $2L_f$.

For $k \geq 1$, define

$$
\bar{s}_k := \frac{s_k}{A_k},
$$

which is the weighted average of previous subgradients. Let $x_{-1}$ be an arbitrary point in $P$ and consider the following initialization for the dual averaging method (Algorithm 3.1):

$$
s_0 = 0 \quad \text{and} \quad x_0 = \arg\min_{x \in P} \{\langle g_{-1}, x \rangle + d(x) \} \quad \text{where } g_{-1} \in \partial f(x_{-1}).
$$

(3.35)
Theorem 3.4. For the dual averaging method (Algorithm 3.1) with initialization (3.35), under the assumption that \( \mu_k = 1 \), we have property (3.32) with

\[
B_k := \frac{\alpha_k^2}{2\sigma A_1} \| g_0 - g_{-1} \|^2_s + \sum_{i=1}^{k-1} \frac{\alpha_i^2}{2\sigma A_{i+1}} \| g_i - \bar{s}_i \|^2_s \leq \frac{M_f^2}{2\sigma} \sum_{i=0}^{k-1} \frac{\alpha_i^2}{A_{i+1}}
\]

Proof. We use induction to show that the following holds for \( k \geq 1 \):

\[
\sum_{i=0}^{k-1} \alpha_i \phi(x_i) \leq \Psi_k^* + B_k . \tag{3.36}
\]

Clearly, if we select \( \hat{x}_k \) as an appropriate weighted average of \( x_0, \ldots, x_{k-1} \) or as the best iterate among \( x_0, \ldots, x_{k-1} \), then (3.36) implies (3.32). Let us demonstrate that (3.36) holds for \( k = 1 \). First notice that by the definition of \( x_0 \) in (3.35), we have:

\[
\langle g_{-1} + \nabla d(x_0), x - x_0 \rangle \geq 0 \quad \text{for all } x \in P . \tag{3.37}
\]
Now, for any $x \in P$, we have:

\[ \Psi_1(x) = \ell_1(x) + \beta_1 d(x) \]

\[ = \alpha_0 [f(x_0) + \langle g_0, x - x_0 \rangle] + \alpha_0 d(x) \]

\[ \geq \alpha_0 [f(x_0) + \langle g_0, x - x_0 \rangle] + \alpha_0 [d(x_0) + \langle \nabla d(x_0), x - x_0 \rangle + \frac{\sigma}{2} \| x - x_0 \|^2] \quad \text{(strong convexity of } d) \]

\[ = \alpha_0 \phi(x_0) + \alpha_0 \langle g_0, x - x_0 \rangle + \alpha_0 \langle \nabla d(x_0), x - x_0 \rangle + \frac{A_1 \sigma}{2} \| x - x_0 \|^2 \quad \alpha_0 = A_1 \]

\[ \geq \alpha_0 \phi(x_0) + \alpha_0 \langle g_0 - g_{-1}, x - x_0 \rangle + \frac{A_1 \sigma}{2} \| x - x_0 \|^2 \quad \text{(3.37)} \]

\[ \geq \alpha_0 \phi(x_0) - \frac{\alpha_0^2}{2 \sigma A_1} \| g_0 - g_{-1} \|^2_* \]

\[ = \alpha_0 \phi(x_0) - B_1 \]

Thus, taking $x$ as $x_1$ in the above yields:

\[ \alpha_0 \phi(x_0) \leq \Psi_1^* + B_1 . \]

Now, assume that (3.36) holds for $k \geq 1$. First note that the definition of $x_k$ implies that

\[ \langle s_k + A_k \nabla d(x_k), x - x_k \rangle \geq 0 \text{ for all } x \in P , \]

which implies that

\[ \langle \nabla d(x_k), x - x_k \rangle \geq -\langle s_k, x - x_k \rangle \text{ for all } x \in P . \]

78
Note also that strong convexity of \(d(\cdot)\) yields for any \(x \in P:\)

\[
d(x) \geq d(x_k) + \langle \nabla d(x_k), x - x_k \rangle + \frac{\sigma}{2} \|x - x_k\|^2 \geq d(x_k) - \langle \bar{s}_k, x - x_k \rangle + \frac{\sigma}{2} \|x - x_k\|^2.
\] (3.38)

Now for any \(x \in P\), we have:

\[
\Psi_{k+1}(x) = \ell_{k+1}(x) + A_{k+1}d(x)
\]

\[
= \ell_k(x) + \alpha_k [f(x_k) + \langle g_k, x - x_k \rangle] + A_kd(x) + (A_{k+1} - A_k)d(x)
\]

\[
= \Psi_k(x) + \alpha_k [f(x_k) + \langle g_k, x - x_k \rangle] + \alpha_k d(x)
\]

\[
\geq \Psi_k^* + \frac{A_k \sigma}{2} \|x - x_k\|^2 + \alpha_k [f(x_k) + \langle g_k, x - x_k \rangle] + \alpha_k d(x) \quad \text{(strong convexity of} \ \Psi_k(\cdot)\text{)}
\]

\[
\geq \Psi_k^* + \frac{A_k \sigma}{2} \|x - x_k\|^2 + \alpha_k [f(x_k) + \langle g_k, x - x_k \rangle] + \alpha_k d(x)
\]

\[
\geq \Psi_k^* + \frac{A_{k+1} \sigma}{2} \|x - x_k\|^2 + \alpha_k \langle g_k - \bar{s}_k, x - x_k \rangle + \alpha_k \phi(x_k) \quad \text{(Hölder)}
\]

\[
\geq \Psi_k^* + \frac{\alpha^2}{2 \sigma A_{k+1}} \|g_k - \bar{s}_k\|^2 + \alpha_k \phi(x_k) \quad \text{(induction hypothesis)}
\]

\[
\geq \sum_{i=0}^{k-1} \alpha_i \phi(x_i) - B_k - \frac{\alpha^2}{2 \sigma A_{k+1}} \|g_k - \bar{s}_k\|^2 + \alpha_k \phi(x_k)
\]

\[
= \sum_{i=0}^{k} \alpha_i \phi(x_i) - B_{k+1}.
\]

Thus taking \(x\) as \(x_{k+1}\) in the above completes the proof. \(\Box\)

We comment that the upper bound on \(B_k\) in Theorem 3.4 is exactly the same expression appearing in the numerator of Theorem 2.1. Thus, in particular, for the choice of sequences
$\alpha_i = i + 1$ and $\beta_i = i(i + 1)/2$, we have, as a consequence of (3.32):

$$\phi(\hat{x}_k) - \phi(x) \leq \frac{2M^2}{\sigma(k + 4)},$$

(3.39)
i.e., $O(1/k)$ convergence for problem (3.2). Note that an $O(1/k)$ convergence bound is standard for subgradient methods applied to strongly convex problems (see, for example, [3] and the references therein). The fact that this bound matches that for Frank-Wolfe is, of course, no coincidence. Finally, note that the analysis presented herein may also be extended to generate primal-dual guarantees, which in terms of Frank-Wolfe would yield guarantees for the Frank-Wolfe primal problem (which is the dual averaging dual problem).
Chapter 4

An Extended Frank-Wolfe Method with “In-Face” Directions, and its Application to Low-Rank Matrix Completion

4.1 Introduction

In the last ten years the problem of matrix completion (see, for example, [18, 19, 84]) has emerged as an important and ubiquitous problem in statistics and machine learning, with applications in diverse areas [17, 97], with perhaps the most notable being recommender systems [7, 8, 58]. In matrix completion one is given a partially observed data matrix $X \in \mathbb{R}^{m \times n}$, i.e., there is only knowledge of the entries $X_{ij}$ for $(i, j) \in \Omega$ where $\Omega \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\}$ (often, $|\Omega| \ll m \times n$), and the task is to predict (fill in) the unobserved entries of $X$. The observed entries are possibly contaminated with noise, i.e., $X = Z^* + E$ where $Z^* \in \mathbb{R}^{m \times n}$ represents the “true data matrix” and $E$ is the noise term, and the goal is to accurately estimate the entire matrix $Z^*$, which most importantly includes estimating the entries $Z^*_{ij}$ for $(i, j) \not\in \Omega$. Clearly, this problem is in general ill-posed – without any restrictions, the unobserved entries can take on any real values. The ill-posed nature of the
problem necessitates that any successful approach must, either explicitly or implicitly, make some type of assumption(s) about underlying *structure* of the matrix $Z^*$. The most common approach, especially without *a priori* knowledge about the data-generating mechanism, is to assume that the matrix $Z^*$ is *low-rank*. This situation is similar to the “bet on sparsity” principle in linear regression [51]: if $Z^*$ does not have low-rank structure, then we cannot expect any method to successfully fill in the missing entries; on the other hand, if $Z^*$ does have low-rank, then a method that makes such a structural assumption should have a better chance at success.

The low-rank structural assumption naturally leads to the following optimization problem:

$$\min\limits_{Z \in \mathbb{R}^{m \times n}} \frac{1}{2} \sum_{(i,j) \in \Omega} (Z_{ij} - X_{ij})^2$$

s.t. rank($Z$) $\leq r$ ,

where $r$ is a parameter representing the assumed belief about the rank of $Z^*$. Notice that (4.1) is a combinatorially hard problem due to the rank constraint [20].

Pioneered by [32], a promising strategy for attacking (4.1) is to use the *nuclear norm* as a proxy for the rank. Recall that, for a given $Z \in \mathbb{R}^{m \times n}$, the sum of the singular values of $Z$ is a norm often referred to as the nuclear norm. Directly replacing the combinatorially hard rank constraint in (4.1) with a constraint on the nuclear norm of $Z$ leads to the following convex optimization problem:

$$\|Z\|_{N1} \leq \delta \ .$$

Let $\mathcal{B}_{N_1}(Z, \delta) := \{Y \in \mathbb{R}^{m \times n} : \|Y - Z\|_{N1} \leq \delta\}$ denote the nuclear norm ball of radius $\delta$ centered at the point $Z$, so that the feasible region of (4.2) is $\mathcal{B}_{N_1}(0, \delta)$. Despite its apparent absence from the problem formulation, it is nevertheless imperative that computed solutions
of \ref{4.2} have low rank. Such low-rank computed solutions are coerced by the nuclear norm constraint, and there has been substantial and influential work showing that, for many types of data generating mechanisms, an optimal solution of \ref{4.2} will have appropriately low rank (see, for instance, \cite{18, 19, 32, 83}). This line of work typically focuses on studying the properties of optimal solutions of \ref{4.2}, and thus abstracts away the choice of algorithm to solve \ref{4.2}. Although this abstraction may be reasonable in some situations, and is certainly a reasonable way to study the benefits of nuclear norm regularization, it may also be limiting. Indeed, in recent years, the notion that “convex optimization is a black box” has become increasingly unreasonable. Concurrently with the explosion of “big data” applications, there has been a substantial amount of recent work on the development and analysis of algorithms for huge-scale convex optimization problems where interior point methods and other polynomial-time algorithms are ineffective. Moreover, there has been an increasing interest in algorithms that directly promote desirable structural properties of their iterates. One such algorithm that satisfies both of these properties – scalability to huge-size problems and structurally favorable iterates – is the Frank-Wolfe Method and its extensions, which is the starting point of the work herein. Indeed, much of the recent computational work for matrix completion is based on directly applying first-order methods and related methods that have structurally favorable iterates \cite{16 56 64 96}. Mazumder et al. \cite{67} develop a related algorithm based on SVD soft thresholding that efficiently utilizes the special structure of matrix completion problems. In one of the earlier works applying the Frank-Wolfe Method to nuclear norm regularized problems, Jaggi and Sulovský \cite{55} consider first lifting the nuclear norm regularized problem \ref{4.2} to a problem over the semidefinite cone and then apply the Frank-Wolfe Method. Tewari et al. \cite{94} as well as Harchaoui et al. \cite{48} pointed out that the Frank-Wolfe Method can be applied directly to the nuclear norm regularized problem \ref{4.2}, and \cite{48} also developed a variant of the method that applies to penalized nuclear norm problems, which was also studied in \cite{96}. Mishra et al. \cite{68} develop a second-order trust region method that shares a few curious similarities with the extended Frank-Wolfe Method developed herein. Mu et al. \cite{69} consider a hybrid proximal gradient/Frank-Wolfe method
for low-rank matrix and tensor recovery. Rao et al. [82] consider a variant of Frank-Wolfe with “backward steps” (which differ from the classical “away steps” of Wolfe [100] and Guélat and Marcotte [47]) in the general context of atomic norm regularization. Backward steps comprise a flexible methodology aimed at producing sparse representations of solutions. In this regard, backward steps are unrelated to away steps except to the extent that both may result in sparse solutions.

**The Frank-Wolfe Method, in-face directions, and structural implications.** Due to its low iteration cost and convenient structural properties (as we shall soon discuss), the Frank-Wolfe Method (also called the conditional gradient method and introduced in Chapter 2) is especially applicable in several areas of machine learning and has thus received much renewed interest in recent years, see [34, 49, 54, 62, 94] and the references therein. The Frank-Wolfe Method, originally developed by [33] in the context of quadratic programming, was later generalized to convex optimization problems with smooth (differentiable) convex objective functions and bounded convex feasible regions, of which (4.2) is a particular instance. Indeed, letting \( f(Z) := \frac{1}{2} \sum_{(i,j) \in \Omega} (Z_{ij} - X_{ij})^2 \) denote the least squares objective in (4.2), it is easy to see that \( f(\cdot) \) is a smooth convex function, and the feasible region of (4.2) is \( B_{N_1}(0, \delta) \), which is a bounded convex set.

As applied to problem (4.2), the Frank-Wolfe Method proceeds at the current iterate \( Z^k \) by solving a linear optimization subproblem to compute \( \tilde{Z}^k \leftarrow \arg \min_{Z \in B_{N_1}(0, \delta)} \{ \nabla f(Z^k) \cdot Z \} \) (here “\( \cdot \)” denotes the usual trace inner product) and updates the next iterate as

\[
Z^{k+1} \leftarrow Z^k + \bar{\alpha}_k (\tilde{Z}^k - Z^k)
\]

(4.3)

for some \( \bar{\alpha}_k \in [0, 1] \). As was discussed in Section 2.3 of Chapter 2 and as developed in [49, 54] as well, for appropriate choices of the step-size sequence \( \{ \bar{\alpha}_k \} \), it holds that

\[
f(Z^k) - f^* \leq \frac{8 \delta^2}{k + 3} \quad \text{and} \quad \text{rank}(Z^k) \leq k + 1.
\]

(4.4)

The bound on the objective function gap in (4.4) is well understood and follows from a
standard analysis of the Frank-Wolfe Method. The bound on the rank of $Z^k$ in (4.4), while also well understood, follows from the special structure of the nuclear norm ball. Specifically, and as we further expand upon in Sections 4.2 and 4.3, for the nuclear norm regularized matrix completion problem (4.2), the solutions to the linear optimization subproblem solved at each iteration are specially structured – they are rank one matrices arising from the leading left and right singular vectors of the matrix $\nabla f(Z^k)$. Thus, assuming that $Z^0$ is a rank one matrix, the simple additive form of the updates (4.3) leads to the bound on the rank in (4.4). The above bound on the rank of $Z^k$ is precisely the “favorable structural property” of the iterates of the Frank-Wolfe Method that was mentioned earlier, and when combined with the bound on the objective function gap in (4.4) yields a nice tradeoff between data fidelity and low-rank structure. However, note that when $k$ is large – as might be necessary if the desired objective function value gap needs to be very small – then the bound on the rank of $Z^k$ might not be as favorable as one might wish. Indeed, one of the primary motivations underlying the research herein is to develop theoretical and practical methods for solving (4.2) that simultaneously achieve both good data fidelity (i.e., a small optimality gap in (4.2)) and low rank of the iterates $Z^k$.

Here we see that in the case of the Frank-Wolfe Method, the properties of the algorithm provide additional insight into how problem (4.2) induces low-rank structure. A natural question is: can the tradeoff given by (4.4) be improved, either theoretically or practically or both? That is, can we modify the Frank-Wolfe Method in a way that maintains the bound on the objective function gap in (4.4) while strictly improving the bound on the rank? This is the motivation for the development of what we call “in-face” directions and their subsequent analysis herein. We define an in-face direction to be any descent direction that keeps the next iterate within the minimal face of $B_{\mathcal{N}1}(0, \delta)$ containing the current iterate (where the minimal face of a point $x \in S$ is the smallest face of the convex set $S$ that contains the point $x$). It turns out that the faces of the nuclear norm ball are characterized by the (thin) SVDs of the matrices contained within them [91]. Therefore an in-face direction will move to a new point $Z^{k+1}$ with a similar SVD structure as $Z^k$, and moreover will keep the rank of
\( Z^{k+1} \) the same (or will lower it, which is even better), i.e., \( \text{rank}(Z^{k+1}) \leq \text{rank}(Z^k) \). Clearly if we can find good in-face directions, then the bound on the rank in (4.4) will be improved. At the same time, if there are no in-face directions that are “good enough” with respect to improvements in objective function values, then a “regular” Frank-Wolfe direction may be chosen, which will usually increase the rank of the next iterate by one. In this chapter, we develop an extension of the Frank-Wolfe Method that incorporates in-face directions and we provide both a precise theoretical analysis of the resulting tradeoff akin to (4.4), as well as computational results that demonstrate significant improvements over existing methods both in terms of ranks and run times.

### 4.1.1 Organization/Results

The chapter is organized as follows. In Section 4.2 after reviewing the basic Frank-Wolfe Method and the away-step modification of Wolfe and Guélat and Marcotte, we present our extended Frank-Wolfe Method based on “in-face” directions (in addition to regular Frank-Wolfe directions), this being the main methodological contribution of the chapter. This In-Face Extended Frank-Wolfe Method is specifically designed to induce iterates that lie on low-dimensional faces of the feasible set \( S \), since low-dimensional faces of the feasible region contain desirable “well-structured” points (sparse solutions when \( S \) is the \( \ell_1 \) ball, low-rank matrices when \( S \) is the nuclear norm ball). The in-face directions are any directions that keep the current iterate in its current minimal face of \( S \). We present two main strategies for computing in-face directions: (i) away-steps as introduced by Wolfe [100] and Guélat and Marcotte [47], and (ii) approximate full optimization of the objective \( f(\cdot) \) over the current minimal face. The In-Face Extended Frank-Wolfe Method uses a simple decision criterion for selecting between in-face and regular Frank-Wolfe directions. In Theorem 4.2 we present computational guarantees for the In-Face Extended Frank-Wolfe Method. These guarantees essentially show that the In-Face Extended Frank-Wolfe Method maintains \( O(c/k) \) convergence after \( k \) iterations (which is optimal for Frank-Wolfe type methods in the absence of polyhedral structure or strong convexity [62]), all the while promoting low-rank iterates via
the parameters of the method which affect the constant $c$ above, see Theorem 4.2 for specific details.

In Section 4.3 we discuss in detail how to apply the In-Face Extended Frank-Wolfe Method to solve the matrix completion problem (4.2). We resolve issues such as characterizing and working with the minimal faces of the nuclear norm ball, solving linear optimization subproblems on the nuclear norm ball and its faces, computing steps to the boundary of the nuclear norm ball, and updating the SVD of the iterates. In Proposition 4.2 we present a bound on the ranks of the matrix iterates of the In-Face Extended Frank-Wolfe Method that specifies how the in-face directions reduce the rank of the iterates over the course of the algorithm. Furthermore, as a consequence of our developments we also demonstrate, for the first time, how to effectively apply the away-step method of [47] to problem (4.2) in a manner that works with the natural parameterization of variables $Z \in \mathbb{R}^{m \times n}$ (as opposed to an “atomic” form of [47], as we expand upon at the end of Section 4.2.1).

Section 4.4 contains a detailed computational evaluation of the In-Face Extended Frank-Wolfe Method and discusses several versions of the method based on different strategies for computing in-face directions and different algorithmic parameter settings. We compare these versions to the regular Frank-Wolfe Method, the away-step method of [47], an atomic version of [47] (as studied in [4, 59, 60, 79]), as well as the “fully corrective” variant of Frank-Wolfe [49, 54, 60] and the CoGEnT “forward-backward” method of [82]. We present several experiments on simulated problem instances as well as on the MovieLens10M dataset. Our results demonstrate that the In-Face Extended Frank-Wolfe Method (in different versions) shows significant computational advantages in terms of delivering low rank and low run time to compute a target optimality gap. Especially for larger instances, one version of our method delivers very low rank solutions with reasonable run times, while another version delivers the best run times, beating existing methods by a factor of 10 or more.
4.1.2 Notation

Let $E$ be a finite-dimensional linear space. For a norm $\| \cdot \|$ on $E$, let $\| \cdot \|^*$ be the associated dual norm, namely $\|c\|^* := \max\{c^T z : \|z\| \leq 1\}$ and $c^T z$ denotes the value of the linear operator $c$ acting on $z$. The ball of radius $\delta$ centered at $\bar{z}$ is denoted $\mathcal{B}(\bar{z}, \delta) := \{ z : \|z - \bar{z}\| \leq \delta \}$. We use $I$ to denote the identity matrix whose dimension is dictated by the context. For $X, Y \in S^{k \times k}$ (the set of $k \times k$ symmetric matrices), we write "$X \succeq 0$" to denote that $X$ is symmetric and positive semidefinite, "$X \succeq Y$" to denote that $X - Y \succeq 0$, and "$X \succ 0$" to denote that $X$ is positive definite, etc. For a given $Z \in \mathbb{R}^{m \times n}$ with $r := \text{rank}(Z)$, the (thin) singular value decomposition (SVD) of $Z$ is $Z = UDV^T$ where $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ are each orthonormal ($U^T U = I$ and $V^T V = I$), and $D = \text{Diag}(\sigma_1, \ldots, \sigma_r)$ comprises the non-zero (hence positive) singular values of $Z$. The nuclear norm of $Z$ is then defined to be $\|Z\|_{N1} := \sum_{j=1}^r \sigma_j$ . (In much of the literature, this norm is denoted $\| \cdot \|_*$; we prefer to limit the use of "*" to dual norms, and hence we use the notation $\| \cdot \|_{N1}$ instead.) Let $\mathcal{B}_{N1}(Z, \delta) := \{ Y \in \mathbb{R}^{m \times n} : \|Y - Z\|_{N1} \leq \delta \}$ denote the nuclear norm ball of radius $\delta$ centered at the point $Z$. Let $\|Z\|_F$ denote the Frobenius norm of $Z$, namely $\|Z\|_F = \sqrt{\sum_{j=1}^r \sigma_j^2} = \sqrt{\text{Tr}(Z^T Z)}$. The dual norm of the nuclear norm is the largest singular value of a matrix and is denoted by $\| \cdot \|^*_{N1} = \| \cdot \|^*_{N\infty}$; given $S \in \mathbb{R}^{m \times n}$ with SVD $S = UDV^T$, then $\|S\|_{N\infty} = \max\{\sigma_1, \ldots, \sigma_r\}$. A spectrahedron is a set of the form $\mathcal{S}^k_t := \{ X \in \mathbb{S}^{k \times k} : X \succeq 0, I \bullet X \leq t \}$ or $\bar{\mathcal{S}}^k_t := \{ X \in \mathbb{S}^{k \times k} : X \succeq 0, I \bullet X = t \}$, where "$\bullet$" denotes the usual trace inner product.

4.2 Frank-Wolfe Method, Away Steps, and In-Face Steps

Problem (4.2) is an instance of the more general problem:

$$f^* := \min_{x \in S} f(x)$$

(4.5)

where $S \subset E$ is a closed and bounded convex set, and $f(\cdot)$ is a differentiable convex function
on $S$. We first review solving instances of (4.5) using the Frank-Wolfe Method, which was introduced in Chapter 2. We restate the basic description of the method in minimization form in Algorithm 4.1.

**Algorithm 4.1** Frank-Wolfe Method for optimization problem (4.5)

Initialize at $x_0 \in S$, (optional) initial lower bound $B_{-1}$, $k \leftarrow 0$.

At iteration $k$:
1. Compute $\nabla f(x_k)$.
2. Compute $\bar{x}_k \leftarrow \arg \min_{x \in S} \{ f(x_k) + \nabla f(x_k)^T (x - x_k) \}$.
   \[
   B_k^w \leftarrow f(x_k) + \nabla f(x_k)^T (\bar{x}_k - x_k).
   \]
   Update best bound: $B_k \leftarrow \max\{B_{k-1}, B_k^w\}$.
3. Set $x_{k+1} \leftarrow x_k + \bar{\alpha}_k (\bar{x}_k - x_k)$, where $\bar{\alpha}_k \in [0, 1]$.

Typically the main computational burden at each iteration of the Frank-Wolfe Method is solving the linear optimization subproblem in Step (2.) of Algorithm 4.1. The quantities $B_k^w$ are lower bounds on the optimal objective function value $f^*$ of (4.5), a fact which follows easily from the gradient inequality, see Jaggi [54] or Chapter 2, and hence $B_k = \max\{B_{k-1}, B_0^w, \ldots, B_k^w\}$ is also a lower bound on $f^*$. In addition to being useful in termination criteria, the lower bound sequence $\{B_k\}$ can be used in a variety of step-size strategies as was developed in Chapter 2.

Recall that when the step-size sequence $\{\bar{\alpha}_k\}$ is chosen using the simple rule $\bar{\alpha}_k := \frac{2}{k+2}$, then the Frank-Wolfe Method has the following computational guarantee at the $k^{th}$ iteration, for $k \geq 0$:

\[
    f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{2LD^2}{k+3}, \quad (4.6)
\]

where $D := \max_{x,y \in S} \|x - y\|$ is the diameter of $S$, and $L$ is a Lipschitz constant of the gradient of $f(\cdot)$ on $S$, namely:

\[
    \|\nabla f(x) - \nabla f(y)\|_* \leq L \|x - y\| \quad \text{for all } x, y \in S. \quad (4.7)
\]

If $\bar{\alpha}_k$ is instead chosen by exact line-search, namely $\bar{\alpha}_k \leftarrow \arg \min_{\alpha \in [0,1]} f(x_k + \alpha (\bar{x}_k - x_k))$, then the guarantee (4.6) still holds, see Section 2.3.4, this being particularly relevant when
\( f(\cdot) \) is a convex quadratic as in \((4.2)\) in which case the exact line-search reduces to a simple formula. Alternatively, one can consider a step-size rule based on minimizing an upper-approximation of \( f(\cdot) \) inherent from the smoothness of the gradient, namely:

\[
 f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \| y - x \|^2 \quad \text{for all } x, y \in S ,
\]

which follows from \((4.7)\) (see Appendix A, for example, for a concise proof). The following is a modest extension of the original analysis of Frank and Wolfe in [33].

**Theorem 4.1. (extension of [33])** Let \( \bar{L} \geq L \) be given, and consider using either an exact line-search or the following step-size rule for the Frank-Wolfe Method:

\[
 \bar{\alpha}_k \leftarrow \min \left\{ \frac{\nabla f(x_k)^T (x_k - \bar{x}_k)}{L \| x_k - \bar{x}_k \|^2} , 1 \right\} \quad \text{for all } k \geq 0 .
\]

Then \( f(x_k) \) is monotone decreasing in \( k \), and it holds that:

\[
 f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{1}{\frac{1}{f(x_0) - B_0} + \frac{k}{2LD^2}} < \frac{2LD^2}{k} .
\]

**Proof:** The first inequality of \((4.10)\) follows from the fact that \( B_k \leq f^* \), and the third inequality follows from the fact that \( f(x^0) \geq f^* \geq B_0 \). The second inequality can be rewritten as:

\[
 \frac{1}{f(x_k) - B_k} \geq \frac{1}{f(x_0) - B_0} + \frac{k}{2LD^2} ,
\]

which states that the reciprocal of the optimality bound gap grows at least according to the indicated linear function in \( k \). The above inequality holds trivially for \( k = 0 \), and hence to prove the second inequality of \((4.10)\) it suffices to show that:

\[
 \frac{1}{f(x_{k+1}) - B_{k+1}} \geq \frac{1}{f(x_k) - B_k} + \frac{1}{2LD^2} \quad \text{for all } k \geq 0 ,
\]

whose proof is given in Appendix B and wherein the monotonicity of \( f(x_k) \) is also proved. \( \square \)
In addition to being the crux of the proof of (4.10), we will also use inequality (4.11) and related inequalities as the basis for choosing among candidate directions in the in-face extension of Frank-Wolfe that we will develop in Section 4.2.2.

4.2.1 Away Steps

In [100] Wolfe introduced the concept of an “away step” in a modified version of the Frank-Wolfe method, and Guélat and Marcotte [47] provided a modification thereof and an extensive treatment of the convergence properties of the away-step-modified Frank-Wolfe method, including eventual linear convergence of the method when the objective function is strongly convex, the feasible region is polyhedral, and a form of strict complementarity holds. Quite recently there has been much renewed interest in the Frank-Wolfe method with away steps, with most of the focus being on demonstrating global linear convergence with computational guarantees for a particular “atomic” version of [47], see Lacoste-Julien and Jaggi [59, 60], Beck and Shtern [4], and Peña et al. [79].

Algorithm 4.2 presents the modified Frank-Wolfe Method with Away Steps as developed in [47]. The algorithm needs to work with the minimal face of a point $x \in S$, which is the smallest face of $S$ that contains the point $x$; here we use the notation $\mathcal{F}_S(x)$ to denote the minimal face of $S$ which contains $x$. Step (2.) of the modified Frank-Wolfe method is the “away step” computation, where $\hat{x}_k$ is the point on the current minimal face $\mathcal{F}_S(x_k)$ that is farthest along the ray from the “bad” solution $\hat{x}_k$ through the current point $x_k$. Step (3.) of the modified method is the regular Frank-Wolfe step computation, which is called a “toward step” in [47]. (Please see [47] as well as [100] for an expanded exposition of away-steps, including illustrative figures.) Notice that implementation of the away-step modified Frank-Wolfe method depends on the ability to characterize and work with the minimal face $\mathcal{F}_S(x_k)$ of the iterate $x_k$. When $S$ is not a polytope this minimal face capability is very much dependent on problem-specific knowledge of the structure of the set $S$.

The convergence of the modified Frank-Wolfe method is proved in Theorem 4 of [47] under
Algorithm 4.2 Modified Frank-Wolfe Method with Away Steps, for optimization problem (4.5)

Initialize at \(x_0 \in S\), (optional) initial lower bound \(B_{-1}, k \leftarrow 0\).

At iteration \(k\):
1. Compute \(\nabla f(x_k)\).
2. Compute \(\hat{x}_k \leftarrow \arg \max_x \{\nabla f(x_k)^T x : x \in F_S(x_k)\}\).
   \(\alpha_{\text{stop}}^k \leftarrow \arg \max_\alpha \{\alpha : x_k + \alpha (x_k - \hat{x}_k) \in F_S(x_k)\}\).
   \(\hat{x}_k \leftarrow x_k + \alpha_{\text{stop}}^k (x_k - \hat{x}_k)\).
3. Compute \(\tilde{x}_k \leftarrow \arg \min_x \{\nabla f(x_k)^T x : x \in S\}\).
   \(B^w_k \leftarrow f(x_k) + \nabla f(x_k)^T (\tilde{x}_k - x_k)\).
   Update best bound: \(B_k \leftarrow \max\{B_{k-1}, B^w_k\}\).
4. Choose descent direction:
   If \(\nabla f(x_k)^T (\tilde{x}_k - x_k) \leq \nabla f(x_k)^T (\hat{x}_k - x_k)\), then \(d_k \leftarrow \tilde{x}_k - x_k\) and \(\bar{\beta}_k \leftarrow 1\);
   Else \(d_k \leftarrow x_k - \hat{x}_k\) and \(\bar{\beta}_k \leftarrow \alpha_{\text{stop}}^k\).
5. Set \(x_{k+1} \leftarrow x_k + \bar{\alpha}_k d_k\), where \(\bar{\alpha}_k \in [0, \bar{\beta}_k]\).

the assumption that \(\bar{\alpha}_k\) in Step (5.) is chosen by exact line-search; however a careful review of the proof therein shows that convergence is still valid if one uses a step-size rule in the spirit of (4.9) that uses the quadratic upper-approximation of \(f(\cdot)\) using \(L\) or \(\tilde{L} \geq L\). The criterion in Step (4.) of Algorithm 4.2 for choosing between the regular Frank-Wolfe step and the away step seems to be tailor-made for the convergence proof in [47]. In examining the proof of convergence in [47], one finds the fact that \(\hat{x}_k\) is an extreme point is not relevant for the proof, nor even is the property that \(\hat{x}_k\) is a solution of a linear optimization problem. Indeed, this begs for a different way to think about both generating and analyzing away steps, which we will do shortly in Subsection 4.2.2.

Away-steps are not affine-invariant. The feasible region \(S\) of (4.5) can always be (implicitly) expressed as \(S = \text{conv}(A)\) where \(A = \{\tilde{x}^j : j \in J\}\) is a (possibly infinite) collection of points in \(S\) that includes all of the extreme points of \(S\). In fact, in many current applications of Frank-Wolfe and its relatives, \(S\) is explicitly constructed as \(S := \text{conv}(A)\) for a given collection \(A\) whose members are referred to as “atoms”; and each atom \(\tilde{x}^j \in A\) is a particularly “simple” point (such as a unit coordinate vector \(\pm e^i\), a rank-1 matrix, etc.). Let us consider the (possibly infinite-dimensional) vector space \(V := \{\alpha \in \mathbb{R}^{|J|} : \text{support}(\alpha)\text{ is finite}\}\), and
define the simplicial set $\Delta_{\mathcal{J}}$ by:

$$\Delta_{\mathcal{J}} := \left\{ \alpha \in V : \alpha \geq 0, \sum_{j \in \mathcal{J}} \alpha_j = 1 \right\},$$

and consider the linear map $M(\cdot) : \Delta_{\mathcal{J}} \to S$ such that $M(\alpha) := \sum_{j \in \mathcal{J}} \alpha_j \tilde{x}^j$. Then it is obvious that the following two optimization problems are equivalent:

$$\min_{x \in S} f(x) \quad \equiv \quad \min_{\alpha \in \Delta_{\mathcal{J}}} f(M(\alpha)),$$

(4.12)

where the left-side is our original given problem of interest (4.5) and the right-side is its re-expression using the convex weights $\alpha \in \Delta_{\mathcal{J}}$ as the variables. Furthermore, it follows from the fundamental affine-invariance of the regular Frank-Wolfe Method (Algorithm 4.1) as articulated by Jaggi [54] that the Frank-Wolfe Method applied to the left-side problem above is equivalent (via the linear mapping $M(\cdot)$) to the Frank-Wolfe Method applied to the right-side problem above. However, this affine invariance property does not extend to the away-step modification of the method, due to the fact that the facial structure of a convex set is not affine invariant – not even so in the case when $S$ is a polytope. This is illustrated in Figure 4-1. The left panel shows a polytopal feasible region $S \subset \mathbb{R}^3$ with $\mathcal{F}_S(x_k)$ highlighted. The polytope $S$ has 10 extreme points. The right panel shows $\mathcal{F}_S(x_k)$ by itself in detail, wherein we see that $x_k = .25\tilde{x}_1 + .25\tilde{x}_2 + .50\tilde{x}_3$ (among several other combinations of other extreme points of $\mathcal{F}_S(x_k)$ as well). Let us now consider the atomic expression of the set $S$ using the 10 extreme points $S$ and instead expressing our problem in the format of the right-side of (4.12), wherein the feasible region is the unit simplex in $\mathbb{R}^{10}$, namely $\Delta_{10} := \{ \alpha \in \mathbb{R}^{10} : \alpha \geq 0, e^T\alpha = 1 \}$ where $e = (1, \ldots, 1)$ is the vector of ones. If the current iterate $x_k$ is given the atomic expression $\alpha_k = (.25, .25, .50, 0, 0, 0, 0, 0, 0, 0)$, then the minimal face $\mathcal{F}_{\Delta_{10}}(\alpha_k)$ of $\alpha_k$ in $\Delta_{10}$ is the sub-simplex $\{ \alpha \in \mathbb{R}^{10} : \alpha \geq 0, e^T\alpha = 1, \alpha_4 = \cdots \alpha_{10} = 0 \}$, which corresponds back in $S \subset \mathbb{R}^3$ to the narrow triangle in the right panel of Figure 4-1 and which is a small subset of the pentagon corresponding to the minimal face $\mathcal{F}_S(x_k)$ of $x_k$ in $S$. Indeed, this example illustrates the general fact that the faces of the atomic expression
of \( S \) will always correspond to subsets of the faces of the facial structure of \( S \). Therefore, away-step sub-problem optimization computations using the original representation of \( S \) will optimize over larger subsets of \( S \) than will the corresponding computations using the atomic re-expression of the problem. Indeed, we will show in Section 4.4 in the context of matrix completion that by working with the original representation of the set \( S \) in the setting of using away-steps, one can obtain significant computational savings over working with the atomic representation of the problem.

Last of all, we point out that the away-step modified Frank-Wolfe methods studied by Lacoste-Julien and Jaggi [59, 60], Beck and Shtern [4], and Peña et al. [79] can all be viewed as applying the away-step method (Algorithm 4.2) to the “atomic” representation of the optimization problem as in the right-side of (4.12).

![Figure 4-1: Illustration that facial structure of a polytope is not affine invariant.](image)

### 4.2.2 An “In-Face” Extended Frank-Wolfe Method

Here we present an “in-face” extension of the Frank-Wolfe method, that is significantly more general than the away-step method of Wolfe [100] and Guélat and Marcotte [47] (Algorithm 4.2), and its atomic version studied by by Lacoste-Julien and Jaggi [59, 60], Beck and Shtern [4], and Peña et al. [79]. The method is motivated by the desire to compute and work with points \( x \) that have specific structure, usually sparsity (in the case when \( x \) is a vector or matrix) or low-rank (in the case when \( x \) is a matrix). More generally, we will think of the structure as being related to the dimension of the minimal face \( \mathcal{F}_S(x) \) of \( S \) containing \( x \). The algorithm is designed to balance progress towards two different goals, namely (i)
progress towards optimizing the objective function, and (ii) the aim of having the iterates lie in low-dimensional faces of $S$. In the case of the matrix completion problem (4.2) in particular, if an iterate lies in a low-dimensional face of $S$ then the iterate will have low rank (see Theorem 4.3). Such low rank is advantageous not only because we want the output solution to have low rank, but also because a low-rank iterate yields a substantial reduction in the computation costs at that iteration. This last point will be further developed and exploited in Sections 4.3 and 4.4.

We present our “In-Face Extended Frank-Wolfe Method” in Algorithm 4.3. At Step (2.) of each iteration the algorithm works with an “in-face” direction $d_k$ which will keep the next candidate point in the current minimal face $\mathcal{F}_S(x_k)$. This is equivalent to requiring that $x_k + d_k$ lies in the affine hull of $\mathcal{F}_S(x_k)$, which is denoted by $\text{Aff}(\mathcal{F}_S(x_k))$. Other than the affine hull condition, the direction $d_k$ can be any descent direction of $f(\cdot)$ at $x_k$ if such a direction exists. The candidate iterate $x_k^B$ is generated by stepping in the direction $d_k$ all the way to the relative boundary of the minimal face of the current point $x_k$. The point $x_k^A$ is the candidate iterate generated using the in-face direction and a suitable step-size $\bar{\beta}_k$, perhaps chosen by exact line-search or by a quadratic approximation rule. In Steps (3a.) and (3b.) the algorithm applies criteria for choosing which, if any, of $x_k^B$ or $x_k^A$ to accept as the next iterate of the method. If the criteria are not met for either $x_k^B$ or $x_k^A$, then the method computes a regular Frank-Wolfe step in Step (3c.) and updates the lower bound $B_k$.

Let us now discuss a few strategies for computing in-face directions. One recovers the away-step direction of the method of Guélat and Marcotte [47] by choosing:

$$d_k \leftarrow x_k - \hat{x}_k,$$

where

$$\hat{x}_k \leftarrow \arg \max_x \{ \nabla f(x_k)^T x : x \in \mathcal{F}_S(x_k) \}.$$

Another natural way to compute a suitable $d_k$, that is computationally facile for relatively low-dimensional faces and for certain problem instances (including matrix completion), is to directly solve for an (approximately) optimal objective function solution over the low-
Algorithm 4.3 In-Face Extended Frank-Wolfe Method for optimization problem (4.5)

Initialize at $x_0 \in S$, (optional) initial lower bound $B_{-1}, k \leftarrow 0$.
Choose $\bar{L} \geq L, \bar{D} \geq D$, and constants $\gamma_1, \gamma_2$ satisfying $0 \leq \gamma_1 \leq \gamma_2 \leq 1$.

At iteration $k$:
1. Compute $\nabla f(x_k) \cdot B_k \leftarrow B_{k-1}$.
2. Compute direction $d_k$ for which $x_k + d_k \in \text{Aff}(\mathcal{F}_S(x_k))$ and $\nabla f(x_k)^T d_k < 0$. (If no $d_k$ exists, go to Step (3c.).)
\[
\alpha_{k}^{\text{stop}} \leftarrow \arg\max \{\alpha : x_k + \alpha d_k \in \mathcal{F}_S(x_k)\}.
\]
\[
x_k^B := x_k + \alpha_{k}^{\text{stop}} d_k.
\]
\[
x_k^A := x_k + \beta_k d_k \text{ where } \beta_k \in [0, \alpha_{k}^{\text{stop}}].
\]
3. Choose next iterate:
   (a.) (Go to a lower-dimensional face.)
   If $\frac{1}{f(x_k^B) - B_k} \geq \frac{1}{f(x_k) - B_k} + \frac{\gamma_1}{2LD^2}$, set $x_{k+1} \leftarrow x_k^B$.
   (b.) (Stay in current face.)
   Else if $\frac{1}{f(x_k^A) - B_k} \geq \frac{1}{f(x_k) - B_k} + \frac{\gamma_2}{2LD^2}$, set $x_{k+1} \leftarrow x_k^A$.
   (c.) (Do regular FW step and update lower bound.) Else, compute:
   $\tilde{x}_k \leftarrow \arg\min \{\nabla f(x_k)^T x : x \in S\}$.
\[
x_{k+1} \leftarrow x_k + \bar{\alpha}_k(\tilde{x}_k - x_k) \text{ where } \bar{\alpha}_k \in [0, 1].
\]
\[
B_k^w \leftarrow f(x_k) + \nabla f(x_k)^T (\tilde{x}_k - x_k), \quad B_k \leftarrow \max\{B_{k-1}, B_k^w\}.
\]
dimensional face $\mathcal{F}_S(x_k)$ and thereby set:

$$
    d_k \leftarrow x_k^M - x_k, \quad \text{where} \quad x_k^M \leftarrow \arg \min_x \{ f(x) : x \in \mathcal{F}_S(x_k) \}.
$$

(4.14)

Note that in this case, we may naturally set $\bar{\beta}_k := 1$. Another related type of in-face direction that may be of interest is to consider a regular Frank-Wolfe step within $\mathcal{F}_S(x_k)$, whereby we select:

$$
    d_k \leftarrow \tilde{x}_k^F - x_k, \quad \text{where} \quad \tilde{x}_k^F \leftarrow \arg \min_x \{ \nabla f(x_k)^T x : x \in \mathcal{F}_S(x_k) \}.
$$

(4.15)

One may interpret this “in-face Frank-Wolfe step” as a single iteration of the Frank-Wolfe Method applied to the subproblem in (4.14). As we elaborate in Section 4.4 when discussing the practical merits of these approaches, our main interests are in the away-step strategy (4.13) and the full optimization strategy (4.14). Both of these in-face Frank-Wolfe step strategies lead to significant computational advantages over the regular Frank-Wolfe Method, as will be shown in Section 4.4.

One immediate advantage of the In-Face Extended Frank-Wolfe Method (Algorithm 4.3) compared to the away-step modified Frank-Wolfe method of Guélat and Marcotte [47] (Algorithm 4.2) has to do with the number and sizes of linear optimization sub-problems that are solved. Algorithm 4.2 needs to solve two linear optimization subproblems at each iteration – a “small” subproblem on the minimal face $\mathcal{F}_S(x_k)$ and a “large” subproblem on the entire set $S$. In contrast, even when computing directions using away-step computations, Algorithm 4.3 must solve the “small” linear optimization problem on the minimal face $\mathcal{F}_S(x_k)$, but the method will only need to solve the “large” subproblem on the entire set $S$ if it needs to process Step (3c.). The computational advantage from not having to solve the “large” subproblem at every iteration will be shown in Section 4.4.

We now discuss the criteria that are used in Step (3.) to choose between the next step $x_k^B$ that lies in the relative boundary of the current minimal face $\mathcal{F}_S(x_k)$, the step $x_k^A$ that does not necessarily lie in the relative boundary of the current minimal face $\mathcal{F}_S(x_k)$, and a regular Frank-Wolfe step. We see from Step (3.) of Algorithm 4.3 that a regular Frank-Wolfe step
will be chosen as the next iterate unless the criterion of either Step (3a.) or (3b.) are met.

The criteria in Step (3a.) is met if $x^B_k$ (which lies on the relative boundary of $\mathcal{F}_S(x_k)$ by virtue of the definition of $\alpha_k^{\text{stop}}$) provides sufficient decrease in the optimality gap as measured with the criterion:

$$\frac{1}{f(x^B_k) - B_k} \geq \frac{1}{f(x_k) - B_k} + \frac{\gamma_1}{2LD^2}.$$

The criteria in Step (3b.) is met if $x^A_k$ provides sufficient decrease in the optimality gap as measured similar to above but using $\gamma_2$ rather than $\gamma_1$. Since $\gamma_1 \leq \gamma_2$, Step (3a.) requires a lesser decrease in the optimality bound gap than does Step (3b.).

In settings when we strongly desire to compute iterates that lie on low-dimensional faces (as in the low-rank matrix completion problem (4.2)), we would like the criteria in Steps (3a.) and (3b.) to be relatively easily satisfied (perhaps with it being even easier to satisfy the criteria in Step (3a.) as this will reduce the dimension of the minimal face). This can be accomplished by setting the values of $\gamma_1$ and $\gamma_2$ to be lower rather than higher. Indeed, setting $\gamma_1 = 0$ ensures in Step (3a.) that the next iterate lies in a lower-dimensional face whenever $x^B_k$ (which by definition lies in a lower dimensional face than $x_k$) does not have a worse objective function value than $f(x_k)$. Also, if one sets $\gamma_2$ to be smaller, then the criteria in Step (3b.) is more easily satisfied, which ensures that the new iterate will remain in the current face $\mathcal{F}_S(x_k)$ as desired when the criterion of Step (3b.) is satisfied.

As we have discussed, the ability to induce solutions on low-dimensional faces by setting $\gamma_1$ and $\gamma_2$ to have low values can be extremely beneficial. However, this all comes at a price in terms of computational guarantees, as we now develop. Before presenting the computational guarantee for Algorithm 4.3 we first briefly discuss step-sizes; the step-size $\beta_k$ for steps to the in-face point $x^A_k$ are determined in Step (2.), and the step-size $\alpha_k$ for regular Frank-Wolfe steps is chosen in Step (3c.). One strategy is to choose these step-sizes using an exact line-search if the line-search computation is not particularly burdensome (such as when $f(\cdot)$ is a quadratic function). Another strategy is to determine the step-sizes according to the quadratic upper approximation of $f(\cdot)$ much as in Theorem 4.1, which in this context means
choosing the step-sizes as follows:

$$\tilde{\beta}_k := \min \left\{ \frac{-\nabla f(x_k)^T d_k}{L \|d_k\|^2}, \alpha_{k, \text{step}} \right\}, \quad \tilde{\alpha}_k := \min \left\{ \frac{\nabla f(x_k)^T (x_k - \tilde{x}_k)}{L \|x_k - \tilde{x}_k\|^2}, 1 \right\}. \quad (4.16)$$

Let $N^a_k$, $N^b_k$, and $N^c_k$ denote the number of times within the first $k$ iterations that the iterates are chosen according to the criteria in Steps (3a.), (3b.), and (3c.), respectively. Then $k = N^a_k + N^b_k + N^c_k$, and we have the following computational guarantee.

**Theorem 4.2.** Suppose that the step-sizes used in Algorithm 4.3 are determined either by exact line-search or by (4.16). After $k$ iterations of Algorithm 4.3 it holds that:

$$f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{1}{f(x_0) - B_0} + \frac{\gamma_1 N^a_k}{2LD^2} + \frac{\gamma_2 N^b_k}{2LD^2} + \frac{N^c_k}{2LD^2} < \frac{2LD^2}{\gamma_1 N^a_k + \gamma_2 N^b_k + N^c_k}. \quad (4.17)$$

**Proof:** The first inequality is true since $B_k \leq f^*$, and the third inequality is true since $f(x_0) \geq B_0$, so we need only prove the second inequality, which can be equivalently written as:

$$\frac{1}{f(x_k) - B_k} \geq \frac{1}{f(x_0) - B_0} + \frac{\gamma_1 N^a_k}{2LD^2} + \frac{\gamma_2 N^b_k}{2LD^2} + \frac{N^c_k}{2LD^2}. \quad (4.17)$$

Notice that (4.17) is trivially true for $k = 0$ since $N^a_k = N^b_k = N^c_k = 0$ for $k = 0$. Let $\Delta^k := (f(x_k) - B_k)^{-1}$ denote the inverse objective function bound gap at iteration $k$. Then if the next iterate is chosen by satisfying the criteria in Step (3a.), it holds that $\Delta^{k+1} \geq (f(x_{k+1}) - B_k)^{-1} \geq \Delta^k + \frac{\gamma_1 N^a_k}{2LD^2}$ where the first inequality derives from $B_{k+1} \geq B_k$ and the second inequality is from the criterion of Step (3a.). Similarly, if the next iterate is chosen by satisfying the criteria in Step (3b.), it holds using similar logic that $\Delta^{k+1} \geq \Delta^k + \frac{\gamma_2 N^b_k}{2LD^2}$. And if the next iterate is chosen in Step (3c.), namely we take a regular Frank-Wolfe step, then inequality (4.11) holds, which is $\Delta^{k+1} \geq \Delta^k + \frac{1}{2LD^2}$. Applying induction then establishes (4.17), which completes the proof.

Here we see that choosing smaller values of $\gamma_1$ and $\gamma_2$ can have a detrimental effect on the progress of the algorithm in terms of the objective function optimality gap, while larger
values ensure better convergence guarantees. At the same time, smaller values of $\gamma_1$ and $\gamma_2$ are more effective at promoting iterates to lie on low-dimensional faces. Thus there is a clear tradeoff between objective function optimality gap accuracy and low-dimensional structure, dictated by the values of $\gamma_1$ and $\gamma_2$. One strategy that is worth studying is setting $\gamma_1 = 0$ and $\gamma_2$ to be relatively large, say $\gamma_2 = 1$ for example. With these values of the parameters we take an in-face step in Step (3a.) (which lowers the dimension of the face of the iterate) whenever doing so will not adversely affect the objective function value. This and other strategies for setting $\gamma_1$ and $\gamma_2$ will be examined in Section 4.4.

A simplified algorithm in the case of full optimization over the current minimal face. Let us further examine the dynamics of Algorithm 4.3 in the case of (4.14), where we select the in-face direction by fully optimizing the objective function $f(\cdot)$ over the low-dimensional face $\mathcal{F}_S(x_k)$. Consider performing an in-face step in this case, i.e., suppose that the next iterate is chosen according to the criteria in Steps (3a.)/(3b.) (recall that we set $\bar{\beta}_k := 1$ in this case). Then, at the next iteration, Algorithm 4.3 is guaranteed to select a regular Frank-Wolfe step via Step (3c.). Indeed, since the next iterate $x_{k+1}$ is chosen as the optimal solution over $\mathcal{F}_S(x_k)$, by definition there are no descent directions at $x_{k+1}$ that remain within $\mathcal{F}_S(x_{k+1}) \subseteq \mathcal{F}_S(x_k)$ and thus no valid in-face directions to be selected. Here we see that the parameters $\gamma_1$ and $\gamma_2$ are superfluous – a much more natural procedure is to simply alternate between regular Frank-Wolfe steps and fully optimizing over $\mathcal{F}_S(x_k)$. This bears some similarity to, but is distinct from, the “fully corrective” variant of Frank-Wolfe, see, e.g., [49, 54, 60]. (Indeed, these two algorithms coincide if we consider this alternating procedure applied to the lifted problem (4.12).) In this case, the following computational guarantee follows simply from Theorem 4.1.

**Proposition 4.1.** Consider a slight variation of Algorithm 4.3 that alternates between full optimizations (4.14) over the current face $\mathcal{F}_S(x_k)$ and regular Frank-Wolfe steps, with step-size $\bar{\alpha}_k$ chosen either by exact line-search or by a quadratic approximation rule (4.16). For simplicity, consider one iteration to consist of both of these operations in sequence.
for all $k \geq 0$, it holds that:

$$f(x_k) - f^* \leq f(x_k) - B_k \leq \frac{1}{f(x_0) - B_0} + \frac{k}{2LD^2} \leq \frac{2LD^2}{k}.$$ 

### 4.3 Solving Matrix Completion Problems using the In-Face Extended Frank-Wolfe Method

We now turn our attention to solving instances of (4.2) using the the In-Face Extended Frank-Wolfe Method (Algorithm 4.3). We work directly with the natural parameterization of variables as $m \times n$ matrices $Z \in \mathbb{R}^{m \times n}$ (although, as we discuss in Section 4.3.6, we utilize low-rank SVD updating to maintain the variables in an extremely memory efficient manner). Recall that the objective function of (4.2) is $f(Z) := \frac{1}{2} \sum_{(i,j) \in \Omega} (Z_{ij} - X_{ij})^2$, whose gradient is $\nabla f(Z) = (Z - X)_\Omega$. The feasible region of (4.2) is $S = \mathcal{B}_{\mathcal{N}_1}(0, \delta)$, which notation we shorten to $\mathcal{B} := \mathcal{B}_{\mathcal{N}_1}(0, \delta)$. We first discuss the specification and implementation issues in using Algorithm 4.3 to solve (4.2).

We will fix the norm on $Z$ to be the nuclear norm $\| \cdot \|_{\mathcal{N}_1}$, whose dual norm is easily seen to be $\| \cdot \|_{\mathcal{N}_1}^* = \| \cdot \|_{\mathcal{N}_\infty}$. Then it is plain to see that under the nuclear norm it holds that the Lipschitz constant of the objective function of (4.2) is $L = 1$. This follows since for any $Z, Y \in \mathbb{R}^{m \times n}$ we have:

$$\| \nabla f(Z) - \nabla f(Y) \|_{\mathcal{N}_\infty} \leq \| \nabla f(Z) - \nabla f(Y) \|_{\mathcal{N}_2} = \| (Z - X)_\Omega - (Y - X)_\Omega \|_F$$

$$\leq \| (Z - Y) \|_F = \| (Z - Y) \|_{\mathcal{N}_2} \leq \| (Z - Y) \|_{\mathcal{N}_1}.$$ 

Since the feasible region of (4.2) is $S = \mathcal{B} := \mathcal{B}_{\mathcal{N}_1}(0, \delta)$ it follows that the diameter of $S$ is $D = 2\delta$. Let us use the superscript $Z^k$ to denote the $k^{th}$ iterate of the algorithm, to avoid confusion with the subscript notation $Z_{ij}$ for indices of the $(i,j)^{th}$ component of $Z$. 

101
4.3.1 Characterization of faces of the nuclear norm ball

In order to implement Algorithm 4.3 we need to characterize and work with the minimal face of $\mathcal{B} = \mathcal{B}_{\mathcal{N}_1}(0, \delta)$ containing a given point. Let $\bar{Z} \in \mathcal{B}$ be given. The minimal face of $\mathcal{B}$ containing $\bar{Z}$ is formally notated as $\mathcal{F}_\mathcal{B}(\bar{Z})$. We have the following characterization of $\mathcal{F}_\mathcal{B}(\bar{Z})$ due to So [91]:

**Theorem 4.3. (So [91])** Let $\bar{Z} \in \mathcal{B}$ have thin SVD $\bar{Z} = UDV^T$ and let $r = \text{rank}(\bar{Z})$. Let $\mathcal{F}_\mathcal{B}(\bar{Z})$ denote the minimal face of $\mathcal{B}$ containing $\bar{Z}$. If $\sum_{j=1}^r \sigma_j = \delta$, then $\bar{Z} \in \partial \mathcal{B}$ and it holds that:

$$\mathcal{F}_\mathcal{B}(\bar{Z}) = \{Z \in \mathbb{R}^{m \times n} : Z = U M V^T \text{ for some } M \in \mathbb{S}^{r \times r}, \ M \succeq 0, \ I \cdot M = \delta\},$$

and $\dim(\mathcal{F}_\mathcal{B}(\bar{Z})) = r(r + 1)/2 - 1$. Otherwise $\sum_{j=1}^r \sigma_j < \delta$ and it holds that $\mathcal{F}_\mathcal{B}(\bar{Z}) = \mathcal{B}$ and $\dim(\mathcal{F}_\mathcal{B}(\bar{Z})) = \dim(\mathcal{B}) = m \times n$. 

Theorem 4.3 above is a reformulation of Theorem 3 of So [91], as the latter pertains to square matrices ($m = n$) and also does not explicitly treat the minimal faces containing a given point, but is a trivial extension of So’s theorem.

Theorem 4.3 explicitly characterizes the correspondence between the faces of the nuclear norm ball and low-rank matrices on its boundary. Note from Theorem 4.3 that if $\bar{Z} \in \partial \mathcal{B}$ and $r = \text{rank}(\bar{Z})$, then $\mathcal{F}_\mathcal{B}(\bar{Z})$ is a linear transformation of the $r \times r$ spectrahedron $\mathcal{S}^r_\delta := \{M \in \mathbb{S}^{r \times r} : M \succeq 0, \ I \cdot M = \delta\}$. This property will be most useful as it will make it very easy to compute in-face directions, especially when $r$ is relatively small, as we will see in Section 4.3.3 and Section 4.3.4.
4.3.2 Linear optimization subproblem solution for regular Frank-Wolfe step

In Step (3c.) of Algorithm 4.3 we need solve a linear optimization problem. Here we show how this can be done efficiently. We need to compute:

\[ \tilde{Z}^k \leftarrow \arg \min_{Z \in \mathcal{B}_{N^1(0, \delta)}} \nabla f(Z^k) \cdot Z. \]  

(4.18)

Then an optimal solution \( \tilde{Z}^k \) is readily seen to be:

\[ \tilde{Z}^k \leftarrow -\delta u_k v_k^T \]

(4.19)

where \( u_k \) and \( v_k \) denote the left and right singular vectors, respectively, of the matrix \( \nabla f(Z^k) \) corresponding to the largest singular value of \( \nabla f(Z^k) \). Therefore computing \( \tilde{Z}^k \) in Step (3c.) is relatively easy so long as the computation of the largest singular value of \( \nabla f(Z^k) \) and associated left and right eigenvalues thereof are easy to accurately compute. If \( |\Omega| \) is relatively small, then there are practically efficient methods (such as power iterations) that can effectively leverage the sparsity of \( \nabla f(Z^k) \).

4.3.3 Strategies and computation of the in-face direction \( D^k \)

Let \( D^k \) denote the in-face direction computed in Step (2.) of Algorithm 4.3. As suggested in Section 4.2.2, we present and discuss two different strategies for generating a suitable \( D^k \), namely (i) using an away-step approach (4.13), and (ii) directly solving for an optimal objective function solution over the low-dimensional face \( \mathcal{F}_B(Z^k) \) (4.14). In either case, computing \( D^k \) requires working with the thin SVD of \( Z^k \), which characterizes \( \mathcal{F}_B(Z^k) \) as stated in Theorem 4.3. Of course, the thin SVD of \( Z^k \) can be recomputed at every iteration, but this is generally very inefficient. As we expand upon in Section 4.3.6 the thin SVD of \( Z^{k+1} \) can be efficiently updated from the thin SVD of \( Z^k \) by utilizing the structure of the
regular Frank-Wolfe and in-face directions. For now, we simply assume that we have access to the thin SVD of $Z^k$ at the start of iteration $k$.

**Away-step Strategy.** Here we choose $D^k$ by setting $D^k \leftarrow Z^k - \hat{Z}^k$ where $\hat{Z}^k$ is the solution of the linear optimization maximization problem over the current minimal face, as in Step (2.) of the away-step algorithm (Algorithm 4.2). We compute the “away-step point” $\hat{Z}^k$ by computing:

\[ \hat{Z}^k \leftarrow \arg \max_{Z \in \mathcal{F}_B(Z^k)} \nabla f(Z^k) \cdot Z, \quad (4.20) \]

and set $D^k \leftarrow Z^k - \hat{Z}^k$. To see how to solve (4.20) efficiently, we consider two cases, namely when $Z^k \in \text{int}(\mathcal{B})$ and when $Z^k \in \partial(\mathcal{B})$. In the case when $Z^k \in \text{int}(\mathcal{B})$, then $\mathcal{F}_B(Z^k) = \mathcal{B}$ and the optimal solution in (4.20) is just the negative of the solution of (4.19), namely $\hat{Z}^k = \delta u_k v_k^T$.

In the case when $Z^k \in \partial(\mathcal{B})$, rank($Z^k$) = $r$, and $Z^k$ has thin SVD $Z^k = UDV^T$, we use the characterization of $\mathcal{F}_B(Z^k)$ in Theorem 4.3 to reformulate (4.20) as:

\[ \hat{Z}^k \leftarrow U \hat{M}^k V^T \quad \text{where} \quad \hat{M}^k \leftarrow \arg \max_{M \in \tilde{\mathcal{S}}_r^k} G^k \cdot M, \quad (4.21) \]

and where $G^k := \frac{1}{2}(V^T \nabla f(Z^k)U + U^T \nabla f(Z^k)V)$ so that $\nabla f(Z^k) \cdot U MV^T = G^k \cdot M$ for all $M \in \tilde{\mathcal{S}}_r^k$. An optimal solution to the subproblem in (4.21) is readily seen to be

\[ \hat{M}^k \leftarrow \delta u_k u_k^T \quad (4.22) \]

where $u_k$ is the normalized eigenvector corresponding to the largest eigenvector of the $r \times r$ symmetric matrix $G^k$. Therefore computing $\hat{Z}^k$ in (4.20) is relatively easy so long as the computation of the largest eigenvalue of $G^k$ and associated eigenvector thereof are easy to accurately compute. Furthermore, note that $\hat{Z}^k = U \hat{M}^k V^T = \delta U u_k u_k^T V^T$ is a rank-one matrix.

The above computational steps require the thin SVD of $Z^k$ as well as being able to efficiently compute the largest eigenvalue/eigenvector pair of $G^k$. Efficient computational
strategies for managing the thin SVD of \( Z^k \) are described in Section \[4.3.6\]. We compute the largest eigenvalue/eigenvector pair of \( G^k \) by either direct factorization of the \( r \times r \) matrix \( G^k \), or by power-method approximation, depending on the value of \( r \).

The development of the in-face Frank-Wolfe step strategy (4.15) in this case is quite similar. Indeed, we simply replace the maximization in (4.21) with a minimization, which corresponds to a smallest eigenvalue computation, and set \( D^k \) accordingly.

**Direct Solution on the Minimal Face.** In this strategy we use the alternating version of Algorithm 4.3 described at the end of Section 4.2.2 and we choose \( D^k \) by setting \( D^k \leftarrow \bar{Z}^k - Z^k \) where \( \bar{Z}^k \) optimizes (exactly or perhaps only approximately) the original objective function \( f(Z) \) over the current minimal face, under the assumption that such optimization can be done efficiently and accurately. Indeed, when \( Z^k \in \text{int}(B) \), then we default to the previous away-step strategy since optimizing over the minimal face is identical to the original problem (4.2). Otherwise, when \( Z^k = UDV^T \in \partial(B) \) we again use the characterization of \( \mathcal{F}_B(Z^k) \) in Theorem 4.3 to compute \( \bar{Z}^k \) as:

\[
\bar{Z}^k \leftarrow U \bar{M}^k V^T \quad \text{where} \quad \bar{M}^k \leftarrow \arg\min_{M \in \mathcal{S}_r^{+}} f(U MV^T). \tag{4.23}
\]

Of course, it is only sensible to consider this strategy when \( Z^k \) has low rank, for otherwise (4.23) is nearly as difficult to solve as the original problem (4.2) whose solution we seek to approximate using the In-face Extended Frank-Wolfe Method. Since \( f(\cdot) \) is a convex quadratic function, it follows that the subproblem in (4.23) is solvable as a semidefinite/second-order conic optimization problem and thus conic interior-point methods may be practical. Alternately, one can approximately solve (4.23) by taking a number of steps of any suitably effective method, such as a proximal/accelerated first-order method [98] (or even the Frank-Wolfe Method itself).
4.3.4 Computing the maximal step-size $\alpha^\text{stop}_k$ in Step (2.)

Here we describe how to efficiently compute the maximal step-size $\alpha^\text{stop}_k$ in Step (2.) of Algorithm 4.3, which is determined as:

$$
\alpha^\text{stop}_k \leftarrow \arg \max_{\alpha} \{\alpha : Z^k + \alpha D^k \in F_B(Z^k)\} .
$$

(4.24)

Let us first assume that $Z^k \in \partial(B)$. We will utilize the SVD of the current iterate $Z^k = UDV^T$. Using either the away-step strategy or the direct solution strategy for determining the in-face direction $D^k$ in Section 4.3.3, it is simple to write $D^k = U\Delta V^T$ for an easily given matrix $\Delta \in \mathbb{S}^{r \times r}$. Since $Z^k \in \partial(B)$ and $Z^k + D^k \in F_B(Z^k)$ it holds that $I \cdot D = \delta$ and hence $I \cdot \Delta = 0$. Using the characterization of $F_B(Z^k)$ in Theorem 4.3 it follows that (4.24) can be reformulated as:

$$
\alpha^\text{stop}_k \leftarrow \arg \max_{\alpha,M} \{\alpha : UDV^T + \alpha U\Delta V^T = UMV^T, \; M \in \tilde{S}_\delta\} = \arg \max_{\alpha} \{\alpha : D + \alpha \Delta \succeq 0\} .
$$

(4.25)

In the case when $D^k$ is chosen using the away-step approach, we have from (4.21) and (4.22) that $\Delta := D - \delta u_k u_k^T$ satisfies $D^k = Z^k - \hat{Z}^k = U\Delta V^T$. In this case the maximum $\alpha$ satisfying (4.25) is easily seen to be $\alpha^\text{stop}_k := (\delta u_k^T D^{-1} u_k - 1)^{-1}$. When $D^k$ is chosen by some other method, such as the direct solution method on the minimal face, the optimal solution of (4.25) is seen to be $\alpha^\text{stop}_k := -\left[\lambda_{\min}\left(D^{-\frac{1}{2}} \Delta D^{-\frac{1}{2}}\right)\right]^{-1}$.

In the case when $Z^k \in \text{int}(B)$, then (4.24) can be written as $\alpha^\text{stop}_k \leftarrow \arg \max\{\alpha : \|Z^k + \alpha D^k\|_{N1} \leq \delta\}$, and we use binary search to approximately determine $\alpha^\text{stop}_k$.

4.3.5 Initial values, step-sizes, and computational guarantees

We initialize Algorithm 4.3 by setting

$$
Z^0 \leftarrow -\delta u_0 v_0^T
$$

(4.26)

106
where $u_0$ and $v_0$ denote the left and right singular vectors, respectively, of the matrix $\nabla f(0)$ corresponding to the largest singular value of $\nabla f(0)$. This initialization corresponds to a “full step” iteration of the Frank-Wolfe Method initialized at 0 and conveniently satisfies $\text{rank}(Z^0) = 1$ and $Z^0 \in \partial \mathcal{B}$. We initialize the lower bound as $B_{-1} \leftarrow \max \{f(0) + \nabla f(0) \cdot Z^0, 0\}$, where the first term inside the max corresponds to the lower bound generated when computing $Z^0$ and the second term is a valid lower bound since $f^* \geq 0$. Moreover, this initialization has a provably good optimality gap, namely $f(Z^0) \leq B_{-1} + 2\delta^2 \leq f^* + 2\delta^2$, which follows from Proposition 2.1 of Chapter 2.

Because $f(\cdot)$ is a convex quadratic function, we use an exact line-search to determine $\bar{\beta}_k$ and $\bar{\alpha}_k$ in Steps (2.) and (3c.), respectively, since the line-search reduces to a simple formula in this case.

Utilizing the bound on the optimality gap for $Z^0$, and recalling that $L = 1$ and $D = 2\delta$, we have from Theorem 4.2 that the computational guarantee for Algorithm 4.3 is:

$$f(Z^k) - B_k \leq f(Z^k) - f^* \leq \frac{1}{f(Z^0) - B_0} \left( \frac{\gamma_1 N_k^a}{8\delta^2} + \frac{\gamma_2 N_k^b}{8\delta^2} + \frac{N_k^c}{8\delta^2} \right) \leq \frac{8\delta^2}{4 + \gamma_1 N_k^a + \gamma_2 N_k^b + N_k^c}.$$ 

4.3.6 Efficiently Updating the Thin SVD of $Z^k$

At each iteration of Algorithm 4.3 we need to access two objects related to the current iterate $Z^k$: (i) the current gradient $\nabla f(Z^k) = (Z^k - X)_\Omega$ (for solving the regular Frank-Wolfe linear optimization subproblem and for computing in-face directions), and (ii) the thin SVD $Z^k = UDV^T$ (for computing in-face directions). For large-scale matrix completion problems, it can be very burdensome to store and access all $mn$ entries of the (typically dense) matrix $Z^k$. On the other hand, if $r := \text{rank}(Z^k)$ is relatively small, then storing the thin SVD of $Z^k$ requires only keeping track of $mr + r + nr$ entries. Thus, when implementing Algorithm 4.3 as discussed above, instead of storing the entire matrix $Z^k$, we store in memory the thin SVD of $Z^k$ (i.e., the matrices $U$, $V$, and $D$), which we initialize from (4.26) and efficiently update as follows. Let $D^k$ denote the direction chosen by Algorithm 4.3 at iteration $k \geq 0,$
which is appropriately scaled so that \( Z^{k+1} = Z^k + D^k \). To compute the thin SVD of \( Z^{k+1} \),
given the thin SVD of \( Z^k \), we consider the cases of regular Frank-Wolfe directions and in-
face directions separately. In the case of a regular Frank-Wolfe direction, we have that
\[ D^k = \bar{\alpha}_k (-\delta u_k v_k^T - Z^k) \]
and therefore:
\[
Z^{k+1} = Z^k + \bar{\alpha}_k (-\delta u_k v_k^T - Z^k) = (1 - \bar{\alpha}_k)Z^k - \bar{\alpha}_k \delta u_k v_k^T = (1 - \bar{\alpha}_k)UDV^T - \bar{\alpha}_k \delta u_k v_k^T.
\]
Thus, given the thin SVD of \( Z^k \), computing the thin SVD of \( Z^{k+1} \) is a scaling plus a rank-1
update of the thin SVD, which can be performed very efficiently in terms of both computation
time and memory requirements, see [11]. An analogous argument applies to the away-step
strategy when \( Z^k \in \text{int}(B) \). Otherwise, when \( Z^k \in \partial(B) \), recall that we can write any in-face
direction as \( D^k = U \Delta V^T \) for an easily given matrix \( \Delta \in S_r \times r \). Thus we have:
\[
Z^{k+1} = Z^k + D^k = UDV^T + U \Delta V^T = U(D + \Delta)V^T.
\]
Recall from [4.25] that we have \( D + \Delta \succeq 0 \). Therefore, to compute the thin SVD of \( Z^{k+1} \),
we first compute an eigendecomposition of the \( r \times r \) symmetric positive semidefinite matrix
\( D + \Delta \), so that \( D + \Delta = RSR^T \) where \( R \) is orthonormal and \( S \) is diagonal with nonnegative
entries, and then update the thin SVD of \( Z^{k+1} \) as \( Z^{k+1} = (UR)S(VR)^T \).

To compute the current gradient from the thin SVD of \( Z^k \), note that \( \nabla f(Z^k) = (Z^k - X)_\Omega \)
is a sparse matrix that is 0 everywhere except on the \( \Omega \) entries; thus computing \( \nabla f(Z^k) \) from
the thin SVD of \( Z^k \) requires performing \( |\Omega| \) length \( r \) inner product calculations. As compared
to storing the entire matrix \( Z^k \), our implementation requires a modest amount of extra work
to compute \( \nabla f(Z^k) \), but the cost of this extra work is far outweighed by the benefits of not
storing the entire matrix \( Z^k \). Alternatively, it is slightly more efficient to update only the \( \Omega \)
entries of \( Z^k \) at each iteration (separately from the thin SVD of \( Z^k \)) and to use these entries
to compute \( \nabla f(Z^k) \).
4.3.7 Rank accounting

As developed throughout this Section, the computational effort required at iteration $k$ of Algorithm 4.3 depends very much on $\text{rank}(Z^k)$ for tasks such as computing the in-face direction $D^k$ (using either the away-step approach or direct solution on the minimal face), computing the maximal step-size $\alpha_k^{\text{stop}}$ in Step (3.), and updating the thin SVD of $Z^k$. Herein we examine how $\text{rank}(Z^k)$ can change over the course of the algorithm. At any given iteration $i$, there are four relevant possibilities for how the next iterate is chosen:

(a) The current iterate $Z^i$ lies on the boundary of $\mathcal{B}$, and the next iterate $Z^{i+1}$ is chosen according to the criteria in Step (3a.).

(b) The current iterate $Z^i$ lies on the boundary of $\mathcal{B}$, and the next iterate $Z^{i+1}$ is chosen according to the criteria in Step (3b.).

(c) The next iterate $Z^{i+1}$ is chosen according to the criteria in Step (3c.).

(d) The current iterate $Z^i$ lies in the interior of $\mathcal{B}$, and the next iterate is chosen according to either the criteria in Step (3a.) or Step (3b.).

The following proposition presents bounds on rank of $Z^k$.

**Proposition 4.2.** Let $N^a_k$, $N^b_k$, $N^c_k$, and $N^d_k$ denote the number of times within the first $k$ iterations that the above conditions (a), (b), (c), and (d) hold, respectively. Then

$$\text{rank}(Z^k) \leq k + 1 - 2N^a_k - N^b_k.$$  \hfill (4.27)

**Proof.** Using the choice of the initial point $Z^0$ developed in Section 4.3.5, it holds that $\text{rank}(Z^0) = 1$. Now consider the $i$th iterate value $Z^i$ for $i = 1, \ldots, k$. If condition (a) holds, then $Z^{i+1}$ lies on a lower-dimensional face of $\mathcal{F}_\mathcal{B}(Z^i) \subset \mathcal{B}$, whence from Theorem 4.3 it follows that $\text{rank}(Z^{i+1}) \leq \text{rank}(Z^i) - 1$. If instead condition (b) holds, then $\text{rank}(Z^{i+1}) = \text{rank}(Z^i)$ since $Z^{i+1}$ lies in the relative interior of $\mathcal{F}_\mathcal{B}(Z^i) \subset \mathcal{B}$. Finally, in either case that condition
(c) or condition (d) holds, it follows from (4.19) that $\tilde{Z}_i$ is a rank-one matrix and thus it holds that $\text{rank}(Z^{i+1}) \leq \text{rank}(Z^i) + 1$. Since the four cases above are exhaustive, we have $k = N_k^a + N_k^b + N_k^c + N_k^d$ and we obtain $\text{rank}(Z^k) \leq 1 + N_k^c + N_k^d - N_k^a = k + 1 - 2N_k^a - N_k^b$. □

4.4 Computational Experiments and Results

In this section we present computational results of experiments wherein we apply different versions of the In-Face Extended Frank-Wolfe Method to the nuclear norm regularized matrix completion problem (4.2). Our main focus is on simulated problem instances, but we also present results for the MovieLens10M dataset. The simulated instances were generated according to the model $X := w_1 U V^T + w_2 \mathcal{E}$, where the entries of $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$ and $\mathcal{E} \in \mathbb{R}^{m \times n}$ are all i.i.d. standard normal random variables, and the scalar parameters $w_1, w_2$ control the signal to noise ratio (SNR), namely $w_1 := 1/\|UV^T\|_F$ and $w_2 := 1/(\text{SNR}\|\mathcal{E}\|_F)$. The set of observed entries $\Omega$ was determined using uniform random sampling of entries with probability $\rho$, where $\rho$ is the target fraction of observed entries. The objective function $f(\cdot)$ values were normalized so that $f(0) = .5$ and we chose the regularization parameter $\delta$ using a cross-validation-like procedure based on an efficient path algorithm variant of Algorithm 4.1.

We study several versions of the In-Face Extended Frank-Wolfe Method (Algorithm 4.3) based on different strategies for setting the parameters $\gamma_1, \gamma_2$, which we compare to the regular Frank-Wolfe Method (Algorithm 4.1) and the away-step method (Algorithm 4.2). We also study the atomic version of the away-step method and the “fully corrective” variant of Frank-Wolfe [49, 54, 60] – both of which reformulate (4.2) in the atomic format of the right-side of (4.12). Finally, we also include comparisons with CoGENT, the “forward-backward” variant of the Frank-Wolfe Method studied in [82]. All methods are implemented according to the

---

1 All computations were performed using MATLAB R2015b on a 3 GHz Intel Core i7 MacBook Pro laptop.

2 Specifically, we apply a version of Algorithm 4.1 that periodically increases the value of $\delta$, utilizing the previously found solution as a warm-start at the new value of $\delta$. We maintain a holdout set $\Omega'$ and ultimately select the value of $\delta$ that minimizes the least-squares error on this set.
details presented in Section 4.3 except for CoGEnT. We focus on the following ten versions of methods with names given below and where “IF-” stands for In-Face:

- **FRANK-WOLFE** – Algorithm 4.1

- **IF-(1,1)** – Algorithm 4.3 using an away-step strategy, with \( \gamma_1 = 1, \gamma_2 = 1 \)

- **IF-(0,1)** – Algorithm 4.3 using an away-step strategy, with \( \gamma_1 = 0, \gamma_2 = 1 \)

- **IF-(0,\infty)** – Algorithm 4.3 using an away-step strategy, with \( \gamma_1 = 0, \gamma_2 = \infty \). This corresponds to always moving to the relative boundary of the minimal face containing \( Z_k \) (thereby reducing the rank of \( Z^{k+1} \)) as long as the objective function value does not increase, while never moving partially within the current face.

- **IF-OPTIMIZATION** – The simplified version of Algorithm 4.3 with full in-face optimization as described at the end of Section 4.2.2. The in-face optimization subproblem is (approximately) solved using the proximal gradient method with matrix entropy prox function.

- **IF-RANK-STRATEGY** – Algorithm 4.3 with the away-step strategy and with \( \gamma_1 \) and \( \gamma_2 \) set dynamically as follows: we initially set \( \gamma_1 = \gamma_2 = \infty \), and then reset \( \gamma_1 = \gamma_2 = 1 \) after we observe five consecutive iterations where rank(\( Z^k \)) does not increase. This version can be interpreted as a two-phase method where we run Algorithm 4.1 until we observe that rank(\( Z^k \)) begins to “stall,” at which point we switch to Algorithm 4.3 with \( \gamma_1 = \gamma_2 = 1 \).

- **FW-AWAY-NATURAL** – Algorithm 4.2

- **FW-AWAY-ATOMIC** – Algorithm 4.2 applied to the atomic reformulation of (4.2) using the right-side of (4.12) [4, 59, 60, 79].

---

3The MATLAB code for CoGEnT was obtained from [81].
• **FW-FULLY-CORRECTIVE** – The “fully corrective” variant of Frank-Wolfe [49, 54, 60], which works with the atomic reformulation of (4.2) and, at each iteration, fully optimizes the objective function of (4.2) over the convex hull of the current set of active atoms. The “correction” optimization subproblem is (approximately) solved using the proximal gradient method with entropy prox function over the standard unit simplex.

• **CoGENT** – The matrix completion variant of the CoGENT Method studied in [82]. This variant uses singular value thresholding for the truncation/backward step – at each iteration, the algorithm computes the SVD of the current iterate and truncates small singular values to zero. This step is followed by an enhancement step that optimizes the objective function over the weights in the SVD. The singular value thresholding parameter is set to $0.05 \cdot \delta$ and the algorithmic parameter $\eta$ is set to 0.5.

Tables 4.1, 4.2, and 4.3 present our aggregate computational results. Before discussing these in detail, it is useful to first study Figure 4-2 which shows the behavior of each method in terms of ranks of iterates\(^4\) (left panel) and relative optimality gap (right panel) as a function of run time, for a particular (and very typical) simulated instance. Examining the rank plots in the left panel, we see that the evolution of rank($Z^k$) is as follows: the four methods **IF-(1,1)**, **IF-(0,1)**, **IF-(0,\(\infty\))**, and **FW-AWAY-NATURAL** all quickly attain rank($Z^k$) $\approx 37$ (the apparent rank of the optimum) and then stay at or near this rank from then on. In contrast, the four methods **FRANK-WOLFE**, **IF-RANK-STRATEGY** **IF-OPTIMIZATION** and **FW-AWAY-ATOMIC** all grow rank($Z^k$) approximately linearly during the early stages (due to a larger percentage of regular Frank-Wolfe steps), and then reach a maximum value that can be an order of magnitude larger than the optimal rank before the rank starts to decrease. Once the rank starts to decrease, **IF-RANK-STRATEGY** and **IF-OPTIMIZATION** decrease rank($Z^k$) rather rapidly, whereas **FRANK-WOLFE** and **FW-AWAY-ATOMIC** decrease rank($Z^k$) painfully slowly.

\(^4\)The rank of a matrix is computed as the number of singular values larger than $10^{-6}$. The rank-1 SVD computation for equation (4.18) is performed using the Matlab function **eigs**.

The right panel of Figure 4-2 shows the relative optimality gaps of the methods. It
is noteworthy that two methods – \text{IF-OPTIMIZATION} and \text{IF-RANK-STRATEGY} – achieve very rapid progress during their early stages, a point that we will soon revisit. However, all methods exhibit eventual slow convergence rates which is in line with the $O(1/k)$ theoretical convergence bound.

Let us now synthesize the two panels of Figure 4-2. The four methods \text{FRANK-WOLFE}, \text{IF-RANK-STRATEGY}, \text{IF-OPTIMIZATION} and \text{FW-AWAY-ATOMIC} each go through two phases: in the first phase each constructs a “high information” (high rank) solution (by taking mostly regular Frank-Wolfe steps), followed by a second phase where the solution is “refined” by lowering the rank while further optimizing the objective function (by taking proportionally more away-steps). \text{FRANK-WOLFE} and \text{FW-AWAY-ATOMIC} build up to very high information but their build-down is sorely ineffective both in terms of ranks and objective function values. \text{IF-RANK-STRATEGY} is extremely effective at the refinement phase, and \text{IF-OPTIMIZATION} is less effective in terms of rank reduction but still more so than the other methods except of course for \text{IF-RANK-STRATEGY}. The other four methods, namely \text{IF-(1,1)} \text{IF-(0,1)} \text{IF-(0,$\infty$)} and \text{FW-AWAY-NATURAL} all rarely exceed rank 37, as they spend a very high proportion of their effort on away-steps. Of these four methods, \text{IF-(0,$\infty$)} tends to perform best in terms of objective function values, as will be seen shortly in Tables 4.1 and 4.2. Last of all, we point out that for very large-scale problems storing the SVD of a high-rank matrix may become burdensome (over and above the computational cost for computing in-face directions on high-dimensional faces); thus it is important that the maximum rank of the iterates be kept small. In this regard Figure 4-2 indicates that excessive memory may arise for \text{FRANK-WOLFE}, \text{IF-RANK-STRATEGY}, \text{FW-AWAY-ATOMIC} and possibly \text{IF-OPTIMIZATION}.

Table 4.1 presents computational results for three different types of small-scale examples, averaged over 25 sample instances generated and run for each type. Note that the run time, final rank, and maximum ranks reported in Table 4.1 are in sync with the patterns observed in Figure 4-2. \text{IF-RANK-STRATEGY} exhibits the best run times, followed by \text{IF-OPTIMIZATION} and then by \text{IF-(0,$\infty$)}, all of which significantly outperform \text{FRANK-WOLFE} \text{IF-RANK-STRATEGY} \text{FW-AWAY-ATOMIC} and possibly
Figure 4-2: Figure showing plots of rank and relative optimality gap (log-scale) versus run time for different methods/strategies, for a single randomly generated problem instance with $m = 2000$, $n = 2500$, $\rho = 0.01$, $r = 10$, SNR = 4, and $\delta = 8.01$. This problem has a (very nearly) optimal solution with rank 37.

Natural, and FW-AWAY-ATOMIC. Furthermore, IF-OPTIMIZATION and IF-(0,\infty) have relatively low values of the maximum rank (unlike IF-RANK-STRATEGY), while not giving up too much in terms of run time relative to IF-RANK-STRATEGY. Note that FW-AWAY-ATOMIC and FW-FULLY-CORRECTIVE are dramatically ineffective at delivering low rank solutions, which is undoubtedly related to the fact that the faces of the atomic representation are simply too small to be effective – see Figure 4-1 and the discussion at the end of Section 4.2. Note that our best In-Face methods – IF-(0,\infty), IF-OPTIMIZATION, and IF-RANK-STRATEGY – significantly beat both FW-FULLY-CORRECTIVE and CoGEnT in both run time and final rank; this fact may be attributed to several factors including the considerable time required to solve the correction/enhancement subproblems when the number of atoms is large.

Table 4.2 presents computational results for eight individual medium-large scale examples. Here we see mostly similar performance of the different methods as was seen for the small-scale examples in Table 4.1. IF-RANK-STRATEGY, IF-(0,\infty) and IF-OPTIMIZATION
deliver the best balance between final rank, maximum rank, and run time, with perhaps \textbf{IF-}(0,\infty) delivering consistently lower rank solutions albeit with higher run times. We note that for these instances \textbf{IF-RANK-STRATEGY} does not consistently deliver low rank solutions, which is due to the extra time it takes before the second phase (“refinement”) of the method commences. We did not include results for \textbf{COGENT} as there was insufficient memory to run \textbf{COGENT} on any of these instances. Similar to observations from Table \ref{tab:4.1}, our best In-Face methods – \textbf{IF-}(0,\infty), \textbf{IF-OPTIMIZATION} and \textbf{IF-RANK-STRATEGY} – significantly beat \textbf{FW-FULLY-CORRECTIVE} in both run time and final rank.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
Data & Metric & Regular FW & In-Face Extended FW (IF-\ldots) & Fully Corrective FW & CoGEnT \\
& & \multicolumn{3}{c|}{\begin{array}{c}
\text{In-Face Rank} \\
\text{Opt.} \\
\text{Strategy}
\end{array}} & \multicolumn{2}{c|}{\begin{array}{c}
\text{Away Steps} \\
\text{Natural} \\
\text{Atomic}
\end{array}} & \multicolumn{2}{c|}{\begin{array}{c}
\text{CoGEnT}
\end{array}} \\
\hline
\multirow{2}{*}{\begin{array}{c}
m = 200, \; n = 400, \; \rho = 0.10
\end{array}} & Time (secs) & 29.51 & 22.86 & 23.07 & 7.89 & \textbf{2.34} & \textbf{2.30} & 14.71 & 6.21 & 8.76 & 20.85 \\
& \multirow{2}{*}{\begin{array}{c}rak{r} = 10, \; \text{SNR} = 5, \; \delta_{\text{avg}} = 3.75
\end{array}} & \text{Final Rank} & 118.68 & 16.36 & 16.36 & \textbf{16.44} & 29.32 & 28.20 & 16.72 & 119.00 & 92.84 & 79.96 \\
& & \text{Maximum Rank} & 146.48 & 19.04 & 17.28 & \textbf{17.56} & 32.08 & 145.20 & 18.04 & 121.96 & 991.60* & ** \\
\hline
\multirow{2}{*}{\begin{array}{c}
m = 200, \; n = 400, \; \rho = 0.20
\end{array}} & Time (secs) & 115.75 & 153.42 & 150.89 & 27.60 & \textbf{20.62} & \textbf{3.48} & 50.52 & 24.52 & 196.29 & 65.88 \\
& \multirow{2}{*}{\begin{array}{c}rak{r} = 15, \; \text{SNR} = 4, \; \delta_{\text{avg}} = 3.82
\end{array}} & \text{Final Rank} & 96.44 & 16.16 & 16.12 & \textbf{16.52} & 19.88 & 21.24 & 16.68 & 106.60 & 107.04 & 93.40 \\
& & \text{Maximum Rank} & 156.52 & 26.72 & 17.96 & \textbf{17.80} & 31.48 & 160.36 & 18.84 & 106.80 & 1812.92* & ** \\
\hline
\multirow{2}{*}{\begin{array}{c}
m = 200, \; n = 400, \; \rho = 0.30
\end{array}} & Time (secs) & 171.23 & 198.96 & 202.01 & 35.93 & 31.67 & \textbf{5.04} & 66.22 & 67.72 & >381.91 & 93.93 \\
& \multirow{2}{*}{\begin{array}{c}rak{r} = 20, \; \text{SNR} = 3, \; \delta_{\text{avg}} = 3.63
\end{array}} & \text{Final Rank} & 91.80 & 20.08 & 20.08 & \textbf{20.60} & \textbf{21.72} & 25.56 & 20.44 & 94.64 & 113.84 & 104.60 \\
& & \text{Maximum Rank} & 162.24 & 25.80 & 22.04 & \textbf{21.96} & 33.36 & 168.72 & 22.16 & 95.08 & 1609.40* & ** \\
\hline
\end{tabular}
\caption{Small-Scale Examples (25 samples per example)}
\end{table}

Table \ref{tab:4.1} shows computational tests on a large-scale real dataset, namely the MovieLens10M dataset, with \(m = 69878, \; n = 10677, \; |\Omega| = 10^7\) (with sparsity approximately 1.3\%), and \(\delta = 2.59\). We only tested \textbf{IF-}(0,\infty) (and benchmarked against \textbf{FRANK-WOLFE} and \textbf{FW-AWAY-NATURAL}) since \textbf{IF-}(0,\infty) appears to be very promising for large-scale instances due to its ability to maintain relatively low-rank iterates throughout, while also per-

\footnote{The \textbf{COGENT} code directly works with the variables \(Z_{ij}\), and thus has large memory requirements. A more efficient implementation of \textbf{COGENT} may be able to run on the instances in Table \ref{tab:4.2} and may also have better performance on the instances in Table \ref{tab:4.1}.}
Terms of both run time and final rank. Also note that in terms of final rank, while it is essentially the same as Wolfe and Frank-Wolfe well as run time to achieve the target optimality gap. Note that IF-(0,∞) generally dominates both Frank-Wolfe and FW-Away-Natural in terms of run time, and dominates Frank-Wolfe in terms of final rank, while it is essentially the same as FW-Away-Natural on the final rank. Also note that FW-Away-Natural generally dominates Frank-Wolfe in terms of both run time and final rank.

Table 4.2: Table reports the time required for each method to reach a relative optimality gap of $10^{-2.5}$, the rank of the corresponding final solution, and the maximum rank of the iterates therein. Numbers highlighted in boldface indicate the methods that perform well with regard to each criteria, while not performing poorly on run time. Numbers highlighted in boldface indicate the methods that perform well with regard to each criteria, while not performing poorly on run time.

*For this algorithm, we counted the maximum number of atoms instead of the maximum rank.

forming well in terms of run time. The results in Table 4.3 further reinforce the findings from Table 4.2 concerning the advantages of [IF-(0,∞)] both in terms of rank of the final iterate as well as run time to achieve the target optimality gap. Note that [IF-(0,∞)] dominates both Frank-Wolfe and FW-Away-Natural in terms of run time, and dominates Frank-Wolfe in terms of final rank, while it is essentially the same as FW-Away-Natural on the final rank. Also note that FW-Away-Natural generally dominates Frank-Wolfe in terms of both run time and final rank.

**Table 4.3:** CPU time and rank of final solutions for Frank-Wolfe, FW-Away-Natural, and IF-(0,∞) for different relative optimality gaps for the MovieLens10M dataset.

<table>
<thead>
<tr>
<th>Relative Optimality Gap</th>
<th>Frank-Wolfe</th>
<th>FW-Away-Natural</th>
<th>IF-(0,∞)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (mins)</td>
<td>Time (mins)</td>
<td>Time (mins)</td>
</tr>
<tr>
<td>10^{-1.5}</td>
<td>7.38</td>
<td>10.86</td>
<td>7.01</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>28.69</td>
<td>23.08</td>
<td>14.73</td>
</tr>
<tr>
<td>10^{-2.5}</td>
<td>69.53</td>
<td>34.78</td>
<td>22.80</td>
</tr>
<tr>
<td>10^{-2.25}</td>
<td>178.54</td>
<td>76.64</td>
<td>42.24</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>Rank</td>
<td>Rank</td>
</tr>
<tr>
<td>10^{-1.5}</td>
<td>103</td>
<td>52</td>
<td>44</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>315</td>
<td>87</td>
<td>79</td>
</tr>
<tr>
<td>10^{-2.5}</td>
<td>461</td>
<td>113</td>
<td>107</td>
</tr>
<tr>
<td>10^{-2.25}</td>
<td>454</td>
<td>141</td>
<td>138</td>
</tr>
</tbody>
</table>
We conclude our computational research with a diagnostic evaluation of the different types of iterations and associated CPU times of different methods. Table 4.4 presents a detailed breakdown of the types of iterations and other algorithmic diagnostics for different methods applied to the middle grouping of 25 small-scale examples of Table 4.1. Recall that there are four types of iterations that can arise in the In-Face Extended Frank-Wolfe method, namely types (a), (b), (c), and (d) as exposited in Section 4.3.7. These types naturally extend to FW-Away-Natural, FW-Away-Atomic and FW-Fully-Corrective, but not to CoGEnT, hence CoGEnT is not included in our evaluation. Rows 2-5 of Table 4.4 break down the iterations into the four types, and rows 7-9 report information on the CPU time spent on in-face directions and regular Frank-Wolfe directions. For methods that use standard away steps (IF-(1,1), IF-(0,1), IF-(0,∞), IF-Rank-Strategy, and FW-Away-Natural), most of the time is spent computing regular Frank-Wolfe directions. Indeed, as the bottom row of the table indicates, for four of these methods the average CPU time spent per in-face direction is a mere 2 – 6% of that spent computing regular Frank-Wolfe directions. IF-Rank-Strategy spends comparatively more time computing the in-face direction because the computational burden of the in-face direction scales with the ranks of the iterates. Also, IF-Optimization spends more time computing in-face directions because solving the proximal gradient algorithm is more expensive than elementary linear optimization. Furthermore, the atom-based methods (FW-Away-Atomic and FW-Fully-Corrective) spend more time computing in face directions because the computational burden scales with the number of atoms and the number of atoms becomes extremely large.

Row 6 of Table 4.4 reports the final rank and the bound on the final rank from Proposition 4.2. Very curiously, the bound from Proposition 4.2 is nearly tight for both IF-(0,∞) and FW-Away-Natural, whereas it is generally very loose otherwise. The tightness of the bounds for these two methods is due to the fact that the different steps taken are almost evenly split between regular Frank-Wolfe steps (type (c)), and steps of type (a) – iterations that go to the boundary of the current minimal face. The former almost always increases the rank by one, whereas the latter always decreases the rank by at least one.
### Summary Conclusions

In addition to its theoretical computational guarantees (Theorem 4.2, Proposition 4.1), the In-Face Extended Frank-Wolfe Method (in different versions) shows significant computational advantages in terms of delivering low rank and low run time to compute a target optimality gap. Especially for larger instances, $\text{IF-}(0, \infty)$ delivers very low rank solutions with reasonable run times. $\text{IF-RANK-STRATEGY}$ delivers the best run times, beating existing methods by a factor of 10 or more. And in the large-scale regime, $\text{IF-OPTIMIZATION}$ generally delivers both low rank and low run times simultaneously, and is usually competitive with the best methods on one or both of rank and run time.
Chapter 5

A New Perspective on Boosting in Linear Regression

5.1 Introduction

Boosting \([86, 37, 42, 87, 51]\) is an extremely successful and popular supervised learning technique that combines multiple “weak” learners into a more powerful “committee.” AdaBoost \([38, 87, 51]\), developed in the context of classification, is one of the earliest and most influential boosting algorithms. In our paper \([36]\), we analyze boosting algorithms in linear regression \([40, 42, 31]\) from the perspective of modern first-order methods in convex optimization. This perspective has two primary upshots: (i) it leads to first-ever computational guarantees for existing boosting algorithms, and (ii) it leads to new boosting algorithms with novel connections to the LASSO \([95]\).

Notation  We use the usual linear regression notation with model matrix \(X = [X_1, \ldots, X_p] \in \mathbb{R}^{n \times p}\), response vector \(y \in \mathbb{R}^{n \times 1}\), and regression coefficients \(\beta \in \mathbb{R}^p\). Each column of \(X\) corresponds to a particular feature or predictor variable, and each row corresponds to a particular observed sample. We assume herein that the features \(X_i\) have been centered to have zero
mean and unit \( \ell_2 \) norm, i.e., \( \| X_i \|_2 = 1 \) for \( i = 1, \ldots, p \), and \( y \) is also centered to have zero mean. For a regression coefficient vector \( \beta \), the predicted value of the response is given by \( X \beta \) and \( r = y - X \beta \) denotes the residuals. Let \( e_j \) denote the \( j \)-th unit vector in \( \mathbb{R}^p \) and let \( \| v \|_0 \) denote the number of nonzero coefficients in the vector \( v \). Denote the empirical least squares loss function by \( L_n(\beta) := \frac{1}{2n} \| y - X \beta \|_2^2 \), let \( L^* = \min_{\beta \in \mathbb{R}^p} L_n(\beta) \), and let \( \hat{\beta}_{\text{LS}} \) denote an arbitrary minimizer of \( L_n(\beta) \), i.e., \( \hat{\beta}_{\text{LS}} \in \arg \min_{\beta \in \mathbb{R}^p} L_n(\beta) \). Finally, let \( \lambda_{\text{pmin}}(X^T X) \) denote the smallest nonzero (and hence positive) eigenvalue of \( X^T X \).

**Boosting and Implicit Regularization** The first boosting algorithm we consider is the Incremental Forward Stagewise algorithm \([51, 31]\) presented below, which we refer to as FS\(_{\varepsilon}\).

**Algorithm:** Incremental Forward Stagewise Regression – FS\(_{\varepsilon}\)

1. Fix the learning rate \( \varepsilon > 0 \) and number of iterations \( M \).
2. Initialize at \( \hat{r}^0 = y \), \( \hat{\beta}^0 = 0 \), \( k = 0 \).
3. For \( 0 \leq k \leq M \) do the following:
   1. Compute: \( j_k \in \arg \max_{j \in \{1,\ldots,p\}} |(\hat{r}^k)^T X_j| \)
   2. \( \hat{\beta}_{j_k}^{k+1} \leftarrow \hat{\beta}_{j_k}^k + \varepsilon \text{ sgn}((\hat{r}^k)^T X_{j_k}) \) and \( \hat{\beta}_j^{k+1} \leftarrow \hat{\beta}_j^k \), \( j \neq j_k \)
   3. \( \hat{r}^{k+1} \leftarrow \hat{r}^k - \varepsilon \text{ sgn}((\hat{r}^k)^T X_{j_k}) X_{j_k} \).

At the \( k \)-th iteration, FS\(_{\varepsilon}\) chooses a column \( X_{j_k} \), corresponding to a particular feature that is the most correlated (in absolute value) with the current residuals and then updates the corresponding regression coefficient by an amount \( \varepsilon > 0 \), called the learning rate (or shrinkage factor).

A close cousin of FS\(_{\varepsilon}\) is the least squares boosting algorithm, or LS-BOOST\((\varepsilon)\), proposed in \([40]\). The LS-BOOST\((\varepsilon)\) algorithm is identical to FS\(_{\varepsilon}\) except that LS-BOOST\((\varepsilon)\) changes the amount by which the selected coefficient is updated at each iteration – at the \( k \)-th iteration, LS-BOOST\((\varepsilon)\) updates:
\[
\hat{\beta}_{j_k}^{k+1} \leftarrow \hat{\beta}_{j_k}^k + \varepsilon \left( \langle \hat{r}^k \rangle^T X_{j_k} \right) \quad \text{and} \quad \hat{\beta}_{j}^{k+1} \leftarrow \hat{\beta}_{j}^k, \ j \neq j_k
\]

\[
\hat{r}^{k+1} \leftarrow \hat{r}^k - \varepsilon \left( \langle \hat{r}^k \rangle^T X_{j_k} \right) X_{j_k},
\]

where now \( \varepsilon \in (0, 1] \). LS-BOOST(\( \varepsilon \)) has old roots — as noted by [13], LS-BOOST with \( M = 2 \) is known as “twicing,” a method proposed by Tukey [99]. The papers [14, 12, 13] present very interesting perspectives on LS-BOOST(\( \varepsilon \)), where they refer to the algorithm as \( L2 \)-Boost. [13] also obtains approximate expressions for the effective degrees of freedom of the \( L2 \)-Boost algorithm. In the non-stochastic setting, this is known as Matching Pursuit [65]. LS-BOOST(\( \varepsilon \)) is also closely related to Friedman’s MART algorithm [44].

Note that both algorithms often lead to models with attractive statistical properties [40, 51, 12, 13]. In this linear regression setting, while there may be several important concerns, it is often of primary importance to produce a parsimonious model with good out of sample predictive performance. When \( p \) is small relative to \( n \), minimizing the empirical least squares loss function \( L_n(\beta) \) usually achieves this goal. On the other hand, when \( n, p \gg 0 \) (and particularly when \( p > n \)), \( \hat{\beta}_{LS} \) often has poor predictive performance; in other words, \( \hat{\beta}_{LS} \) overfits the training data. Additionally \( \hat{\beta}_{LS} \) is almost always fully dense. Regularization techniques enable one to find a model with better predictive performance by balancing two competing objectives: \( (i) \) data fidelity, or how well the model fits the training data, and \( (ii) \) “shrinkage,” or a measure of model simplicity. Shrinkage is often measured using \( \| \beta \| \) for some appropriate norm \( \| \cdot \| \), whereby a coefficient vector with a relatively small value of \( \| \beta \| \) exhibits more shrinkage. The FS\( \varepsilon \) and LS-BOOST(\( \varepsilon \)) algorithms are effective, even in settings where \( n, p \gg 0 \) and/or \( p > n \), because they each impart a type of implicit regularization by tracing out a path of models with varying levels of data fidelity and shrinkage.

For both FS\( \varepsilon \) and LS-BOOST(\( \varepsilon \)), the choices of \( \varepsilon \) and \( M \) play crucial roles in the statistical behavior of the algorithm. Let us consider LS-BOOST(\( \varepsilon \)) alone for now. Setting \( \varepsilon = 1 \) corresponds to minimizing the empirical least squares loss function \( L_n(\beta) \) along the direction of the selected feature, i.e., it holds that \( \langle \hat{r}^k \rangle^T X_{j_k} = \arg \min_{u \in \mathbb{R}} L_n(\beta^k + u e_{j_k}) \). Qualitatively speaking, LS-BOOST(\( \varepsilon \)) does eventually minimize the empirical least squares loss function.
as long as $\varepsilon > 0$, but a small value of $\varepsilon$ (for example, $\varepsilon = 0.001$) slows down the rate of convergence as compared to the choice $\varepsilon = 1$. Thus it may seem counterintuitive to set $\varepsilon < 1$; however with a small value of $\varepsilon$ it is possible to explore a larger class of models, with varying degrees of shrinkage. It has been observed empirically that small values of $\varepsilon$ often lead to models with better predictive power [40]. In practice, one might set $\varepsilon$ relatively small and use a holdout dataset to select the best performing model found throughout the course of the algorithm; in many instances the selected model is found long before convergence to the empirical least squares solution. The role of $M$ and $\varepsilon$ in $\text{FS}_\varepsilon$ is very similar. In short, both $M$ and $\varepsilon$ together control the training error (data fidelity) and the amount of shrinkage (regularization) for both $\text{LS-Boost}(\varepsilon)$ and $\text{FS}_\varepsilon$. We refer the reader to Figure 5.1 depicting the evolution of the algorithmic properties of $\text{FS}_\varepsilon$ and $\text{LS-Boost}(\varepsilon)$ as a function of $M$ and $\varepsilon$.

Up until the present work, and as pointed out by [51], the understanding of how the algorithmic parameters $\varepsilon$ and $M$ control the tradeoffs between data fidelity and shrinkage in $\text{FS}_\varepsilon$ and $\text{LS-Boost}(\varepsilon)$ has been rather qualitative. One of the contributions of this chapter is a precise quantification of this tradeoff, for both $\text{FS}_\varepsilon$ and $\text{LS-Boost}(\varepsilon)$. Indeed, we present, for the first time, precise descriptions of how the quantities $\varepsilon$ and $M$ control the amount of training error and regularization in $\text{FS}_\varepsilon$ and $\text{LS-Boost}(\varepsilon)$. These precise computational guarantees are enabled by new connections to first-order methods in convex optimization. In particular, Section 5.3 presents a new unifying framework for interpreting $\text{FS}_\varepsilon$ and $\text{LS-Boost}(\varepsilon)$ as instances of the subgradient descent method of convex optimization, applied to the problem of minimizing the largest correlation between residuals and predictors.

Our results in Section 5.2 show that the predicted values $\mathbf{X}\hat{\beta}^k$ obtained from $\text{LS-Boost}(\varepsilon)$ converge (at a globally linear rate) to the least squares fit as $k \to \infty$, this holding true for any value of $\varepsilon \in (0, 1]$. On the other hand, for $\text{FS}_\varepsilon$ with $\varepsilon > 0$, the iterates $\mathbf{X}\hat{\beta}^k$ need not necessarily converge to the least squares fit as $k \to \infty$. Indeed, the $\text{FS}_\varepsilon$ algorithm, by its operational definition, has a uniform learning rate $\varepsilon$ which remains fixed for all iterations; this makes it impossible to always guarantee convergence to a least squares solution with
\[ \rho = 0 \]
\[ \rho = 0.5 \]
\[ \rho = 0.9 \]

**Figure 5-1:** Evolution of LS-Boost \( (\varepsilon) \) and FS \( \varepsilon \) versus iterations (in the log-scale), run on a synthetic dataset with \( n = 50, p = 500 \); the features are drawn from a Gaussian distribution with pairwise correlations \( \rho \). The true \( \beta \) has ten nonzeros with \( \beta_i = 1, i \leq 10 \) and SNR = 1. Three different values of \( \rho \) have been considered (\( \rho = 0, 0.5 \) and 0.9) and \( \varepsilon \) varies from \( \varepsilon = 10^{-5} \) to \( \varepsilon = 1 \). The top row shows the training errors for different learning rates, and the bottom row shows the \( \ell_1 \) norm of the coefficients produced by the different algorithms for different learning rates. (Here the values have all been re-scaled so that the y-axis lies in \([0, 1]\)).

accuracy less than \( O(\varepsilon) \). While the predicted values of LS-Boost \( (\varepsilon) \) converge to a least squares solution at a linear rate, we show in Section 5.3 that the predictions from the FS \( \varepsilon \) algorithm converges to an approximate least squares solution, albeit at a global sublinear rate. For another perspective on the similarities and differences between FS \( \varepsilon \) and LS-Boost \( (\varepsilon) \), see [13].

\[ \text{For the purposes of this paper, linear convergence of a sequence } \{a_i\} \text{ will mean that } a_i \to \bar{a} \text{ and there exists a scalar } \gamma < 1 \text{ for which } (a_i - \bar{a})/(a_{i-1} - \bar{a}) \leq \gamma \text{ for all } i. \text{ Sublinear convergence will mean that there is no such } \gamma < 1 \text{ that satisfies the above property. For much more general versions of linear and sublinear convergence, see [9] for example.} \]
Boosting and LASSO

As mentioned previously, FS_ε and LS-BOOST(ε) are effective even in high-dimensional settings where p > n since they implicitly deliver regularized models. An alternative and very popular approach in such settings is based on an explicit regularization scheme, namely ℓ_1-regularized regression, i.e., LASSO [95]. The constraint version of LASSO with regularization parameter δ ≥ 0 is given by the following convex quadratic optimization problem:

\[
\text{LASSO} : \quad L^*_{n,\delta} := \min_{\beta} \frac{1}{2n} \| Y - X\beta \|_2^2 \quad \text{s.t.} \quad \| \beta \|_1 \leq \delta.
\]  

(5.1)

The nature of regularization via the LASSO is explicit — by its very formulation, it is set up to find the best least squares solution subject to a constraint on the ℓ_1 norm of the regression coefficients. This is in contrast to boosting algorithms like FS_ε and LS-BOOST(ε), wherein regularization is imparted implicitly as a consequence of the structural properties of the algorithm with ε and M controlling the amount of shrinkage. Although LASSO and the previously discussed boosting methods originate from different perspectives, there are interesting similarities between the two, as is nicely explored in [51, 31, 50]. Figure 5-2 (top panel) shows an example where the LASSO profile/path (the set of solutions of (5.1) as δ varies) is similar to the trajectories of FS_ε and LS-BOOST(ε) (for small values of ε). Although they are different in general (Figure 5-2, bottom panel), [31, 50] explores the connection more deeply.

Efforts to understand boosting algorithms in general and in particular the FS_ε algorithm paved the way for the celebrated Least Angle Regression aka the Lar algorithm [31] (see also [51]). The Lar algorithm is a democratic version of Forward Stepwise. Upon identifying the variable most correlated with the current residual in absolute value (as in Forward Stepwise), it moves the coefficient of the variable towards its least squares value in a continuous fashion. An appealing aspect of the Lar algorithm is that it provides a unified algorithmic framework for variable selection and shrinkage – one instance of Lar leads to a path algorithm for the LASSO, and a different instance leads to the limiting case of the FS_ε algorithm as ε → 0+, namely FS_0. In fact, the Stagewise version of the Lar algorithm provides an
Coefficient Profiles: LS-Boost($\varepsilon$), FS$_\varepsilon$ and Lasso

- **Lasso**
- LS-Boost($\varepsilon$), $\varepsilon = 0.01$
- FS$_\varepsilon$, $\varepsilon = 10^{-5}$

Figure 5-2: Coefficient Profiles for different algorithms as a function of the $\ell_1$ norm of the regression coefficients on two different datasets. The top row corresponds to a dataset where the coefficient profiles look very similar, and the bottom row corresponds to a dataset where the coefficient profile of LASSO is seen to be different from FS$_\varepsilon$ and LS-Boost($\varepsilon$).

efficient way to compute the coefficient profile for FS$_0$.

In spite of the various nice perspectives on FS$_\varepsilon$ and its connections to the LASSO as described above, the present understanding about the relationships between LASSO, FS$_\varepsilon$, and LS-Boost($\varepsilon$) for arbitrary datasets and $\varepsilon > 0$ is still fairly limited. One of the aims of our work is to contribute some substantial further understanding of the relationship between LASSO, FS$_\varepsilon$, and LS-Boost($\varepsilon$), particularly for arbitrary datasets where such understanding is still fairly limited. Motivated thusly, in Section 5.4, we introduce a new boosting algorithm, called R-FS$_{\varepsilon,\delta}$ (regularized FS$_\varepsilon$) that has close connections to LASSO. In particular, R-FS$_{\varepsilon,\delta}$ can be viewed as subgradient descent applied to the following optimization problem:

$$P_{\delta} : \minimize_r \| X^T r \|_{\infty} + \frac{1}{2\delta} \| r - y \|^2$$

where $r = y - X \beta$ for some $\beta$, \hspace{1cm} (5.2)

which is equivalent to the dual problem of the LASSO \hspace{1cm} (5.1). Moreover, just like the LAR
algorithm can be viewed as a master algorithm with special instances being the LASSO and FS0, in this chapter we establish that FSε, LS-BOOST(ε) and LASSO can all be viewed as special instances of subgradient descent applied to a particular problem in the parametric family \( \delta \in [0, \infty) \).

5.1.1 Contributions

A summary of the contributions of this chapter is as follows:

1. We analyze several boosting algorithms popularly used in the context of linear regression via the lens of first-order methods in convex optimization. We show that existing boosting algorithms, namely FSε and LS-BOOST(ε), can be viewed as instances of the subgradient descent method aimed at minimizing the maximum absolute correlation between the covariates and residuals, namely \( \|X^T r\|_\infty \). This viewpoint provides several insights about the operational characteristics of these boosting algorithms.

2. We derive novel computational guarantees for FSε and LS-BOOST(ε). These results quantify the rate at which the estimates produced by a boosting algorithm make their way towards an unregularized least squares fit (as a function of the number of iterations and the learning rate \( \varepsilon \)). In particular, we demonstrate that for any value of \( \varepsilon \in (0, 1] \) the estimates produced by LS-BOOST(ε) converge linearly to their respective least squares values and the \( \ell_1 \) norm of the coefficients grows at a rate \( O(\sqrt{\varepsilon k}) \). FSε on the other hand demonstrates a slower sublinear convergence rate to an \( O(\varepsilon) \)-approximate least squares solution, while the \( \ell_1 \) norm of the coefficients grows at a rate \( O(\varepsilon k) \).

3. Our computational guarantees yield precise characterizations of the amount of data-fidelity (training error) and regularization imparted by running a boosting algorithm for \( k \) iterations. These results apply to any dataset and do not rely upon any distributional or structural assumptions on the data generating mechanism.

4. We show that subgradient descent applied to a regularized version of the loss function
\[ \|X^T r\|_\infty, \] with regularization parameter \( \delta \), leads to a new algorithm which we call \( R\text{-FS}_{\varepsilon, \delta} \), that is a natural and simple generalization of \( \text{FS}_\varepsilon \). When compared to \( \text{FS}_\varepsilon \), the algorithm \( R\text{-FS}_{\varepsilon, \delta} \) performs a seemingly minor rescaling of the coefficients at every iteration. As the number of iterations \( k \) increases, \( R\text{-FS}_{\varepsilon, \delta} \) delivers an approximate \text{LASSO} solution (5.1). Moreover, as the algorithm progresses, the \( \ell_1 \) norms of the coefficients evolve as a geometric series towards the regularization parameter value \( \delta \). We derive precise computational guarantees that inform us about the training error and regularization imparted by \( R\text{-FS}_{\varepsilon, \delta} \).

5. We present an adaptive extension of \( R\text{-FS}_{\varepsilon, \delta} \), called \( \text{PATH-R-FS}_\varepsilon \), that delivers a path of approximate \text{LASSO} solutions for any prescribed grid sequence of regularization parameters. We derive guarantees that quantify the average distance from the approximate path traced by \( \text{PATH-R-FS}_\varepsilon \) to the \text{LASSO} solution path.

We emphasize that all of these results apply to the finite sample setup with no assumptions about the dataset nor about the relative sizes of \( p \) and \( n \).

**Organization of the Chapter** This chapter is organized as follows. In Section 5.2 we analyze the convergence behavior of the \( \text{LS-Boost}(\varepsilon) \) algorithm. In Section 5.3 we present a unifying algorithmic framework for \( \text{FS}_\varepsilon \) and \( \text{LS-Boost}(\varepsilon) \) as subgradient descent. In Section 5.4 we present the regularized correlation minimization Problem (5.2) and a naturally associated boosting algorithm \( R\text{-FS}_{\varepsilon, \delta} \), as instantiations of subgradient descent on the family of Problems (5.2). In each of the above cases, we present precise computational guarantees of the algorithms for convergence of residuals, training errors, and shrinkage and study their statistical implications. In Section 5.5 we briefly discuss a few computational experiments that explore the statistical properties of the various boosting algorithms studied herein. To improve readability, most of the technical details have been placed in the Appendix C.
5.2 LS-Boost(\(\varepsilon\)): Computational Guarantees and Statistical Implications

We begin our formal study by examining the LS-Boost(\(\varepsilon\)) algorithm. We study the rate at which the coefficients generated by LS-Boost(\(\varepsilon\)) converge to the set of unregularized least square solutions. This characterizes the amount of data-fidelity as a function of the number of iterations and \(\varepsilon\). In particular, we show (global) linear convergence of the regression coefficients to the set of least squares coefficients, with similar convergence rates derived for the prediction estimates and the boosting training errors delivered by LS-Boost(\(\varepsilon\)). We also present bounds on the shrinkage of the regression coefficients \(\hat{\beta}_k\) as a function of \(k\) and \(\varepsilon\), thereby describing how the amount of shrinkage of the regression coefficients changes as a function of the number of iterations \(k\).

5.2.1 Computational Guarantees and Intuition

We first review some useful properties associated with the familiar least squares regression problem:

\[
\text{LS} : \quad L^*_n := \min_\beta L_n(\beta) := \frac{1}{2n} \|y - X\beta\|_2^2 \\
\text{s.t.} \quad \beta \in \mathbb{R}^p ,
\]

where \(L_n(\cdot)\) is the least squares loss, whose gradient is:

\[
\nabla L_n(\beta) = -\frac{1}{n} X^T (y - X\beta) = -\frac{1}{n} X^T r
\]

where \(r = y - X\beta\) is the vector of residuals corresponding to the regression coefficients \(\beta\). It follows that \(\beta\) is a least-squares solution of LS if and only if \(\nabla L_n(\beta) = 0\), which leads to the well known normal equations:

\[
0 = -X^T (y - X\beta) = -X^T r .
\]
It also holds that:

$$n \cdot \|\nabla L_n(\beta)\|_\infty = \|X^T r\|_\infty = \max_{j \in \{1, \ldots, p\}} \{|r^T X_j|\}.$$  \hfill (5.6)

The following theorem describes precise computational guarantees for LS-Boost(\(\varepsilon\)):
linear convergence of LS-Boost(\(\varepsilon\)) with respect to (5.3), and bounds on the \(\ell_1\) shrinkage of the coefficients produced. Note that the theorem uses the quantity \(\lambda_{p\text{min}}(X^T X)\) which denotes the smallest nonzero (and hence positive) eigenvalue of \(X^T X\).

**Theorem 5.1. (Linear Convergence of LS-Boost(\(\varepsilon\)) for Least Squares)** Consider the LS-Boost(\(\varepsilon\)) algorithm with learning rate \(\varepsilon \in (0, 1]\), and define the linear convergence rate coefficient \(\gamma\):

$$\gamma := \left(1 - \frac{\varepsilon(2 - \varepsilon)\lambda_{p\text{min}}(X^T X)}{4p}\right) < 1.$$  \hfill (5.7)

For all \(k \geq 0\) the following bounds hold:

(i) (training error): \(L_n(\hat{\beta}^k) - L_n^* \leq \frac{1}{2n}\|X_{LS}\|_2^2 \cdot \gamma^k\)

(ii) (regression coefficients): there exists a least squares solution \(\hat{\beta}_{LS}^k\) such that:

$$\|\hat{\beta}^k - \hat{\beta}_{LS}^k\|_2 \leq \frac{\|X_{LS}\|_2}{\sqrt{\lambda_{p\text{min}}(X^T X)}} \cdot \gamma^{k/2}.$$

(iii) (predictions): for every least-squares solution \(\hat{\beta}_{LS}\) it holds that

$$\|X\hat{\beta}^k - X\hat{\beta}_{LS}\|_2 \leq \|X_{LS}\|_2 \cdot \gamma^{k/2}.$$

(iv) (gradient norm/correlation values): \(\|\nabla L_n(\hat{\beta}^k)\|_\infty = \frac{1}{n}\|X^T r^k\|_\infty \leq \frac{1}{n}\|X_{LS}\|_2 \cdot \gamma^{k/2}\).
(v) ($\ell_1$-shrinkage of coefficients):

$$
\|\hat{\beta}^k\|_1 \leq \min \left\{ \sqrt{k} \sqrt{\frac{2}{2-\varepsilon}} \sqrt{\|X\hat{\beta}_{LS}\|^2_2 - \|X\hat{\beta}_{LS} - X\hat{\beta}^k\|^2_2}, \frac{\varepsilon\|X\hat{\beta}_{LS}\|_2}{1 - \sqrt{\gamma}} (1 - \gamma^{k/2}) \right\}
$$

(vi) (sparsity of coefficients): $\|\hat{\beta}^k\|_0 \leq k$. 

Before remarking on the various parts of Theorem 5.1, we first discuss the quantity $\gamma$ defined in (5.7), which is called the linear convergence rate coefficient. We can write $\gamma = 1 - \frac{\varepsilon}{\kappa(X^TX)}$, where $\kappa(X^TX)$ is defined to be the ratio $\kappa(X^TX) := \frac{p}{\lambda_{p_{\min}}(X^TX)}$. Note that $\kappa(X^TX) \in [1, \infty)$. To see this, let $\tilde{\beta}$ be an eigenvector associated with the largest eigenvalue of $X^TX$, then:

$$
0 < \lambda_{p_{\min}}(X^TX) \leq \lambda_{\max}(X^TX) = \frac{\|X\tilde{\beta}\|_2^2}{\|\tilde{\beta}\|_2^2} \leq \frac{\|X\|_{1,2}^2 \|\tilde{\beta}\|_2^2}{\|\tilde{\beta}\|_2^2} \leq p ,
$$

(5.8)

where the last inequality uses our assumption that the columns of $X$ have been normalized (whereby $\|X\|_{1,2} = 1$), and the fact that $\|\tilde{\beta}\|_1 \leq \sqrt{p}\|\tilde{\beta}\|_2$. This then implies that $\gamma \in [0.75, 1.0)$ – independent of any assumption on the dataset – and most importantly it holds that $\gamma < 1$.

Let us now make the following immediate remarks on Theorem 5.1:

- The bounds in parts (i)-(iv) state that the training errors, regression coefficients, predictions, and correlation values produced by LS-Boost($\varepsilon$) converge linearly (also known as geometric or exponential convergence) to their least squares counterparts: they decrease by at least the constant multiplicative factor $\gamma < 1$ for part (i), and by $\sqrt{\gamma}$ for parts (ii)-(iv), at every iteration. The bounds go to zero at this linear rate as $k \to \infty$.

- The computational guarantees in parts (i) - (vi) provide characterizations of the data-fidelity and shrinkage of the LS-Boost($\varepsilon$) algorithm for any given specifications of the learning rate $\varepsilon$ and the number of boosting iterations $k$. Moreover, the quantities
appearing in the bounds can be computed from simple characteristics of the data that can be obtained \textit{a priori} without even running the boosting algorithm. (And indeed, one can even substitute \( \|y\|_2 \) in place of \( \|X\hat{\beta}_{LS}\|_2 \) throughout the bounds if desired since \( \|X\hat{\beta}_{LS}\|_2 \leq \|y\|_2 \).)

For the bounds in parts (i) and (iii) of Theorem 5.1 the asymptotic limits (as \( k \to \infty \)) are the unregularized least squares training error and predictions — which are quantities that are uniquely defined even in the underdetermined case.

The bound in part (ii) of Theorem 5.1 is a statement concerning the regression coefficients. In this case, the notion of convergence needs to be appropriately modified from parts (i) and (iii), since the natural limiting object \( \hat{\beta}_{LS} \) is not necessarily unique. In this case, perhaps not surprisingly, the regression coefficients \( \hat{\beta}^k \) need not converge. The result in part (ii) of the theorem states that \( \hat{\beta}^k \) converges at a linear rate to the set of least squares solutions. In other words, at every LS-Boost(\( \varepsilon \)) boosting iteration, there exists a least squares solution \( \hat{\beta}^k_{LS} \) for which the presented bound holds. Here \( \hat{\beta}^k_{LS} \) is in fact the closest least squares solution to \( \hat{\beta}^k \) in the \( \ell_2 \) norm — and the particular candidate least squares solution \( \hat{\beta}^k_{LS} \) may be different for each iteration.

Comments on the global linear convergence rate in Theorem 5.1 The global linear convergence of LS-Boost(\( \varepsilon \)) proved in Theorem 5.1 while novel, is not at odds with the present understanding of such convergence for optimization problems. One can view LS-Boost(\( \varepsilon \)) as performing steepest descent optimization steps with respect to the \( \ell_1 \) norm unit ball (rather than the \( \ell_2 \) norm unit ball which is the canonical version of the steepest descent method, see [80]). It is known [80] that canonical steepest decent exhibits global linear convergence for convex quadratic optimization so long as the Hessian matrix \( Q \) of the quadratic objective function is positive definite, i.e., \( \lambda_{\text{min}}(Q) > 0 \). And for the least squares loss function \( Q = \frac{1}{n}X^TX \), which yields the condition that \( \lambda_{\text{min}}(X^TX) > 0 \). As discussed in [10], this result extends to other norms defining steepest descent as well. Hence what is modestly surprising herein is not the linear convergence \textit{per se}, but rather that LS-
Boost($\varepsilon$) exhibits global linear convergence even when $\lambda_{\text{min}}(X^T X) = 0$, i.e., even when $X$ does not have full column rank (essentially replacing $\lambda_{\text{min}}(X^T X)$ with $\lambda_{\text{pmin}}(X^T X)$ in our analysis). This derives specifically from the structure of the least squares loss function, whose function values (and whose gradient) are invariant in the null space of $X$, i.e., $L_n(\beta + d) = L_n(\beta)$ for all $d$ satisfying $Xd = 0$, and is thus rendered “immune” to changes in $\beta$ in the null space of $X^T X$.

5.2.2 Statistical Insights from the Computational Guarantees

Note that in most noisy problems, the limiting least squares solution is statistically less interesting than an estimate obtained in the interior of the boosting profile, since the latter typically corresponds to a model with better bias-variance tradeoff. We thus caution the reader that the bounds in Theorem 5.1 should not be merely interpreted as statements about how rapidly the boosting iterations reach the least squares fit. We rather intend for these bounds to inform us about the evolution of the training errors and the amount of shrinkage of the coefficients as the LS-BOOST($\varepsilon$) algorithm progresses and when $k$ is at most moderately large. When the training errors are paired with the profile of the $\ell_1$-shrinkage values of the regression coefficients, they lead to the ordered pairs:

\[
\left( \frac{1}{2n} \| y - X\hat{\beta}^k \|_2^2, \| \hat{\beta}^k \|_1 \right), \quad k \geq 1 ,
\]

which describes the data-fidelity and $\ell_1$-shrinkage tradeoff as a function of $k$, for the given learning rate $\varepsilon > 0$. This profile is described in Figure 5-3 for several data instances. The bounds in Theorem 5.1 provide estimates for the two components of the ordered pair (5.9), and they can be computed prior to running the boosting algorithm. For simplicity, let us use the following crude estimate:

\[
\ell_k := \min \left\{ \frac{\| X\hat{\beta}_{LS} \|_2}{2 - \varepsilon} \sqrt{\frac{k\varepsilon}{2}} , \frac{\varepsilon \| X\hat{\beta}_{LS} \|_2}{1 - \sqrt{\gamma}} \left( 1 - \gamma^{\frac{k}{2}} \right) \right\},
\]
which is an upper bound of the bound in part (v) of the theorem, to provide an upper approximation of $\|\hat{\beta}_k\|_1$. Combining the above estimate with the guarantee in part (i) of Theorem 5.1 in (5.9), we obtain the following ordered pairs:

$$\left(\frac{1}{2n}\|X\hat{\beta}_{LS}\|_2^2 \cdot \gamma^k + L_n^*, \ell_k\right), \quad k \geq 1,$$

which describe the entire profile of the training error bounds and the $\ell_1$-shrinkage bounds as a function of $k$ as suggested by Theorem 5.1. These profiles, as described above in (5.10), are illustrated in Figure 5.4.

$\ell_1$ shrinkage versus data-fidelity tradeoffs: LS-Boost($\varepsilon$), FS$\varepsilon$, and Lasso

![Figure 5-3: Figure showing profiles of $\ell_1$ norm of the regression coefficients versus training error for LS-Boost($\varepsilon$), FS$\varepsilon$ and Lasso. (Left panel) Shows profiles for a synthetic dataset where the covariates are drawn from a Gaussian distribution with pairwise correlations $\rho = 0.5$. The true $\beta$ has ten non-zeros with $\beta_i = 1$ for $i = 1, \ldots, 10$, and SNR = 1. Here we ran LS-Boost($\varepsilon$) with $\varepsilon = 1$ and ran FS$\varepsilon$ with $\varepsilon = 10^{-2}$. The middle (and right) panel profiles corresponds to the Prostate cancer dataset [31]. Here we ran LS-Boost($\varepsilon$) with $\varepsilon = 0.01$ and we ran FS$\varepsilon$ with $\varepsilon = 10^{-5}$. The right panel figure is a zoomed-in version of the middle panel in order to highlight the difference in profiles between LS-Boost($\varepsilon$), FS$\varepsilon$ and Lasso. The vertical axes have been normalized so that the training error at $k = 0$ is one, and the horizontal axes have been scaled to the unit interval (to express the $\ell_1$-norm of $\hat{\beta}^k$ as a fraction of the maximum).]

It is interesting to consider the profiles of Figure 5.4 alongside the explicit regularization framework of the LASSO (5.1) which also traces out a profile of the form (5.9):

$$\left(\frac{1}{2n}\|y - X\hat{\beta}^\star_0\|_2^2 \cdot \delta, \|\hat{\beta}^\star_0\|_1\right), \quad \delta \geq 0,$$

(5.11)
LS-Boost(ε) algorithm: ℓ₁-shrinkage versus data-fidelity tradeoffs (theoretical bounds)

Synthetic dataset (κ = 1)

Synthetic dataset (κ = 25)

Leukemia dataset

Figure 5-4: Figure showing profiles of ℓ₁ shrinkage of the regression coefficients versus training error for the LS-Boost(ε) algorithm, for different values of the learning rate ε (denoted by the moniker “eps” in the legend). The profiles have been obtained from the computational bounds in Theorem 5.1. The left and middle panels correspond to synthetic values of the ratio κ = p_\text{max}/p_{\text{min}}, and for the right panel profiles the value of κ (here, κ = 270.05) is extracted from the Leukemia dataset, taken from [24]. The vertical axes have been normalized so that the training error at k = 0 is one, and the horizontal axes have been scaled to the unit interval.

as a function of δ, where, \( \hat{\beta}_δ^* \) is a solution to the LASSO problem \((5.1)\). For a value of δ := ℓ_k the optimal objective value of the LASSO problem will serve as a lower bound of the corresponding LS-Boost(ε) loss function value at iteration k. Thus the training error of \( \hat{\beta}_δ^k \) delivered by the LS-Boost(ε) algorithm will be sandwiched between the following lower and upper bounds:

\[
L_{i,k} := \frac{1}{2n} \| y - X\hat{\beta}_δ^* \|_2^2 \leq \frac{1}{2n} \| y - X\hat{\beta}_δ^k \|_2^2 \leq \frac{1}{2n} \| X\hat{\beta}_\text{LS} \|_2^2 \cdot \gamma^k + L_*^n =: U_{i,k}
\]

for every k. Note that the difference between the upper and lower bounds above, given by: \( U_{i,k} - L_{i,k} \) converges to zero as k → ∞. Figure 5-3 shows the training error versus shrinkage profiles for LS-Boost(ε) and LASSO for different datasets.

**Interpreting the parameters and algorithm dynamics** There are several determinants of the quality of the bounds in the different parts of Theorem 5.1 which can be
grouped into:

- algorithmic parameters: this includes the learning rate \( \varepsilon \) and the number of iterations \( k \), and
- data dependent quantities: \( \|X\hat{\beta}_{LS}\|_2 \), \( \lambda_{\text{pmin}}(X^TX) \), and \( p \).

The coefficient of linear convergence is given by the quantity \( \gamma := 1 - \frac{\varepsilon(2-\varepsilon)}{4\kappa(X^TX)} \), where \( \kappa(X^TX) := \frac{p}{\lambda_{\text{pmin}}(X^TX)} \). Note that \( \gamma \) is monotone decreasing in \( \varepsilon \) for \( \varepsilon \in (0, 1] \), and is minimized at \( \varepsilon = 1 \). This simple observation confirms the general intuition about LS-BOOST(\( \varepsilon \)): \( \varepsilon = 1 \) corresponds to the most aggressive model fitting behavior in the LS-BOOST(\( \varepsilon \)) family, with smaller values of \( \varepsilon \) corresponding to a slower model fitting process. The ratio \( \kappa(X^TX) \) is a close cousin of the condition number associated with the data matrix \( X \) — and smaller values of \( \kappa(X^TX) \) imply a faster rate of convergence.

In the overdetermined case with \( n \geq p \) and \( \text{rank}(X) = p \), the condition number \( \bar{\kappa}(X^TX) := \frac{\lambda_{\text{max}}(X^TX)}{\lambda_{\text{min}}(X^TX)} \) plays a key role in determining the stability of the least-squares solution \( \hat{\beta}_{LS} \) and in measuring the degree of multicollinearity present. Note that \( \bar{\kappa}(X^TX) \in [1, \infty) \), and that the problem is better conditioned for smaller values of this ratio. Furthermore, since \( \text{rank}(X) = p \) it holds that \( \lambda_{\text{pmin}}(X^TX) = \lambda_{\text{min}}(X^TX) \), and thus \( \bar{\kappa}(X^TX) \leq \kappa(X^TX) \) by (5.8). Thus the condition number \( \kappa(X^TX) \) always upper bounds the classical condition number \( \bar{\kappa}(X^TX) \), and if \( \lambda_{\text{max}}(X^TX) \) is close to \( p \), then \( \bar{\kappa}(X^TX) \approx \kappa(X^TX) \) and the two measures essentially coincide. Finally, since in this setup \( \hat{\beta}_{LS} \) is unique, part (ii) of Theorem 5.1 implies that the sequence \( \{\hat{\beta}^k\} \) converges linearly to the unique least squares solution \( \hat{\beta}_{LS} \).

In the underdetermined case with \( p > n \), \( \lambda_{\text{min}}(X^TX) = 0 \) and thus \( \bar{\kappa}(X^TX) = \infty \). On the other hand, \( \kappa(X^TX) < \infty \) since \( \lambda_{\text{pmin}}(X^TX) \) is the smallest nonzero (hence positive) eigenvalue of \( X^TX \). Therefore the condition number \( \kappa(X^TX) \) is similar to the classical condition number \( \bar{\kappa}(\cdot) \) restricted to the subspace \( \mathcal{S} \) spanned by the columns of \( X \) (whose dimension is \( \text{rank}(X) \)). Interestingly, the linear rate of convergence enjoyed by LS-BOOST(\( \varepsilon \)) is in a sense adaptive — the algorithm automatically adjusts itself to the convergence rate dictated by the parameter \( \gamma \) “as if” it knows that the null space of \( X \) is not relevant.
5.3 Boosting Algorithms as Subgradient Descent

In this section we present a new unifying framework for interpreting the three boosting algorithms that were discussed in Section 5.1, namely FS_\(\varepsilon\) and LS-BOOST(\(\varepsilon\)). We show herein that all three algorithmic families can be interpreted as instances of the subgradient descent method of convex optimization, applied to the problem of minimizing the largest correlation between residuals and predictors. Interestingly, this unifying lens will also result in a natural generalization of FS_\(\varepsilon\) with very strong ties to the LASSO solutions, as we will present in Sections 5.4. The framework presented in this section leads to convergence guarantees for FS_\(\varepsilon\). In Theorem 5.2 herein, we present a theoretical description of the evolution of the FS_\(\varepsilon\) algorithm, in terms of its data-fidelity and shrinkage guarantees as a function of the number of boosting iterations. These results are a consequence of the computational guarantees for FS_\(\varepsilon\) that inform us about the rate at which the FS_\(\varepsilon\) training error, regression coefficients, and predictions make their way to their least squares counterparts. In order to develop these results, we first motivate and briefly review the subgradient descent method of convex optimization.

5.3.1 Brief Review of Subgradient Descent

We briefly motivate and review the subgradient descent method for non-differentiable convex optimization problems. Consider the following optimization problem:

\[
\begin{align*}
    f^{*} := \min_{x} f(x) \\
    \text{s.t.} \quad x \in P ,
\end{align*}
\]

(5.12)

where \( P \subseteq \mathbb{R}^n \) is a closed convex set and \( f(\cdot) : P \rightarrow \mathbb{R} \) is a convex function. If \( f(\cdot) \) is differentiable, then \( f(\cdot) \) will satisfy the following gradient inequality:

\[
f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for any } x, y \in P ,
\]

136
which states that \( f(\cdot) \) lies above its first-order (linear) approximation at \( x \). One of the most intuitive optimization schemes for solving (5.12) is the method of gradient descent. This method is initiated at a given point \( x^0 \in P \). If \( x^k \) is the current iterate, then the next iterate is given by the update formula:

\[
x^{k+1} \leftarrow \Pi_P(x^k - \alpha_k \nabla f(x^k)).
\]

In this method the potential new point is \( x^k - \alpha_k \nabla f(x^k) \), where \( \alpha_k > 0 \) is called the step-size at iteration \( k \), and the step is taken in the direction of the negative of the gradient. If this potential new point lies outside of the feasible region \( P \), it is then projected back onto \( P \). Here recall that \( \Pi_P(\cdot) \) is the Euclidean projection operator, namely

\[
\Pi_P(\cdot) := \arg\min_{y \in P} \|x - y\|_2.
\]

Now suppose that \( f(\cdot) \) is not differentiable. By virtue of the fact that \( f(\cdot) \) is convex, then \( f(\cdot) \) will have a subgradient at each point \( x \). Recall that \( g \) is a subgradient of \( f(\cdot) \) at \( x \) if the following subgradient inequality holds:

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

which generalizes the gradient inequality above and states that \( f(\cdot) \) lies above the linear function on the right side of (5.13). Because there may exist more than one subgradient of \( f(\cdot) \) at \( x \), let \( \partial f(x) \) denote the set of subgradients of \( f(\cdot) \) at \( x \). Then \( g \in \partial f(x) \) denotes that \( g \) is a subgradient of \( f(\cdot) \) at the point \( x \), and so \( g \) satisfies (5.13) for all \( y \). The subgradient descent method (see [90], for example) is a simple generalization of the method of gradient descent to the case when \( f(\cdot) \) is not differentiable. One simply replaces the gradient by the subgradient, yielding the following update scheme:

\[
\text{Compute a subgradient of } f(\cdot) \text{ at } x^k : \quad g^k \in \partial f(x^k)
\]

\[
\text{Perform update at } x^k : \quad x^{k+1} \leftarrow \Pi_P(x^k - \alpha_k g^k).
\]

The following proposition summarizes a well-known computational guarantee associated with the subgradient descent method.

**Proposition 5.1. (Convergence Bound for Subgradient Descent[80, 73])** Consider the subgradient descent method (5.14), using a constant step-size \( \alpha_i = \alpha \) for all \( i \). Let \( x^* \)}
be an optimal solution of (5.12) and suppose that the subgradients are uniformly bounded, namely \( \| \mathbf{g}_i \|_2 \leq G \) for all \( i \geq 0 \). Then for each \( k \geq 0 \), the following inequality holds:

\[
\min_{i \in \{0, \ldots, k\}} f(x^i) \leq f^* + \frac{\|x^0 - x^*\|_2^2}{2(k+1)\alpha} + \frac{\alpha G^2}{2} \ . \quad (5.15)
\]

The left side of (5.15) is simply the best objective function value obtained among the first \( k \) iterations. The right side of (5.15) bounds the best objective function value from above, namely the optimal value \( f^* \) plus a nonnegative quantity that is a function of the number of iterations \( k \), the constant step-size \( \{\alpha_i\} \), the bound \( G \) on the norms of subgradients, and the distance from the initial point to an optimal solution \( x^* \) of (5.12). Note that for a fixed step-size \( \alpha > 0 \), the right side of (5.15) goes to \( \frac{\alpha G^2}{2} \) as \( k \to \infty \). In the interest of completeness, we include a proof of Proposition 5.1 in Appendix C.2.1.

### 5.3.2 A Subgradient Descent Framework for Boosting

We now show that the boosting algorithms discussed in Section 5.1, namely FS_\( \varepsilon \) and LS-BOOST(\( \varepsilon \)), can be interpreted as instantiations of the subgradient descent method to minimize the largest absolute correlation between the residuals and predictors.

Let \( P_{\text{res}} := \{ r \in \mathbb{R}^n : r = y - X\beta \text{ for some } \beta \in \mathbb{R}^p \} \) denote the affine space of residuals and consider the following convex optimization problem:

\[
\text{Correlation Minimization (CM)} : \quad f^* := \min_r f(r) := \|X^T r\|_\infty \quad \text{s.t. } r \in P_{\text{res}} \ . \quad (5.16)
\]

which we dub the “Correlation Minimization” problem, or CM for short. Note an important subtlety in the CM problem, namely that the optimization variable in CM is the residual \( r \) and not the regression coefficient vector \( \beta \).

Since the columns of \( X \) have unit \( \ell_2 \) norm by assumption, \( f(r) \) is the largest absolute correlation between the residual vector \( r \) and the predictors. Therefore (5.16) is the convex
optimization problem of minimizing the largest correlation between the residuals and the predictors, over all possible values of the residuals. From (5.5) with \( r = y - X\beta \) we observe that \( X^T r = 0 \) if and only if \( \beta \) is a least squares solution, whereby \( f(r) = \|X^T r\|_\infty = 0 \) for the least squares residual vector \( r = \hat{r}_{\text{LS}} = y - X\hat{\beta}_{\text{LS}} \). Since the objective function in (5.16) is nonnegative, we conclude that \( f^* = 0 \) and the least squares residual vector \( \hat{r}_{\text{LS}} \) is also the unique optimal solution of the CM problem (5.16). Thus CM can be viewed as an optimization problem which also produces the least squares solution.

The following proposition states that the two boosting algorithms, FS_\( \varepsilon \) and LS-BOOST(\( \varepsilon \)), can both be viewed as instantiations of the subgradient descent method to solve the CM problem (5.16).

**Proposition 5.2.** Consider the subgradient descent method (5.14) with step-size sequence \( \{\alpha_k\} \) to solve the correlation minimization (CM) problem (5.16), initialized at \( \hat{r}_0 = y \). Then:

(i) the FS_\( \varepsilon \) algorithm is an instance of subgradient descent, with a constant step-size \( \alpha_k := \varepsilon \) at each iteration,

(ii) the LS-BOOST(\( \varepsilon \)) algorithm is an instance of subgradient descent, with non-uniform step-sizes \( \alpha_k := \varepsilon |\tilde{u}_{j_k}| \) at iteration \( k \), where \( \tilde{u}_{j_k} := (\hat{r}_k)^T X_{j_k} = \arg \min_u \|\hat{r}_k - X_{j_k} u\|_2^2 \).

Proof. We first prove (i). Recall the update of the residuals in FS_\( \varepsilon \):

\[
\hat{r}_{k+1} = \hat{r}_k - \varepsilon \cdot \text{sgn}((\hat{r}_k)^T X_{j_k}) X_{j_k}. 
\]

We first show that \( g_k := \text{sgn}((\hat{r}_k)^T X_{j_k}) X_{j_k} \) is a subgradient of the objective function \( f(r) = \|X^T r\|_\infty \) of the correlation minimization problem CM (5.16) at \( r = \hat{r}_k \). At iteration \( k \), FS_\( \varepsilon \) chooses the coefficient to update by selecting \( j_k \in \arg \max_{j \in \{1, \ldots, p\}} |(\hat{r}_k)^T X_j| \), whereby...
\[ \text{sgn}((\hat{r}^k)^T X_{jk}) ((\hat{r}^k)^T X_{jk}) = \|X^T (\hat{r}^k)\|_\infty, \text{ and therefore for any } r \text{ it holds that:} \]

\[ f(r) = \|X^T r\|_\infty \geq \text{sgn}((\hat{r}^k)^T X_{jk}) ((\hat{r}^k)^T X_{jk}) \]

\[ = \text{sgn}((\hat{r}^k)^T X_{jk}) ((X_{jk})^T (\hat{r}^k + r - \hat{r}^k)) \]

\[ = \|X^T (\hat{r}^k)\|_\infty + \text{sgn}((\hat{r}^k)^T X_{jk}) ((X_{jk})^T (r - \hat{r}^k)) \]

\[ = f(\hat{r}^k) + \text{sgn}((\hat{r}^k)^T X_{jk}) ((X_{jk})^T (r - \hat{r}^k)) . \]

Therefore using the definition of a subgradient in (5.13), it follows that \( g^k := \text{sgn}((\hat{r}^k)^T X_{jk}) X_{jk} \) is a subgradient of \( f(r) = \|X^T r\|_\infty \) at \( r = \hat{r}^k \). Therefore the update \( \hat{r}^{k+1} = \hat{r}^k - \varepsilon \cdot \text{sgn}((\hat{r}^k)^T X_{jk}) X_{jk} \) is of the form \( \hat{r}^{k+1} = \hat{r}^k - \varepsilon g^k \) where \( g^k \in \partial f(\hat{r}^k) \). Last of all notice that the update can also be written as \( \hat{r}^k - \varepsilon g^k = \hat{r}^{k+1} = y - X\hat{\beta}^{k+1} \in P_{\text{res}} \), hence \( \Pi_{P_{\text{res}}} (\hat{r}^k - \varepsilon g^k) = \hat{r}^k - \varepsilon g^k \), i.e., the projection step is superfluous here, and therefore \( \hat{r}^{k+1} = \Pi_{P_{\text{res}}} (\hat{r}^k - \varepsilon g^k) \), which is precisely the update for the subgradient descent method with step-size \( \alpha_k := \varepsilon \).

The proof of (ii) is the same as (i) with a step-size choice of \( \alpha_k = \frac{\varepsilon}{\varphi_j} \) at iteration \( k \).

Proposition 5.2 is interesting especially since FS\( _\varepsilon \) and LS-Boost(\( \varepsilon \)) have been traditionally interpreted as greedy coordinate descent or steepest descent type procedures \[51, 44\].

We now develop relevant convergence properties of FS\( _\varepsilon \), some of which are direct consequences of Proposition 5.2 based on well-known computational guarantees associated with the subgradient descent method (Proposition 5.1).
5.3.3 Deriving and Interpreting Computational Guarantees for FS\(_{\varepsilon}\)

The following theorem presents the convergence properties of FS\(_{\varepsilon}\), which are a consequence of the interpretation of FS\(_{\varepsilon}\) as an instance of the subgradient descent method.

**Theorem 5.2. (Convergence Properties of FS\(_{\varepsilon}\))** Consider the FS\(_{\varepsilon}\) algorithm with learning rate \(\varepsilon\). Let \(k \geq 0\) be the total number of iterations. Then there exists an index \(i \in \{0, \ldots, k\}\) for which the following bounds hold:

(i) (training error): \(L_n(\hat{\beta}^i) - L^*_n \leq \frac{p}{2n\lambda_{\text{min}}(X^TX)} \left[ \frac{\|X\hat{\beta}_{LS}\|_2^2}{\varepsilon(k+1)} + \varepsilon \right]^2\)

(ii) (regression coefficients): there exists a least squares solution \(\hat{\beta}_{LS}^i\) such that:

\[
\|\hat{\beta}^i - \hat{\beta}_{LS}^i\|_2 \leq \frac{\sqrt{p}}{\lambda_{\text{min}}(X^TX)} \left[ \frac{\|X\hat{\beta}_{LS}\|_2^2}{\varepsilon(k+1)} + \varepsilon \right]
\]

(iii) (predictions): for every least-squares solution \(\hat{\beta}_{LS}\) it holds that

\[
\|X\hat{\beta}^i - X\hat{\beta}_{LS}\|_2 \leq \frac{\sqrt{p}}{\sqrt{\lambda_{\text{min}}(X^TX)}} \left[ \frac{\|X\hat{\beta}_{LS}\|_2^2}{\varepsilon(k+1)} + \varepsilon \right]
\]

(iv) (correlation values) \(\|X^T\hat{r}^i\|_\infty \leq \frac{\|X\hat{\beta}_{LS}\|_2^2}{2\varepsilon(k+1)} + \frac{\varepsilon}{2}\)

(v) (\(\ell_1\)-shrinkage of coefficients): \(\|\hat{\beta}^i\|_1 \leq k\varepsilon\)

(vi) (sparsity of coefficients): \(\|\hat{\beta}^i\|_0 \leq k\).

The proof of Theorem 5.2 is presented in Appendix C.2.2.

**Interpreting the Computational Guarantees** Theorem 5.2 accomplishes for FS\(_{\varepsilon}\) what Theorem 5.1 did for LS-BOOST\((\varepsilon)\) — parts (i) – (iv) of the theorem describe the rate in which the training error, regression coefficients, and related quantities make their way towards their \((O(\varepsilon)\)-approximate) unregularized least squares counterparts. Part (v) of the theorem also
The FS\(_{\varepsilon}\) algorithm: \(\ell_1\) shrinkage versus data-fidelity tradeoffs (theoretical bounds)

Synthetic dataset (\(\kappa = 1\))

Figure 5.5: Figure showing profiles of \(\ell_1\) shrinkage bounds of the regression coefficients versus training error bounds for the FS\(_{\varepsilon}\) algorithm, for different values of the learning rate \(\varepsilon\). The profiles have been obtained from the bounds in parts (i) and (v) of Theorem 5.2. The left panel corresponds to a hypothetical dataset using \(\kappa = \frac{p}{\lambda_{\text{pmin}}} = 1\), and the middle and right panels use the parameters of the Leukemia dataset.

describes the rate at which the shrinkage of the regression coefficients evolve as a function of the number of boosting iterations. The rate of convergence of FS\(_{\varepsilon}\) is sublinear, unlike the linear rate of convergence for LS-Boost(\(\varepsilon\)). Note that this type of sublinear convergence implies that the rate of decrease of the training error (for instance) is dramatically faster in the very early iterations as compared to later iterations. Taken together, Theorems 5.2 and 5.1 highlight an important difference between the behavior of algorithms LS-Boost(\(\varepsilon\)) and FS\(_{\varepsilon}\):

- the limiting solution of the LS-Boost(\(\varepsilon\)) algorithm (as \(k \to \infty\)) corresponds to the unregularized least squares solution, but

- the limiting solution of the FS\(_{\varepsilon}\) algorithm (as \(k \to \infty\)) corresponds to an \(O(\varepsilon)\) approximate least squares solution.

As demonstrated in Theorems 5.1 and 5.2, both LS-Boost(\(\varepsilon\)) and FS\(_{\varepsilon}\) have nice convergence properties with respect to the unconstrained least squares problem (5.3). However, unlike
the convergence results for $\text{LS-Boost}(\varepsilon)$ in Theorem 5.1, $\text{FS}_\varepsilon$ exhibits a sublinear rate of convergence towards a suboptimal least squares solution. For example, part (i) of Theorem 5.2 implies in the limit as $k \to \infty$ that $\text{FS}_\varepsilon$ identifies a model with training error at most:

$$L^*_n + \frac{p\varepsilon^2}{2n(\lambda_{\text{pmin}}(X^T X))}.$$  \hspace{1cm} (5.17)

In addition, part (ii) of Theorem 5.2 implies that as $k \to \infty$, $\text{FS}_\varepsilon$ identifies a model whose distance to the set of least squares solutions $\{\hat{\beta}_{LS} : X^T X \hat{\beta}_{LS} = X^T y\}$ is at most: $\frac{\varepsilon \sqrt{p}}{\lambda_{\text{pmin}}(X^T X)}$.

Let us examine more carefully the properties of the sequence of models explored by $\text{FS}_\varepsilon$ and the corresponding tradeoffs between data fidelity and model complexity. Let $\text{TBound}$ and $\text{SBound}$ denote the training error bound and shrinkage bound in parts (i) and (v) of Theorem 5.2, respectively. Then simple manipulation of the arithmetic in these two bounds yields the following tradeoff equation:

$$\text{TBound} = \frac{p}{2n\lambda_{\text{pmin}}(X^T X)} \left[ \frac{\|X \hat{\beta}_{LS}\|_2^2}{\text{SBound} + \varepsilon} + \varepsilon \right]^2.$$

The above tradeoff between the training error bound and the shrinkage bound is illustrated in Figure 5-5, which shows this tradeoff curve for four different values of the learning rate $\varepsilon$. Except for very small shrinkage levels, lower values of $\varepsilon$ produce smaller training errors. But unlike the corresponding tradeoff curves for $\text{LS-Boost}(\varepsilon)$, there is a range of values of the shrinkage for which smaller values of $\varepsilon$ actually produce larger training errors, though admittedly this range is for very small shrinkage values. For more reasonable shrinkage values, smaller values of $\varepsilon$ will correspond to smaller values of the training error.

Part (v) of Theorems 5.1 and 5.2 presents shrinkage bounds for $\text{FS}_\varepsilon$ and $\text{LS-Boost}(\varepsilon)$, respectively. Let us briefly compare these bounds. Examining the shrinkage bound for $\text{LS-Boost}(\varepsilon)$, we can bound the left term from above by $\sqrt{k} \varepsilon \|X \hat{\beta}_{LS}\|_2$. We can also bound the right term from above by $\varepsilon \|X \hat{\beta}_{LS}\|_2/(1 - \sqrt{\gamma})$ where recall from Section 5.2 that $\gamma$ is the linear convergence rate coefficient $\gamma := 1 - \frac{\varepsilon(2-\varepsilon)\lambda_{\text{pmin}}(X^T X)}{4p}$. We may therefore alternatively
write the following shrinkage bound for LS-BOOST(\(\varepsilon\)):

\[
\|\hat{\beta}^k\|_1 \leq \|X\hat{\beta}_{LS}\|_2 \min \left\{ \sqrt{k\sqrt{\varepsilon}}, \frac{\varepsilon}{(1 - \sqrt{\gamma})} \right\}.
\] (5.18)

The shrinkage bound for FS\(\varepsilon\) is simply \(k\varepsilon\). Comparing these two bounds, we observe that not only does the shrinkage bound for FS\(\varepsilon\) grow at a faster rate as a function of \(k\) for large enough \(k\), but also the shrinkage bound for FS\(\varepsilon\) grows unbounded in \(k\), unlike the right term above for the shrinkage bound of LS-BOOST(\(\varepsilon\)).

5.4 Boosting and LASSO

In this section we introduce a new boosting algorithm, parameterized by a scalar \(\delta \geq 0\), which we denote by R-FS\(\varepsilon,\delta\) (for Regularized incremental Forward Stagewise regression), that is obtained by incorporating a simple rescaling step to the coefficient updates in FS\(\varepsilon\). We then introduce a regularized version of the Correlation Minimization (CM) problem (5.16) which we refer to as RCM. We show that the adaptation of the subgradient descent algorithmic framework to the Regularized Correlation Minimization problem RCM exactly yields the algorithm R-FS\(\varepsilon,\delta\). The new algorithm R-FS\(\varepsilon,\delta\) may be interpreted as a natural extension of popular boosting algorithms like FS\(\varepsilon\), and has the following notable properties:

- Whereas FS\(\varepsilon\) updates the coefficients in an additive fashion by adding a small amount \(\varepsilon\) to the coefficient most correlated with the current residuals, R-FS\(\varepsilon,\delta\) first shrinks all of the coefficients by a scaling factor \(1 - \frac{\varepsilon}{\delta} < 1\) and then updates the selected coefficient in the same additive fashion as FS\(\varepsilon\).

- R-FS\(\varepsilon,\delta\) delivers \(O(\varepsilon)\)-accurate solutions to the LASSO in the limit as \(k \to \infty\), unlike FS\(\varepsilon\) which delivers \(O(\varepsilon)\)-accurate solutions to the unregularized least squares problem.

- R-FS\(\varepsilon,\delta\) has computational guarantees similar in spirit to the ones described in the context of FS\(\varepsilon\) – these quantities directly inform us about the data-fidelity \textit{vis-à-vis}
shrinkage tradeoffs as a function of the number of boosting iterations and the learning rate $\varepsilon$.

The notion of using additional regularization along with the implicit shrinkage imparted by boosting is not new in the literature. Various interesting notions have been proposed in [45, 15, 102, 26, 41], see also [36] for further discussion. However, the framework we present here is new. We present a unified subgradient descent framework for a class of regularized CM problems that results in algorithms that have appealing structural similarities with forward stagewise regression type algorithms, while also being very strongly connected to the LASSO.

**Boosting with additional shrinkage \( \text{R-FS}_{\varepsilon, \delta} \)** Here we give a formal description of the R-FS_{\varepsilon, \delta} algorithm. R-FS_{\varepsilon, \delta} is controlled by two parameters: the learning rate $\varepsilon$, which plays the same role as the learning rate in FS_{\varepsilon}, and the “regularization parameter” $\delta \geq \varepsilon$. Our reason for referring to $\delta$ as a regularization parameter is due to the connection between R-FS_{\varepsilon, \delta} and the LASSO, which will be made clear later. The shrinkage factor, i.e., the amount by which we shrink the coefficients before updating the selected coefficient, is determined as $1 - \frac{\varepsilon}{\delta}$. Supposing that we choose to update the coefficient indexed by $j_k$ at iteration $k$, then the coefficient update may be written as:

$$
\hat{\beta}_{k+1} \leftarrow (1 - \frac{\varepsilon}{\delta}) \hat{\beta}_k + \varepsilon \cdot \text{sgn}(\hat{r}_k^T X_{j_k}) e_{j_k}.
$$

Below we give a concise description of R-FS_{\varepsilon, \delta}, including the update for the residuals that corresponds to the update for the coefficients stated above.

**Algorithm: R-FS_{\varepsilon, \delta}**

Fix the learning rate $\varepsilon > 0$, regularization parameter $\delta > 0$ such that $\varepsilon \leq \delta$, and number of iterations $M$.

Initialize at $\hat{r}^0 = y$, $\hat{\beta}^0 = 0$, $k = 0$. 

1. For $0 \leq k \leq M$ do the following:

2. Compute: $j_k \in \arg\max_{j \in \{1, \ldots, p\}} |(\hat{r}^k)^T X_j|$

3. $\hat{r}^{k+1} \leftarrow \hat{r}^k - \varepsilon \left[\text{sgn}((\hat{r}^k)^T X_{j_k}) X_{j_k} + \frac{1}{\delta}(\hat{r}^k - y)\right]$

   $\hat{\beta}^{k+1}_{j_k} \leftarrow (1 - \frac{\varepsilon}{\delta}) \hat{\beta}^k_{j_k} + \varepsilon \text{sgn}((\hat{r}^k)^T X_{j_k})$ and $\hat{\beta}^{k+1}_j \leftarrow (1 - \frac{\varepsilon}{\delta}) \hat{\beta}^k_j$, $j \neq j_k$

Note that R-FS$_{\varepsilon,\delta}$ and FS$_{\varepsilon}$ are structurally very similar – and indeed when $\delta = \infty$ then R-FS$_{\varepsilon,\delta}$ is exactly FS$_{\varepsilon}$. Note also that R-FS$_{\varepsilon,\delta}$ shares the same upper bound on the sparsity of the regression coefficients as FS$_{\varepsilon}$, namely for all $k$ it holds that: $\|\hat{\beta}^k\|_0 \leq k$. When $\delta < \infty$ then, as previously mentioned, the main structural difference between R-FS$_{\varepsilon,\delta}$ and FS$_{\varepsilon}$ is the additional rescaling of the coefficients by the factor $1 - \frac{\varepsilon}{\delta}$. This rescaling better controls the growth of the coefficients and, as will be demonstrated next, plays a key role in connecting R-FS$_{\varepsilon,\delta}$ to the LASSO.

**Regularized Correlation Minimization (RCM) and LASSO** The starting point of our formal analysis of R-FS$_{\varepsilon,\delta}$ is the Correlation Minimization (CM) problem (5.16), which we now modify by introducing a regularization term that penalizes residuals that are far from the vector of observations $y$. This modification leads to the following parametric family of optimization problems indexed by $\delta \in (0, \infty]$:

$$\text{RCM}_\delta : \quad f^*_\delta := \min_r f_\delta(r) \quad := \|X^T r\|_\infty + \frac{1}{2\delta} \|r - y\|_2^2$$

s.t. $r \in P_{\text{res}} := \{r \in \mathbb{R}^n : r = y - X\beta \text{ for some } \beta \in \mathbb{R}^p\}$, 

(5.19)

where “RCM” connotes Regularized Correlation Minimization. Note that RCM reduces to the correlation minimization problem CM (5.16) when $\delta = \infty$. RCM may be interpreted as the problem of minimizing, over the space of residuals, the largest correlation between the residuals and the predictors plus a regularization term that penalizes residuals that are far from the response $y$ (which itself can be interpreted as the residuals associated with the
Interestingly, as we show in Appendix C.3.1, RCM (5.19) is equivalent to the Lasso (5.1) via duality. This equivalence provides further insight about the regularization used to obtain RCM\(\delta\). Comparing the Lasso and RCM, notice that the space of the variables of the Lasso is the space of regression coefficients \(\beta\), namely \(\mathbb{R}^p\), whereas the space of the variables of RCM is the space of model residuals, namely \(\mathbb{R}^n\), or more precisely \(P_{\text{res}}\). The duality relationship shows that RCM\(\delta\) (5.19) is an equivalent characterization of the Lasso problem, just like the correlation minimization (CM) problem (5.16) is an equivalent characterization of the (unregularized) least squares problem. Recall that Proposition 5.2 showed that subgradient descent applied to the CM problem (5.19) (which is RCM\(\delta\) with \(\delta = \infty\)) leads to the well-known boosting algorithm FS\(\varepsilon\). We now extend this theme with the following Proposition, which demonstrates R-FS\(\varepsilon,\delta\) is equivalent to subgradient descent applied to RCM\(\delta\).

**Proposition 5.3.** The R-FS\(\varepsilon,\delta\) algorithm is an instance of subgradient descent to solve the regularized correlation minimization (RCM\(\delta\)) problem (5.19), initialized at \(\hat{r}^0 = y\), with a constant step-size \(\alpha_k := \varepsilon\) at each iteration.

The proof of Proposition 5.3 is presented in Appendix C.3.2.

### 5.4.1 R-FS\(\varepsilon,\delta\): Computational Guarantees and their Implications

In this subsection we present computational guarantees and convergence properties of the boosting algorithm R-FS\(\varepsilon,\delta\). Due to the structural equivalence between R-FS\(\varepsilon,\delta\) and subgradient descent applied to the RCM\(\delta\) problem (5.19) (Proposition 5.3) and the close connection between RCM\(\delta\) and the Lasso (Appendix C.3.1), the convergence properties of R-FS\(\varepsilon,\delta\) are naturally stated with respect to the Lasso problem (5.1). Similar to Theorem 5.2 which described such properties for FS\(\varepsilon\) (with respect to the unregularized least squares problem), we have the following properties for R-FS\(\varepsilon,\delta\).

**Theorem 5.3. (Convergence Properties of R-FS\(\varepsilon,\delta\) for the Lasso)** Consider the R-FS\(\varepsilon,\delta\) algorithm with learning rate \(\varepsilon\) and regularization parameter \(\delta \in (0, \infty)\), where \(\varepsilon \leq \delta\).
Then the regression coefficient $\hat{\beta}^k$ is feasible for the LASSO problem (5.1) for all $k \geq 0$. Let $k \geq 0$ denote a specific iteration counter. Then there exists an index $i \in \{0, \ldots, k\}$ for which the following bounds hold:

(i) (training error): $L_n(\hat{\beta}^i) - L_n^* \leq \frac{\delta}{n} \left[ \frac{\|X\hat{\beta}_{LS}\|^2_2}{2\varepsilon (k+1)} + 2\varepsilon \right]$

(ii) (predictions): for every LASSO solution $\hat{\beta}_{LS}^*$ it holds that

$$\|X\hat{\beta}^i - X\hat{\beta}_{LS}^*\|_2 \leq \sqrt{\frac{\delta\|X\hat{\beta}_{LS}\|^2_2}{\varepsilon (k+1)}} + 4\delta\varepsilon$$

(iii) ($\ell_1$-shrinkage of coefficients): $\|\hat{\beta}\|_1 \leq \delta \left[ 1 - \left(1 - \frac{\varepsilon}{\delta}\right)^k \right] \leq \delta$

(iv) (sparsity of coefficients): $\|\hat{\beta}\|_0 \leq k$.

The proof of Theorem 5.3 is presented in Appendix C.3.3.

**Figure 5-6**: Figure showing the evolution of the R-FS$_{\varepsilon, \delta}$ algorithm (with $\varepsilon = 10^{-4}$) for different values of $\delta$, as a function of the number of boosting iterations for the Prostate cancer dataset, with $n = 10, p = 44$, appearing in the bottom panel of Figure 5-7. [Left panel] shows the change of the $\ell_1$-norm of the regression coefficients. [Middle panel] shows the evolution of the training errors, and [Right panel] is a zoomed-in version of the middle panel. Here we took different values of $\delta$ given by $\delta = \text{frac} \times \delta_{\text{max}}$, where, $\delta_{\text{max}}$ denotes the $\ell_1$-norm of the minimum $\ell_1$-norm least squares solution, for 7 different values of frac.
Interpreting the Computational Guarantees  The statistical interpretations implied by the computational guarantees presented in Theorem 5.3 are analogous to those previously discussed for LS-BOOST(\(\varepsilon\)) (Theorem 5.1) and FS\(\varepsilon\) (Theorem 5.2). These guarantees inform us about the data-fidelity vis-à-vis shrinkage tradeoffs as a function of the number of boosting iterations, as nicely demonstrated in Figure 5-6. There is, however, an important differentiation between the properties of R-FS\(\varepsilon,\delta\) and the properties of LS-BOOST(\(\varepsilon\)) and FS\(\varepsilon\), namely:

- For LS-BOOST(\(\varepsilon\)) and FS\(\varepsilon\), the computational guarantees (Theorems 5.1 and 5.2) describe how the estimates make their way to a unregularized (\(O(\varepsilon)\)-approximate) least squares solution as a function of the number of boosting iterations.

- For R-FS\(\varepsilon,\delta\), our results (Theorem 5.3) characterize how the estimates approach a (\(O(\varepsilon)\)-approximate) LASSO solution.

Notice that like FS\(\varepsilon\), R-FS\(\varepsilon,\delta\) traces out a profile of regression coefficients. This is reflected in item (iii) of Theorem 5.3 which bounds the \(\ell_1\)-shrinkage of the coefficients as a function of the number of boosting iterations \(k\). Due to the rescaling of the coefficients, the \(\ell_1\)-shrinkage may be bounded by a geometric series that approaches \(\delta\) as \(k\) grows. Thus, there are two important aspects of the bound in item (iii): (a) the dependence on the number of boosting iterations \(k\) which characterizes model complexity during early iterations, and (b) the uniform bound of \(\delta\) which applies even in the limit as \(k \rightarrow \infty\) and implies that all regression coefficient iterates \(\hat{\beta}^k\) are feasible for the LASSO problem (5.1).

On the other hand, item (i) characterizes the quality of the coefficients with respect to the LASSO solution, as opposed to the unregularized least squares problem as in FS\(\varepsilon\). In the limit as \(k \rightarrow \infty\), item (i) implies that R-FS\(\varepsilon,\delta\) identifies a model with training error at most \(L*_{n,\delta} + \frac{2\delta}{n}\). This upper bound on the training error may be set to any prescribed error level by appropriately tuning \(\varepsilon\); in particular, for \(\varepsilon \approx 0\) and fixed \(\delta > 0\) this limit is essentially \(L*_{n,\delta}\). Thus, combined with the uniform bound of \(\delta\) on the \(\ell_1\)-shrinkage, we see that the R-FS\(\varepsilon,\delta\) algorithm delivers the LASSO solution in the limit as \(k \rightarrow \infty\).
It is important to emphasize that R-FS\(_{\varepsilon,\delta}\) should not just be interpreted as an algorithm to solve the LASSO. Indeed, like FS\(_{\varepsilon}\), the trajectory of the algorithm is important and R-FS\(_{\varepsilon,\delta}\) may identify a more statistically interesting model in the interior of its profile. Thus, even if the LASSO solution for \(\delta\) leads to overfitting, the R-FS\(_{\varepsilon,\delta}\) updates may visit a model with better predictive performance by trading off bias and variance in a more desirable fashion suitable for the particular problem at hand.

Figure 5-7 shows the profiles of R-FS\(_{\varepsilon,\delta}\) for different values of \(\delta \leq \delta_{\text{max}}\), where \(\delta_{\text{max}}\) is the \(\ell_1\)-norm of the minimum \(\ell_1\)-norm least squares solution. Curiously enough, Figure 5-7 shows that in some cases, the profile of R-FS\(_{\varepsilon,\delta}\) bears a lot of similarities with that of the LASSO (as presented in Figure 5-2). However, the profiles are in general different. Indeed, R-FS\(_{\varepsilon,\delta}\) imposes a uniform bound of \(\delta\) on the \(\ell_1\)-shrinkage, and so for values larger than \(\delta\) we cannot possibly expect R-FS\(_{\varepsilon,\delta}\) to approximate the LASSO path. However, even if \(\delta\) is taken to be sufficiently large (but finite) the profiles may be different. In this connection it is helpful to draw the analogy between the curious similarities between the FS\(_{\varepsilon}\) (i.e., R-FS\(_{\varepsilon,\delta}\) with \(\delta = \infty\)) and LASSO coefficient profiles, even though the profiles are different in general.

As an aside, we point out that one can also interpret R-FS\(_{\varepsilon,\delta}\) as the Frank-Wolfe algorithm in convex optimization applied to the LASSO (5.1) in line with [3]. We refer the reader to the paper [36] for an expanded discussion of this point.

A Modified Forward Stagewise Algorithm for Computing the LASSO Path As we have established, R-FS\(_{\varepsilon,\delta}\) (which is a very close cousin of FS\(_{\varepsilon}\)) delivers solutions to the LASSO problem (5.1) for a fixed but arbitrary \(\delta\), in the limit as \(k \to \infty\) with \(\varepsilon \approx 0\). While R-FS\(_{\varepsilon,\delta}\) by itself may be considered as a regularization scheme with excellent statistical properties, the boosting profile delivered by R-FS\(_{\varepsilon,\delta}\) might in some cases be different from the LASSO coefficient profile, as we saw in Figure 5-7. Therefore, it is natural to investigate the following question: is it possible to modify the R-FS\(_{\varepsilon,\delta}\) algorithm, while still retaining its basic algorithmic characteristics, so that it delivers an approximate LASSO coefficient profile for any dataset? We answer this question in the affirmative. Indeed, we show that an
adaptive version of $\text{R-FS}_{\varepsilon,\delta}$, which we call PATH-R-FS, approximates the path of LASSO solutions with precise bounds that quantify the approximation error over the path. These bounds are analogous to those in Theorem 5.3, except that now the relevant measures of data fidelity (i.e., the training errors) are shown to be sufficiently small on average across the entire path of $\delta$ values. In the interest of brevity we do not include these results here; instead we refer the reader to the full paper [36].

Figure 5-7: Coefficient profiles for $\text{R-FS}_{\varepsilon,\delta}$ as a function of the $\ell_1$-norm of the regression coefficients, for the same datasets appearing in Figure 5-2. For each example, different values of $\delta$ have been considered. The left panel corresponds to the choice $\delta = \infty$, i.e., $\text{FS}_\varepsilon$. In all the above cases, the algorithms were run for a maximum of 100,000 boosting iterations with $\varepsilon = 10^{-4}$. [Top Panel] Corresponds to the Prostate cancer dataset with $n = 98$ and $p = 8$. All the coefficient profiles look similar, and they all seem to coincide with the LASSO profile (see also Figure 5-2). [Bottom Panel] Shows the Prostate cancer dataset with a subset of samples $n = 10$ with all interactions included with $p = 44$. The coefficient profiles in this example are sensitive to the choice of $\delta$ and are seen to be more constrained towards the end of the path, for decreasing $\delta$ values. The profiles are different than the LASSO profiles, as seen in Figure 5-2. The regression coefficients at the end of the path correspond to approximate LASSO solutions, for the respective values of $\delta$. 
5.5 Statistical properties of boosting algorithms: an empirical study

We performed some experiments to better understand the statistical behavior of the different boosting methods described in this paper. We summarize our findings here; for details (including tables, figures and discussions) we refer the reader to the full paper [36].

Sensitivity of the Learning Rate in LS-Boost(\(\varepsilon\)) and FS\(_\varepsilon\) We explored how the training and test errors for LS-Boost(\(\varepsilon\)) and FS\(_\varepsilon\) change as a function of the number of boosting iterations and the learning rate. We observed that the best predictive models were sensitive to the choice of \(\varepsilon\)—the best models were obtained at values larger than zero and smaller than one. When compared to LASSO, stepwise regression [31] and FS\(_0\) [31]; FS\(_\varepsilon\) and LS-Boost(\(\varepsilon\)) were found to be as good as the others, in some cases the better than the rest.

Statistical properties of R-FS\(_{\varepsilon,\delta}\), LASSO and FS\(_\varepsilon\): an empirical study We performed some experiments to evaluate the performance of R-FS\(_{\varepsilon,\delta}\), in terms of predictive accuracy and sparsity of the optimal model, versus the more widely known methods FS\(_\varepsilon\) and LASSO. We found that when \(\delta\) was larger than the best \(\delta\) for the LASSO (in terms of obtaining a model with the best predictive performance), R-FS\(_{\varepsilon,\delta}\) delivered a model with excellent statistical properties – R-FS\(_{\varepsilon,\delta}\) led to sparse solutions and the predictive performance was as good as, and in some cases better than, the LASSO solution. We observed that the choice of \(\delta\) does not play a very crucial role in the R-FS\(_{\varepsilon,\delta}\) algorithm, once it is chosen to be reasonably large; indeed the number of boosting iterations play a more important role. The best models delivered by R-FS\(_{\varepsilon,\delta}\) were more sparse than FS\(_\varepsilon\).
Chapter 6

A New Perspective on Boosting in Logistic Regression and AdaBoost

6.1 Review of Mirror Descent for Nonsmooth Convex Optimization

Herein we review relevant facts about the mirror descent method for nonsmooth convex optimization. We also refer the reader to Section 3.2 of Chapter 3 for several relevant facts about the greedy coordinate descent method.

Notation  For a vector $x \in \mathbb{R}^n$, $x_i$ denotes the $i^{th}$ coordinate; we use superscripts to index vectors in a sequence $\{x^k\}$. Let $e_j$ denote the $j^{th}$ unit vector in $\mathbb{R}^n$, and let $e = (1, \ldots, 1)$. Let $\|\cdot\|_q$ denote the $\ell_q$-norm for $q \in [1, \infty]$ with unit ball $B_q$, and let $\|v\|_0$ denote the number of non-zero coefficients of the vector $v$. For $A \in \mathbb{R}^{m \times n}$, let $\|A\|_{q_1, q_2} := \max_{x: \|x\|_{q_1} \leq 1} \|Ax\|_{q_2}$ be the operator norm. For a scalar $\alpha$, $\text{sgn}(\alpha)$ denotes the sign of $\alpha$, and $\alpha^+, \alpha^-$ denote the positive and negative parts of $\alpha$, respectively. For a given norm $\|\cdot\|$ on $\mathbb{R}^n$, $\|\cdot\|_*$ denotes the dual norm defined by $\|s\|_* = \max_{x: \|x\| \leq 1} s^T x$. Let $\partial f(\cdot)$ denote the subdifferential operator.
of a convex function $f(\cdot)$. The notation “$\tilde{v} \leftarrow \arg\max_{v \in S} \{f(v)\}$” denotes assigning $\tilde{v}$ to be any optimal solution of the problem $\max_{v \in S} \{f(v)\}$. For a convex set $P$ let $\Pi_P(\cdot)$ denote the Euclidean projection operator onto $P$, namely $\Pi_P(x) := \arg\min_{x \in P} \|x - \bar{x}\|_2$.

The mirror descent method (a generalization of the subgradient descent method) is designed to solve an optimization problem in the form:

$$\text{(Primal): } \min_{x \in P} \; f(x) ,$$

where $P \subseteq \mathbb{R}^n$ is a closed convex set, $\mathbb{R}^n$ is considered with the given norm $\| \cdot \|$, and $f(\cdot) : P \rightarrow \mathbb{R}$ is a (possibly non-smooth) convex function. Recall that $g$ is a subgradient of $f(\cdot)$ at $x$ if $f(y) \geq f(x) + g^T(y - x)$ for all $y \in P$, and we denote the set of subgradients of $f(\cdot)$ at $x$ by $\partial f(x)$. We assume with no loss of generality that $\partial f(x) \neq \emptyset$ for all $x \in P$. We presume that computation of a subgradient at $x \in P$ is not a burdensome task. Furthermore, we assume that $f(\cdot)$ has Lipschitz function values with Lipschitz constant $L_f$, i.e., it holds that $|f(x) - f(y)| \leq L_f \|x - y\|$ for all $x, y \in P$.

The classical subgradient descent method for solving (6.1) determines the next iterate by taking a step $\alpha$ in the negative direction of a subgradient at the current point, and then projecting the resulting point back onto the set $P$. If $x^k$ is the current iterate, subgradient descent proceeds by computing a subgradient $g^k \in \partial f(x^k)$, and determines the next iterate as $x^{k+1} \leftarrow \Pi_P(x^k - \alpha_k g^k)$, where $\alpha_k$ is the step-length, and $\Pi_P(\cdot)$ is the Euclidean projection onto the set $P$.

The formal statement of the subgradient descent method and an associated well known computational guarantee is presented in Section 5.3 of Chapter 5.

We now turn to a brief review of the mirror descent method. For our purposes we are primarily interested in mirror descent for instances of (6.1) where $f(\cdot)$ is conveyed with bilinear structure, namely:

$$f(x) := \max_{\lambda \in \mathbb{Q}} \; x^T A \lambda ,$$

(6.2)
where \( Q \subseteq \mathbb{R}^m \) is a convex and compact set and \( A \in \mathbb{R}^{n \times m} \). In the case when \( P \) in (6.1) is bounded, we define a dual function \( p(\cdot) : Q \to \mathbb{R} \) by

\[
p(\lambda) := \min_{x \in P} x^T A \lambda ,
\]

(6.3)

for which we may be interested in solving the dual problem:

\[
\text{(Dual): } \max_{\lambda \in Q} p(\lambda) .
\]

(6.4)

Let \( f^* \) denote the optimal value of (6.1). When \( P \) is bounded let \( p^* \) denote the optimal value of (6.4), and the compactness of \( P \) and \( Q \) ensure that weak and strong duality hold: \( p(\lambda) \leq p^* = f^* \leq f(x) \) for all \( \lambda \in Q \) and \( x \in P \). The choice to call (6.1) the primal and (6.4) the dual is of course arbitrary, but this choice is relevant since mirror descent is not symmetric in its treatment of the primal and dual computations.

Note that in the case when \( f(\cdot) \) has minmax structure (6.2), the ability to compute subgradients depends very much on the ability to solve the subproblem in the definition (6.2). Indeed,

\[
\text{if } \bar{\lambda}^k \in \arg \max_{\lambda \in Q} \{(x^k)^T A \lambda\} \text{ , then } g^k \leftarrow A \bar{\lambda}^k \in \partial f(x^k) ,
\]

(6.5)

that is, \( g^k \) is a subgradient of \( f(\cdot) \) at \( x^k \). This fact is very easy to derive, and is also a special case of the more general result known as Danskin’s Theorem, see [9].

The mirror descent method [72, 5] is a generalization of the subgradient descent method. The mirror descent method requires the selection of a differentiable “1-strongly convex” function \( d(\cdot) : P \to \mathbb{R} \) which is defined to be a function with the following (strong) convexity property:

\[
d(x) \geq d(y) + \nabla d(y)^T (x - y) + \frac{1}{2} \|x - y\|^2 \text{ for all } x, y \in P ,
\]

where \( \| \cdot \| \) is again an arbitrary norm on \( \mathbb{R}^n \) (not necessarily Euclidean). The function \( d(\cdot) \)
is typically called the “prox function.” The given prox function \( d(\cdot) \) is also used to define a distance metric:

\[
D(x, y) := d(x) - d(y) - \nabla d(y)^T (x - y) \geq \frac{1}{2} \| x - y \|^2 \text{ for all } x, y \in P . \tag{6.6}
\]

One can think of \( D(x, y) \) as a not-necessarily-symmetric generalization of the distance metric induced by the given norm, in that \( D(x, y) \geq \frac{1}{2} \| x - y \|^2 \geq 0 \), \( D(x, y) = 0 \) if and only if \( x = y \), but it is not generally true (nor is it useful) that \( D(x, y) = D(y, x) \). \( D(x, y) \) is called the Bregman function or the Bregman distance. With these objects in place, the mirror descent method for solving (6.1) is presented in Algorithm 6.1.

**Algorithm 6.1** Mirror Descent Method (applied to problems with minmax structure)

Initialize at \( x^0 \in P, \lambda^0 = 0, k = 0 \)

At iteration \( k \):

1. Compute:
   \[
   \tilde{\lambda}^k \leftarrow \arg \max_{\lambda \in Q} (x^k)^T A \lambda
   \]
   \[
   g^k \leftarrow A \tilde{\lambda}^k
   \]

2. Choose \( \alpha_k \geq 0 \) and set:
   \[
   x^{k+1} \leftarrow \arg \min_{x \in P} \left\{ \alpha_k (g^k)^T x + D(x, x^k) \right\}
   \]
   \[
   \lambda^{k+1} \leftarrow \frac{\sum_{i=0}^{k} \alpha_i \tilde{\lambda}^i}{\sum_{i=0}^{k} \alpha_i}
   \]

The sequence \( \{\lambda^k\} \) constructed in the last line of step (2.) of mirror descent plays no role in the actual dynamics of Algorithm 6.1 and so could be ignored; however \( \lambda^k \) is a feasible solution to the dual problem (6.4) and we will see that the sequence \( \{\lambda^k\} \) has precise computational guarantees with respect to problem (6.4). Note also that the more general mirror descent method (without minmax structure) replaces step (1.) with “compute \( g_k \in \partial f(x_k) \)” and, of course, disregards the dual sequence \( \{\lambda^k\} \). The construction of \( x^{k+1} \) in step (2.) of mirror descent involves the solution of an optimization subproblem; this problem
is often called the projection problem. Notice in general that the prox function \( d(\cdot) \) should be chosen so that this subproblem can be easily solved, i.e., in closed form or with a very efficient algorithm.

Note that the subgradient descent method is a special case of mirror descent using the “Euclidean” prox function \( d(x) := \frac{1}{2} \| x \|_2^2 \). With this choice of prox function, step (2.) of Algorithm 6.1 becomes:

\[
x^{k+1} \leftarrow \arg \min_{x \in P} \left\{ (\alpha_k g^k - x^k)^T x + \frac{1}{2} x^T x \right\} = \Pi_P(x^k - \alpha_k g^k),
\]

(since \( D(x, x^k) = (-x^k)^T x + \frac{1}{2} x^T x + \frac{1}{2} \| x^k \|_2^2 \)), and we recover the subgradient descent method with step-size sequence \( \{\alpha_k\} \). Indeed, the sequence \( \{\alpha_k\} \) in the mirror descent method is called the “step-size” sequence in light of the analogy to subgradient descent. Below we present an example of a version of mirror descent with a prox function that is not Euclidean, which will be used in the analysis of the algorithm AdaBoost in Section 6.2.

**Example 6.1. Multiplicative Weight Updates for Optimization on the Standard Simplex in \( \mathbb{R}^n \)**

Consider optimizing \( f(x) \) on \( P := \Delta_n := \{ x \in \mathbb{R}^n : e^T x = 1, x \geq 0 \} \), the standard simplex in \( \mathbb{R}^n \), and let \( d(x) = e(x) := \sum_{i=1}^n x_i \ln(x_i) + \ln(n) \) be the entropy prox function (note that we define \( 0 \ln(0) := 0 \)). It is well-known that \( e(\cdot) \) is a 1-strongly convex function on \( \Delta_n \) with respect to the \( \ell_1 \) norm, see for example [74] for a short proof. Given any \( c \in \mathbb{R}^n \), it is straightforward to verify that the optimal solution \( \bar{x} \) of a problem of format \( \min_{x \in P} \{ c^T x + d(x) \} \) is given by:

\[
\bar{x}_i = \frac{\exp(-c_i)}{\sum_{i=1}^n \exp(-c_i)} \quad i = 1, \ldots, n.
\]  

(6.7)

Using the entropy prox function, it follows that for each \( i \in \{1, \ldots, n\} \), the update of \( x^k \) in
Step (2.) of Algorithm 6.1 assigns:

\[ x_{i}^{k+1} \propto \exp(-\alpha_{k}g_{i} - \nabla e(x^{k})) = \exp(1 + \ln(x_{i}^{k}) - \alpha_{k}g_{i}^{k}) \propto x_{i}^{k} \cdot \exp(-\alpha_{k}g_{i}^{k}), \]

which is an instance of the multiplicative weights update rule \[1\].

We now state two well-known complexity bounds for the mirror descent method (Algorithm 6.1), see for example \[5\]. In the general case we present a bound on the optimality gap of the sequence \( \{x^{k}\} \) for the primal problem (6.1) that applies for any step-size sequence \( \{\alpha_{k}\} \), and in the case when \( P \) is compact we present a similar bound on the duality gap of the sequences \( \{x^{k}\} \) and \( \{\lambda^{k}\} \). Both bounds can be specified to \( O\left(\frac{1}{\sqrt{k}}\right) \) rates for suitably chosen step-sizes.

**Theorem 6.1. (Complexity of Mirror Descent\[5, 80, 73\])** Let \( \{x^{k}\} \) and \( \{\lambda^{k}\} \) be generated according to the Mirror Descent method (Algorithm 6.1). Then for each \( k \geq 0 \) and for any \( x \in P \), the following inequality holds:

\[
\min_{i \in \{0, \ldots, k\}} f(x^{i}) - f(x) \leq \frac{D(x, x^{0}) + \frac{1}{2}L_{f}^{2} \sum_{i=0}^{k} \alpha_{i}^{2}}{\sum_{i=0}^{k} \alpha_{i}}. \tag{6.8}
\]

If \( P \) is compact and \( \bar{D} \geq \max_{x \in P} D(x, x^{0}) \), then for each \( k \geq 0 \) the following inequality holds:

\[
\min_{i \in \{0, \ldots, k\}} f(x^{i}) - p(\lambda^{k+1}) \leq \frac{\bar{D} + \frac{1}{2}L_{f}^{2} \sum_{i=0}^{k} \alpha_{i}^{2}}{\sum_{i=0}^{k} \alpha_{i}}. \tag{6.9}
\]

These bounds are quite general; one can deduce specific bounds, for example, by specifying a step-size sequence \( \{\alpha_{k}\} \), a prox function \( d(\cdot) \), a value of \( x \) in (6.8) such as \( x = x^{*} \), etc., see for example Remark 6.1 where such specifications are illustrated in the case when \( P \) is compact.

**Remark 6.1.** Suppose we a priori fix the number of iterations \( k \) of Algorithm 6.1 and use a
constant step-size sequence:
\[ \alpha_i = \bar{\alpha} = \frac{1}{L_f} \sqrt{\frac{2D}{k+1}} \quad \text{for} \quad i = 0, \ldots, k. \] (6.10)

Then it follows immediately from (6.9) by substituting in (6.10) that
\[ \min_{i \in \{0, \ldots, k\}} f(x^i) - p(\lambda^{k+1}) \leq L_f \sqrt{\frac{2D}{k+1}}. \] (6.11)

Indeed, the bound (6.11) is in fact the best possible bound (up to an absolute constant factor) for a generic subgradient method, see [72].

### 6.2 Greedy Methods for Logistic Regression

We consider the well-known logistic regression problem. We are given data \((x_1, y_1), \ldots, (x_n, y_n)\) where \(x_i \in \mathbb{R}^p\) is the \(i\)th feature vector and \(y_i \in \{-1, +1\}\) is the corresponding label. Let \(X \in \mathbb{R}^{n \times p}\) be the feature matrix where the \(i\)th row of \(X\) is \(x_i\), and let \(Y \in \mathbb{R}^{n \times n}\) be the diagonal matrix with \(Y_{ii} := y_i\), i.e., \(Y := \text{diag}(y)\). The logistic loss function \(L_n(\cdot) : \mathbb{R}^p \to \mathbb{R}\) is defined by:
\[ L_n(\beta) := \frac{1}{n} \sum_{i=1}^{n} \ln \left(1 + \exp \left(-y_i \beta^T x_i\right)\right). \] (6.12)

The logistic regression problem is that of minimizing the logistic loss function \(L_n(\cdot)\) and is formulated as:
\[
\text{LR : } L_n^* := \min_{\beta} \quad L_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} \ln \left(1 + \exp \left(-y_i \beta^T x_i\right)\right) \tag{6.13}
\text{ s.t. } \beta \in \mathbb{R}^p.
\]

Note that \(L_n(\beta) > 0\) for all \(\beta\), hence \(L_n^* \geq 0\) and is therefore finite. Motivated by the greedy coordinate descent approach to boosting [66], Friedman et al. [43] applied the greedy coordinate descent algorithm to logistic regression, and the resulting algorithm is referred to...
as LogitBoost, for which we give a complete description in Algorithm 6.2.

**Algorithm 6.2 LogitBoost: Greedy Coordinate Descent for Logistic Regression problem (6.13)**

Initialize at $\beta^0 = 0, k = 0$

At iteration $k$:

1. Compute Gradient:

   $$w^k_i \leftarrow \frac{1}{1 + \exp(y_i \beta^k)^T x_i}, \ i = 1, \ldots, n$$
   $$g^k \leftarrow -\frac{1}{n} X^T Y w^k.$$  

2. Compute $j_k \in \arg \max_{j \in \{1, \ldots, p\}} \{ |g^k_j| \}$

3. Choose $\alpha_k \geq 0$ and set:

   $$\beta^{k+1} \leftarrow \beta^k + \alpha_k \text{sgn}(g^k_{j_k}) e_{j_k}$$

Regarding Step (1.) of LogitBoost, notice that it follows from elementary rules of calculus that

$$\nabla L_n(\beta) = -\frac{1}{n} X^T Y w$$

where $w_i := \frac{1}{1 + \exp(y_i \beta^k)^T x_i}, \ i = 1, \ldots, n$.

Due to the update scheme in Step (3.), LogitBoost has the following desirable regularization properties for any $k \geq 0$:

$$\|\beta^k\|_1 \leq \sum_{i=0}^{k-1} \alpha_i \quad \text{and} \quad \|\beta^k\|_0 \leq k.$$  

Unlike the least-squares loss function, it is not *a priori* clear if the logistic regression problem (6.13) has an optimal solution. Indeed, in the case when the examples are linearly separable, i.e., there exists a vector $\beta^* \in \mathbb{R}^p$ satsifying $y_i(\beta^*)^T x_i > 0$ for $i = 1, \ldots, n$, then $L_n(\theta | \beta^*) \to 0 = L_n^*$ as $\theta \to +\infty$, and hence (6.13) does not attain its optimum. However, it is well understood that logistic regression is most sensible in the case where the data is not separable, in which case $L_n(\cdot)$ is a relevant performance metric.

In order to develop some useful properties of the logistic loss function, we introduce the following “prox” function $d(\cdot) : [0, 1]^n \to \mathbb{R}$ defined by:
\[ d(w) := \frac{1}{n} \left[ \sum_{i=1}^{n} w_i \ln(w_i) + (1 - w_i) \ln(1 - w_i) \right], \quad (6.15) \]

where \( \alpha \ln(\alpha) := 0 \) for \( \alpha = 0 \). It can be shown that \( d(\cdot) \) is a \( \sigma = \frac{4}{n} \)-strongly convex function with respect to the Euclidean norm \( \|w\| := \|w\|_2 \).

**Lemma 6.1. (Properties of the Logistic Loss Function)** The logistic loss function \( L_n(\cdot) \) has the following properties:

(i) for any \( \beta \in \mathbb{R}^p \) it holds that:

\[
L_n(\beta) = \max_{w \in [0,1]^n} \left\{ -w^T \left[ \frac{1}{n} Y \begin{bmatrix} X \end{bmatrix} \right] \beta - d(w) \right\}, \quad (6.16)
\]

where \( d(\cdot) \) is given by (6.15),

(ii) for any \( \beta \in \mathbb{R}^p \), the unique optimal solution to the maximization problem in (6.16) is:

\[
w^*(\beta)_i = \frac{1}{1 + \exp(y_i \beta^T x_i)} , \quad i = 1, \ldots, n , \quad (6.17)
\]

(iii) \( \nabla L_n(\beta) = -\frac{1}{n} X^T Y w^*(\beta) \) with \( w^*(\beta) \) given by (6.17),

(iv) \( L_n(\cdot) \) is \( L = \frac{1}{4n} \|X\|_{1,2}^2 \)-smooth with respect to the \( \ell_1 \)-norm on \( \mathbb{R}^p \), i.e., it holds that:

\[
\| \nabla L_n(\beta) - \nabla L_n(\beta') \|_\infty \leq \frac{1}{4n} \|X\|_{1,2}^2 \|\beta - \beta'\|_1 \quad \text{for all } \beta, \beta' \in \mathbb{R}^p , \text{ and}
\]

(v) \( L_n(\beta') \leq L_n(\beta) + \nabla L_n(\beta)^T (\beta' - \beta) + \frac{1}{2n} \|X\|_{1,2}^2 \|\beta' - \beta\|_1^2 \quad \text{for all } \beta, \beta' \in \mathbb{R}^p . \)

The proof of Lemma 6.1 is presented in Appendix D.

Motivated by the notion that logistic regression is most appropriate when the data is not linearly separable, we now introduce a novel *condition number* to measure the degree of non-separability of a given dataset. The given data \((x_1, y_1), \ldots, (x_n, y_n)\) is said to be (strictly) separable if there is a \( \beta \in \mathbb{R}^p \) such that \( y_i \beta^T x_i > 0 \) for all \( i = 1, \ldots, n \); otherwise
the data is said to be non-separable. We say that data example \( i \) is strictly misclassified by \( \beta \) if \( y_i \beta^T x_i < 0 \). Recalling that \( Y \) denotes the diagonal matrix whose \( i^{th} \) component is \( y_i \), then with this notation the data is separable if there is a \( \beta \in \mathbb{R}^p \) such that \( YX\beta > 0 \), and data example \( i \) is strictly misclassified by \( \beta \) if \( (YX\beta)_i < 0 \). We say that data example \( i \) is unclassified by \( \beta \) if \( y_i \beta^T x_i = 0 \), and that \( \beta \) is a “null model” if \( YX\beta = 0 \), i.e., \( \beta \) leaves all data examples unclassified. Letting \( S := \{ \beta \in \mathbb{R}^p : YX\beta = 0 \} \) denote the null space of \( YX \), then \( \beta \) is a null model if and only \( \beta \in S \). Notice that \( S \) is non-trivial, i.e., \( S \neq \{0\} \), in the case when \( p > n \).

We say that the data examples are strictly non-separable if it holds that:

\[
YX\beta \neq 0 \quad \Rightarrow \quad YX\beta \not\geq 0 , \quad (6.18)
\]

which says that if \( \beta \) is not a null model then it strictly misclassifies at least one data example. Strict non-separability of the data implies non-separability of the data but is a slightly stronger condition on the data examples.

Let \( S^\perp \) denote the orthogonal subspace of \( S \); \( \beta \) can be decomposed as \( \beta = \beta_1 + \beta_2 \) where \( \beta_1 \in S \) and \( \beta_2 \in S^\perp \). Notice that \( L_n(\beta + d) = L_n(\beta) \) for any \( d \in S \), i.e., the logistic loss function is invariant in \( S \). In this sense all of the “action” in the logistic loss function is taking place in the space \( S^\perp \), and we could focus exclusively on \( \beta \in S^\perp \) if so desired.

Clearly, some data sets might be “more non-separable” than others, so let us now introduce a way to measure the extent to which the data are non-separable. We define the following measure of non-separability:

\[
\text{NSEP}^* := \min_{\beta \in \mathbb{R}^p} \quad \frac{1}{n} \sum_{i=1}^{n} [y_i \beta^T x_i]^- \\
\text{s.t.} \quad \|\beta\|_1 = 1 \quad \(6.19\)
\]

\[
\beta \in S^\perp .
\]
The objective function of (6.19) measures the average (over all data examples) of the extent of misclassification of the model \( \beta \). Because this objective function is positively homogeneous in \( \beta \), we normalize \( \beta \) using the constraint \( \| \beta \|_1 = 1 \). Also, since the objective function is invariant in the null space \( \mathcal{S} \), we restrict attention to \( \mathcal{S}^\perp \). Notice that NSEP* > 0 if and only if the data examples are strictly non-separable. Furthermore, NSEP* will be larger to the extent to which the data are more non-separable. For this reason we call NSEP* the non-separability condition number of the data examples.

Now recall the distance metric introduced in the context of the Greedy Coordinate Descent Method in Chapter 3 in equation (3.6), which in the context of the logistic loss function and LogitBoost is:

\[
\text{Dist}_0 = \max_{\beta : L_n(\beta) \leq \ln(2)} \left\{ \min_{\beta : L_n(\beta^*) = L^*_n} \| \beta - \beta^* \|_1 \right\},
\]

since in particular it holds that \( L_n(\beta^0) = L_n(0) = \ln(2) \). The following proposition shows that the logistic regression problem (6.13) is well-behaved when the data examples are strictly non-separable.

**Proposition 6.1.** When the data examples are strictly non-separable, then the logistic regression problem (6.13) attains its optimum, and

\[
\text{Dist}_0 \leq \frac{\ln(2) + L^*_n}{\text{NSEP}^*} \leq \frac{2\ln(2)}{\text{NSEP}^*}.
\]

**Proof:** Notice that \( \ln(1 + e^{-t}) \geq t^- \) for any \( t \), and hence the objective function of (6.19) satisfies \( L_n(\beta) \geq \frac{1}{n} \sum_{i=1}^n [y_i \beta^T x_i]^- \) for any \( \beta \). It then follows from (6.19) that \( L_n(\beta) \geq \text{NSEP}^* \| \beta \|_1 \) for all \( \beta \in \mathcal{S}^\perp \), which rearranges to:

\[
\| \beta \|_1 \leq \frac{L_n(\beta)}{\text{NSEP}^*} \quad \text{for all } \beta \in \mathcal{S}^\perp.
\]

Now let \( \beta \) satisfy \( L_n(\beta) \leq L_n(0) = \ln(2) \) and write \( \beta = \beta_1 + \beta_2 \) where \( \beta_1 \in \mathcal{S} \) and
\( \beta_2 \in \mathcal{S}^\perp \), whereby it follows from the above that \( \| \beta_2 \|_1 \leq \frac{L_n(\beta_2)}{\text{NSEP}^*} \leq \frac{\ln(2)}{\text{NSEP}^*} \). Therefore \( \beta_2 \) lies in the compact set:

\[
V := \left\{ \hat{\beta} \in \mathbb{R}^p : \| \hat{\beta} \|_1 \leq \frac{\ln(2)}{\text{NSEP}^*}, \; \hat{\beta} \in \mathcal{S}^\perp \right\}
\]

and it further follows from compactness and continuity that (6.13) attains its optimum at some point \( \hat{\beta}^* \in V \subset \mathcal{S}^\perp \).

Now notice that \( \hat{\beta}^* + \beta_1 \) is also an optimal solution of (6.13), whereby

\[
\min_{\beta : L_n(\beta^*) = L_n^*} \| \beta - \beta^* \| \leq \| \beta - (\hat{\beta}^* + \beta_1) \|_1 = \| \beta_2 - \hat{\beta}^* \|_1 \leq \| \beta_2 \|_1 + \| \hat{\beta}^* \|_1 \leq \frac{\ln(2)}{\text{NSEP}^*} + \frac{L_n^*}{\text{NSEP}^*},
\]

from which the proof follows.

We are now ready to present computational guarantees associated with LogitBoost. Recall that \( \lambda_{\text{pmin}}(X^T X) \) denotes the smallest non-zero (and hence positive) eigenvalue of \( X^T X \).

**Theorem 6.2. (Computational Guarantees for LogitBoost)**

**(i)** Suppose that LogitBoost is implemented using the step-size sequence:

\[
\alpha_k := \frac{4n \| \nabla L_n(\beta_k) \|_\infty}{\| X \|_{1,2}^2} \quad \text{for all } k \geq 0. \tag{6.21}
\]

Then for all \( k \geq 0 \) it holds that:

\[
\min_{i \in \{0, \ldots, k\}} \| \nabla L_n(\beta^i) \|_\infty \leq \| X \|_{1,2} \sqrt{\frac{(\ln(2) - L_n^*)}{2n(k + 1)}} , \tag{6.22}
\]

and when the data is strictly non-separable it also holds that:

\[
L_n(\beta_k) - L_n^* \leq \frac{1}{\ln(2) - L_n^*} + \frac{1}{k \cdot n \cdot (\text{NSEP}^*)^2} < \frac{2(\ln(2))^2 \| X \|_{1,2}^2}{k \cdot n \cdot (\text{NSEP}^*)^2} . \tag{6.23}
\]

**(ii)** Suppose that LogitBoost is implemented using an arbitrary given step-size sequence
\{\alpha_k\}. Then for all \(k \geq 0\) it holds that:

\[
\min_{i \in \{0, \ldots, k\}} \|\nabla L_n(\beta^i)\|_\infty \leq \frac{\ln(2) - L_n^* + \|X\|_{1,2}^2}{8n} \sum_{i=0}^{k} \alpha_i^2 \leq \frac{\ln(2) - L_n^* + \|X\|_{1,2}^2}{8n} \sum_{i=0}^{k} \alpha_i.
\]

(6.24)

**Proof:** This is a simple application of Theorem 3.1, Lemma 6.1, and Proposition 6.1. Inequalities (3.9) and (3.10) of Theorem 3.1 present computational guarantees for the greedy coordinate descent algorithm for the step-size rule (3.8) in terms of the initial objective function value (which in this case is \(L_n(\beta^0) = L_n(0) = \ln(2)\)), the Lipschitz constant \(L\), and the distance measure \(\text{Dist}_0\) as defined in (3.6). From Lemma 6.1, part (iv), we can take the Lipschitz constant \(L\) of the gradient of the logistic loss function \(L_n(\cdot)\) to be \(L = \frac{1}{4n}\|X\|_{1,2}^2\) (this is with respect to the \(\ell_1\) norm on \(\mathbb{R}^p\)). And from Proposition 6.1 we have that \(\text{Dist}_0 \leq \frac{2\ln(2)}{\text{NSEP}^*}\). Substituting these values into the step-size formula (3.8) and the computational guarantees (3.9) and (3.10) of Theorem 3.1 yields precisely the step-size rule (6.21) with the two computational guarantees (6.22) and (6.23). Last of all, substituting the above values into (3.7) of Theorem 3.1 yields precisely the bound (6.24). \(\square\)

**Corollary 6.1.**

(i) If LogitBoost is run for \(k \geq 0\) iterations using the constant step-size \(\alpha_i := \frac{2\sqrt{2n\ln(2)}}{\|X\|_{1,2}\sqrt{k+1}}\) for all \(i = 0, \ldots, k\), then:

\[
\min_{i \in \{0, \ldots, k\}} \|\nabla L_n(\beta^i)\|_\infty \leq \frac{\|X\|_{1,2}\sqrt{\ln(2)}}{\sqrt{2n(k + 1)}}.
\]

(6.25)

(ii) If LogitBoost is run using the adaptive step-size \(\alpha_i := \frac{2\sqrt{2n\ln(2)}}{\|X\|_{1,2}\sqrt{i+1}}\), then:

\[
\min_{i \in \{0, \ldots, k\}} \|\nabla L_n(\beta^i)\|_\infty \leq \frac{\|X\|_{1,2}\ln(2)[2 + \ln(k + 1)]}{4\sqrt{2n}(\sqrt{k + 2} - 1)} = O\left(\frac{\|X\|_{1,2}\ln(k)}{\sqrt{nk}}\right).
\]

(6.26)
Proof: Part (i) follows by substituting the step-size directly into the bound (6.24) of Theorem 6.2, using the inequality $L^*_n \geq 0$, and simplifying terms. To prove part (ii), first substitute the step-size rule $\alpha_i := \frac{2\sqrt{2n \ln(2)}}{\|x\|_{1,2} \sqrt{i+1}}$ into the bound (6.24) of Theorem 6.2, and using the inequality $L^*_n \geq 0$ one obtains:

$$
\min_{i \in \{0, \ldots, k\}} \|\nabla L_n(\beta^i)\|_\infty \leq \frac{\|x\|_{1,2} \sqrt{\ln(2)} \left(1 + \sum_{i=0}^{k} \frac{1}{i+1}\right)}{2\sqrt{2n} \left(\sum_{i=0}^{k} \frac{1}{\sqrt{i+1}}\right)} .
$$

The proof is completed by using the integral bounds

$$
1 + \sum_{i=0}^{k} \frac{1}{i+1} \leq 2 + \int_{1}^{k+1} \frac{1}{t} dt = 2 + \ln(k + 1) ,
$$

and

$$
\sum_{i=0}^{k} \frac{1}{\sqrt{i + 1}} = \sum_{i=1}^{k+1} \frac{1}{\sqrt{i}} \geq \int_{1}^{k+2} \frac{1}{\sqrt{t}} dt = 2\sqrt{k + 2} - 2 .
$$

These results are summarized in Table 6.1.

6.2.1 The Boosting Setup

In boosting for binary classification problems, one is interested in building a classifier that accurately predicts a binary attribute $y \in \{-1, +1\}$, associated with an input $x \in \mathcal{X}$, where $\mathcal{X}$ is an arbitrary space of inputs. The training data are $n$ examples $(x_1, y_1), \ldots, (x_n, y_n)$, consisting of pairs of inputs $x_i \in \mathcal{X}$ and responses $y_i \in [-1, +1]$, where the latter are interpreted as confidence-rated attributes. For example, if $y_i < 0$ then $y_i$ is associated with the attribute $y = -1$, and the closer $y_i$ is to $-1$, the more confident one is about the association and vice versa.

Boosting methods in binary classification are premised on the notion of constructing a sequence of classifiers that are linear combinations of base classifiers (also called weak hy-
Step-size Rule | Computational Bound Guarantees
---|---
Greedy gradient: \( \alpha_k := \frac{4n \| \nabla L_n(\beta^k) \|_\infty}{\| X \|_{1,2}} \) | \( \| X \|_{1,2} \sqrt{\frac{\ln(2) - L_n^*}{2n(k+1)}} \) | \( \frac{2(\ln(2))^2 \| X \|_{1,2}^2}{kn(\text{NSEP}^*)^2} \)
Constant: \( \alpha_i := \frac{2\sqrt{2n \ln(2)}}{\| X \|_{1,2}\sqrt{k+i}} \) for \( i = 0, \ldots, k \) | \( \frac{\| X \|_{1,2} \sqrt{\ln(2)}}{\sqrt{2n(k+1)}} \)
Adaptive: \( \alpha_k := \frac{2\sqrt{2n \ln(2)}}{\| X \|_{1,2}\sqrt{k+i}} \) | \( O\left( \frac{\| X \|_{1,2} \ln(k)}{\sqrt{n}k} \right) \)

| Table 6.1: Computational bound guarantees for LogitBoost for various step-size rules based on Theorem 6.2 and Corollary 6.1 |
p

The base classifiers are often simple models such as decision trees or decision stumps, and are treated as arbitrary black-box classifiers in this analysis. In boosting methods, the classifier at the current iteration is modified by adding one new base classifier in order to construct the next iteration, where the new classifier is chosen in a greedy fashion by a weak learner oracle that returns the base classifier that performs best (among a large ground-set of base classifiers) on a particular linear combination of examples.

More formally, the set of base classifiers is \( \mathcal{H} = \{ h_1, \ldots, h_p \} \) where each \( h_j : \mathcal{X} \to [-1, 1] \) so as to allow for confidence-rated prediction. One is often interested in the more practically relevant case where \( p \gg n \). The weak learner \( \mathcal{W}(\cdot) : \mathbb{R}^n \to \{ 1, \ldots, p \} \) returns, for any weight vector \( w \) on the examples, an index \( j^* \) of a base classifier \( h_{j^*} \) in \( \mathcal{H} \) that performs best on the weighted example determined by \( w \). That is, the weak learner \( \mathcal{W}(w) \) computes \( j^* \in \arg \max_{j \in \{1, \ldots, p\}} \sum_{i=1}^{n} w_i y_i h_j(x_i) \) and we write “\( j^* \in \mathcal{W}(w) \)” in a slight abuse of notation. Even though \( p \) may be extremely large, we assume that it is easy to compute an index \( j^* \in \mathcal{W}(w) \) for any \( w \).

For convenience define the feature matrix \( A \in \mathbb{R}^{m \times n} \) component-wise by \( A_{ij} := y_i h_j(x_i) \),
and let $A_j$ denote the $j$th column of $A$. Suppose that $|A_{ij}| \leq 1$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, p$. It then follows that $\|A\|_{1,2} \leq \sqrt{p}$ and the Lipschitz constant $L$ for $\nabla L_n(\cdot)$ satisfies $L \leq \frac{1}{4}$.

In this setting, a boosting algorithm maintains a sequence of classifiers $\{H_k\}$ that are linear combinations of base classifiers in $\mathcal{H}$. Strictly speaking, a linear combination $H = \sum_{j=1}^{p} \beta_j h_j$ of base classifiers in $\mathcal{H}$ is a function from $\mathcal{X}$ into the reals, and the classifier determined by $H$ is $\text{sign}(H)$; however, for simplicity we will refer to the linear combination $H$ as a classifier, and we say that the coefficient vector $\beta \in \mathbb{R}^p$ determines the classifier $H$.

In the regime of a weak learner oracle, let us consider Algorithm 6.3, which is an equivalent representation of LogitBoost. Notice the structural similarity between Algorithm 6.3 and AdaBoost [87]. Indeed, the only structural difference between AdaBoost and Algorithm 6.3 is in the initial choice of weights $w^0$ and in the formula for updating the weights in Step (2).

---

**Algorithm 6.3 LogitBoost formatted to use a Weak Learner**

Initialize at $w^0 = (1/2, \ldots, 1/2), H^0 = 0, k = 0$

At iteration $k$:

1. Compute $j_k \in \mathcal{W}(w^k)$
2. Choose $\alpha_k \geq 0$ and set:

   $H^{k+1} \leftarrow H^k + \alpha_k \text{sgn}((w^k)^T A_{jk}) h_{jk}$

   $w_i^{k+1} \leftarrow \left(1 + \left[\frac{1}{w_i^k} - 1\right] \exp(\alpha_k \text{sgn}((w^k)^T A_{jk}) A_{i, j_k})\right)^{-1}, i = 1, \ldots, n$

---

### 6.2.2 Regularized Logistic Loss Minimization and its Properties

Let us now consider directly minimizing the logistic loss function in the following regularized problem:
\[ L^*_{n, \delta} := \min_{\beta} \quad L_n(\beta) \]
\[
\text{s.t.} \quad \|\beta\|_1 \leq \delta \quad \beta \geq 0 ,
\]

where \( \delta > 0 \) is the regularization parameter, and let \( L^*_{n, \delta} \) denote the optimal objective function value of (6.27).

Algorithm 6.4, which we refer to as the “FW-LogitBoost algorithm,” is an algorithm for solving (6.27), and hence explicitly considers the regularization \( \|\beta\|_1 \leq \delta \). The FW-LogitBoost algorithm is also a specific instantiation of the Frank-Wolfe algorithm [33].

**Algorithm 6.4 FW-LogitBoost Algorithm**

Initialize at \( \beta^0 \geq 0 \) with \( \|\beta^0\|_1 \leq \delta \), \( w^0_i = \left(1 + \exp \left(y_i (\beta^0)^T x_i \right) \right)^{-1}, i = 1, \ldots, n, k = 0 \)

At iteration \( k \):
1. Compute \( j_k \in \mathcal{W}(w^k) \)
2. Choose \( \bar{\alpha}_k \in [0, 1] \) and set:
   \[ \beta_{j_k}^{k+1} \leftarrow (1 - \bar{\alpha}_k)\beta_{j_k}^k + \bar{\alpha}_k \delta \]
   \[ \beta_{j}^{k+1} \leftarrow (1 - \bar{\alpha}_k)\beta_{j}^k, j \neq j_k \]
   \[ w_{i}^{k+1} \leftarrow \left(1 + \left[\frac{1}{\bar{\alpha}_i} - 1\right]^{1-\bar{\alpha}_i} \exp(\bar{\alpha}_i \delta y_i X_{i,j_k}) \right)^{-1}, i = 1, \ldots, n \]

Note that Algorithm 6.4 is structurally quite similar to AdaBoost and the version of LogitBoost presented in Algorithm 6.3, as these algorithms maintain a weight vector \( w^k \) that is used the same way by the weak learner to identify the index \( j_k \) of the classifier to be boosted. The iterates \( \{\beta^k\} \) of Algorithm 6.4 satisfy \( \|\beta^k\|_0 \leq k \) (for \( k \geq 1 \)) and \( \|\beta^k\|_1 \leq \delta \) so long as \( \bar{\alpha}_0 = 1 \). All of these algorithms update the weight vector \( w^k \) in a simple manner.

Furthermore, note that in step (2.) of Algorithm 6.4 we do not actually need to update \( \beta_{j}^{k+1} \) for all coordinates \( j \in \{1, \ldots, p\} \); in fact we only need to update those coordinates which are currently non-zero. As \( p \) is typically very large, this is an advantage.

Theorem 6.3 states a computational guarantee for (a version of) Algorithm 6.4.
Theorem 6.3. (Computational Properties of FW-LogitBoost Algorithm) If we use the FW-LogitBoost algorithm (Algorithm 6.4), which corresponds to the Frank-Wolfe method to solve (6.27), with either the fixed step-size rule \( \bar{\alpha}_k := \frac{2}{k+2} \) or a line-search to determine \( \bar{\alpha}_k \), then for all \( k \geq 1 \) the following inequality holds:

\[
L_n(\beta^k) - L^*_n, \delta \leq \frac{2\delta^2 \|X\|^2_{1,2}}{n(k+3)}, \tag{6.28}
\]

where \( L^*_n, \delta \) is the optimal objective value of (6.27).

Proof. Noting item (iv) of Lemma 6.1 and noting that the diameter of the feasible region of (6.27) is \( 2\delta \), then (6.28) is a very straightforward application of Bound 2.1 in Section 2.3 of Chapter 2.

6.3 AdaBoost and Computational Guarantees for the Margin and the Log-Exponential Loss

Herein we consider the same setup of boosting algorithms with a weak learner oracle as considered in Section 6.2.1. Algorithm 6.5 is the algorithm AdaBoost, which constructs a sequence of weight vectors \( \{w^k\} \) (also called distributions since \( w^k \in \Delta_m = \{w \in \mathbb{R}^m : e^T x = 1, x \geq 0\} \)) and a sequence \( \{H_k\} \) of nonnegative combinations of base classifiers with the intent of designing a classifier \( \text{sign}(H_k) \) that performs significantly better than any individual base classifier in \( \mathcal{H} \).

It follows inductively that \( H_k = \sum_{i=0}^{k-1} \alpha_i h_{j_i} \), and we define the normalization of \( H_k \) as:

\[
\bar{H}_k := \frac{H_k}{\sum_{i=0}^{k-1} \alpha_i} = \sum_{i=0}^{k-1} \frac{\alpha_i h_{j_i}}{\sum_{i=0}^{k-1} \alpha_i}. \tag{6.29}
\]

Let \( \{\hat{\lambda}^k\} \) denote the sequence of coefficient vectors of the un-normalized classifiers \( \{H_k\} \) produced by AdaBoost and let \( \{\lambda^k\} \) denote the sequence of coefficients of the normalized
Algorithm 6.5 AdaBoost

Initialize at $w^0 = (1/m, \ldots, 1/m)$, $H_0 = 0, k = 0$

At iteration $k$:
1. Compute $j_k \in W(w^k)$
2. Choose $\alpha_k \geq 0$ and set:

$$
H_{k+1} \leftarrow H_k + \alpha_k h_{j_k} \\
w_i^{k+1} \leftarrow w_i^k \exp(-\alpha_k y_i h_{j_k}(x_i)) \quad i = 1, \ldots, m,$$

and re-normalize $w^{k+1}$ so that $e^T w^{k+1} = 1$

classifiers $\{\bar{H}_k\}$, whereby

$$
\hat{\lambda}^k = \sum_{i=0}^{k-1} \alpha_i e_{j_i}, \quad \lambda^k = \frac{\sum_{i=0}^{k-1} \alpha_i e_{j_i}}{\sum_{i=0}^{k-1} \alpha_i}
$$

(6.30)

Due to the update scheme in Step (2.), AdaBoost has the following desirable regularization properties for any $k \geq 0$:

$$
\|\hat{\lambda}^k\|_1 \leq \sum_{i=0}^{k-1} \alpha_i \quad \text{and} \quad \|\hat{\lambda}^k\|_0 = \|\lambda^k\|_0 \leq k.
$$

(6.31)

For convenience define the feature matrix $A \in \mathbb{R}^{m \times n}$ componentwise by $A_{ij} := y_i h_j(x_i)$, and let $A_j$ denote the $j^{th}$ column of $A$. The margin achieved by the coefficient vector $\lambda$ on example $i$ is $(A\lambda)_i$, whereby

$$
p(\lambda) := \min_{i \in \{1, \ldots, m\}} (A\lambda)_i = \min_{w \in \Delta_m} w^T A\lambda
$$

(6.32)

is the least margin achieved by $\lambda$ over all examples; $p(\lambda)$ is simply referred to as the margin of $\lambda$ and/or the margin of the classifier determined by $\lambda$. Because $p(\lambda)$ is positively homogeneous ($p(\gamma \lambda) = \gamma p(\lambda)$ for $\gamma \geq 0$), it makes sense to normalize $\lambda$ when measuring the margin, which we do by re-scaling $\lambda$ so that $\lambda \in \Delta_n$. The optimization problem of maximizing the margin

171
over all normalized coefficient vectors $\lambda$ is:

$$M : \quad \rho^* := \max_{\lambda \in \Delta_n} p(\lambda) . \tag{6.33}$$

Note that it is without loss of generality that we assume $\lambda \geq 0$ since for any base classifier $h_j \in \mathcal{H}$ one may add the classifier $-h_j$ to the set $\mathcal{H}$ if necessary. We therefore assume as well that the set $\mathcal{H}$ is closed under negation, from which it follows that $\rho^* \geq 0$.

In addition to the margin function $p(\lambda)$, it will also be useful to consider the log-exponential loss function $L_l(\cdot) : \mathbb{R}^n \to \mathbb{R}$, which is defined as:

$$L_l(\lambda) = \ln \left( \frac{1}{m} \sum_{i=1}^m \exp \left( -(A\lambda)_i \right) \right) . \tag{6.34}$$

The function $L_l(\cdot)$ is a smooth convex function, and it is well-known that $L_l(\cdot)$ and $p(\cdot)$ are related by:

$$- p(\lambda) - \ln(m) \leq L_l(\lambda) \leq - p(\lambda) \quad \text{for any } \lambda \in \mathbb{R}^n , \tag{6.35}$$

see Lemma D.1 which describes a variety of other useful properties of $L_l(\cdot)$ as well. Unlike the margin function $p(\cdot)$, $L_l(\cdot)$ is not positively homogeneous. We therefore consider the problem of minimizing $L_l(\cdot)$ over all (nonnegative) coefficient vectors $\lambda$:

$$\text{LE} : \quad L_l^* := \min_{\lambda \geq 0} \quad L_l(\lambda) = \log \left( \frac{1}{m} \sum_{i=1}^m \exp \left( -(A\lambda)_i \right) \right) . \tag{6.36}$$

Note that the nonnegativity condition on $\lambda$ in (6.36) is without loss of generality since it is assumed that $\mathcal{H}$ is closed under negation. Unlike the least-squares loss function, it is not a priori obvious that (6.36) has an optimal solution. Indeed, in the case when the examples are separable, then $\rho^* > 0$ and there is a vector $\lambda^* \in \Delta_n$ satisfying $A\lambda^* > 0$ ($\lambda^*$ separates the data since since $A\lambda^* > 0$), and $L_l(\alpha\lambda^*) \to -\infty$ as $\alpha \to +\infty$. In the case when the examples are not separable, then $\rho^* = 0$ and (6.35) implies that $L_l^* \geq - \ln(m)$; nevertheless there is no guarantee that there exists a coefficient vector $\lambda^*$ attaining $L_l(\lambda^*) = L_l^*$. Note also that when
\( \rho^* = 0 \), achieving the maximum margin is trivial; for example the classifier \( \frac{1}{2}h_1 + \frac{1}{2}(-h_1) \) achieves the optimal margin, but this solution is not informative in any practical sense.

The above discussion suggests a natural division in the analysis of the binary classification problem in general and also for the analysis of the algorithm AdaBoost, namely to apply different analyses to the case of separable data and non-separable data. In the separable data case, where \( \rho^* > 0 \), it is both intuitively and theoretically desirable to find a classifier with high (or maximum) margin \([8], \) i.e., one whose margin is close to the optimal value \( \rho^* \); in this case it is not very sensible to minimize the log-exponential loss as \( L_t(\cdot) \) can be driven to \(-\infty\) along any direction with positive margin as discussed above. In the non-separable data case, where \( \rho^* = 0 \), the margin is no longer informative, but the log-exponential loss function \( L_t(\cdot) \) is a relevant performance metric. For a given coefficient vector \( \lambda \geq 0 \), let \( g(\lambda) \) denote the misclassification error of the classifier determined by \( \lambda \), which is the fraction of the examples that are misclassified by the classifier determined by \( \lambda \). It is well known (and also simple to derive) that
\[
L_t(\lambda) > \log(g(\lambda)) ,
\]
whereby lower values of \( L_t(\cdot) \) will guarantee lower misclassification error.

We examine two interpretations of AdaBoost that will enable us to analyze the convergence and derive precise computational guarantees for AdaBoost in both the separable and non-separable data cases. The first is the well-known interpretation of AdaBoost as greedy coordinate descent (Algorithm 3.2) to minimize the exponential loss or, equivalently, to minimize \( L_t(\lambda) \) in \([6.36], [66], [43] \). We review this interpretation in the following proposition, for which we include a proof for completeness.

**Proposition 6.2. (AdaBoost as greedy coordinate descent \([66], [43] \))** The sequence of un-normalized coefficient vectors \( \{\hat{\lambda}^k\} \) of the classifiers \( H_k \) in AdaBoost \((6.30) \) arise as the sequence of primal variables in greedy coordinate descent (Algorithm 3.2) applied to the log-exponential loss minimization problem \((6.36) \) with step-size sequence \( \{\alpha_k\} \) and initialized at \( \hat{\lambda}^0 = 0 \).
Proof: We first show the following relationship between the iterate values of $w^k$ and the coefficient vectors $\hat{\lambda}^k$ at any iteration $k \geq 0$ of AdaBoost:

$$w_i^k = \frac{\exp(-(A\hat{\lambda})_i)}{\sum_{\ell=1}^m \exp(-(A\lambda)_{\ell})}, \quad i = 1, \ldots, m.$$  \hspace{1cm} (6.37)

Clearly (6.37) holds for $k = 0$, since $\hat{\lambda}^0 = 0$ and $w^0 = (1/m, \ldots, 1/m)$. By way of induction, suppose that (6.37) holds at iteration $k$. Then by the update in step (2.) of AdaBoost as well as (6.30) we have that $\hat{\lambda}^{k+1} = \hat{\lambda}^k + \alpha_k e_{j_k}$, and for each $i = 1, \ldots, m$ it holds that:

$$w_i^{k+1} \propto w_i^k \exp(-\alpha_k A_{ij_k})$$

$$\propto \exp(-(A\hat{\lambda})_i) \cdot \exp(-\alpha_k A_{ij_k})$$

$$= \exp(-(A(\hat{\lambda}^k + \alpha_k e_{j_k}))_i)$$

$$= \exp(-(A\hat{\lambda}^{k+1})_i),$$

which demonstrates (6.37) for all $k$ by induction. It then follows from basic properties of $L_l(\cdot)$ in (D.3) of Lemma D.1 that

$$\nabla L_l(\hat{\lambda}) = -A^T w^k.$$  \hspace{1cm} (6.38)

By definition of the weak learner and the fact that $\mathcal{H}$ is closed under negation, we have:

$$j_k \in \mathcal{W}(w^k) \implies j_k \in \text{arg max}_{j \in \{1,\ldots,n\}} |(w^k)^T A_j| \iff j_k \in \text{arg max}_{j \in \{1,\ldots,n\}} |\nabla L_l(\hat{\lambda})_j|.$$ 

Thus, in step (1.) of AdaBoost, the weak learner is computing the same index $j_k$ as greedy coordinate descent. Since $\mathcal{H}$ is closed under negation we have $(w^k)^T A_{j_k} \geq 0$, which implies that $\nabla L_l(\hat{\lambda})_{j_k} \leq 0$ and $\hat{\lambda}^{k+1} \leftarrow \hat{\lambda}^k - \alpha_k \cdot \text{sgn}(\nabla L_l(\hat{\lambda})) \cdot e_{j_k}$, which is exactly the greedy coordinate descent update rule. \qed

Furthermore, Mukherjee et al. [70] give precise convergence rates for (6.36), for the version of AdaBoost with step-sizes determined by a line-search, see also Telgarsky [92].
Armed with the computational guarantees of greedy coordinate descent in Theorem 3.1, the equivalence of AdaBoost to greedy coordinate descent can be used to give precise computational guarantees for AdaBoost. Towards this goal, using the $\ell_1$ norm on $\mathbb{R}^n$, note from Lemma D.1 part (iii) that $L_l(\cdot)$ is $L$-smooth with Lipschitz constant $L = 1$. Consider the step-size rule:

$$\alpha_k = |(A^T w^k)_j| \quad \text{for } k \geq 0 .$$ (6.39)

The quantity $(A^T w^k)_j$ is called the “edge” of classifier $j_k$ under the empirical distribution $w^k$, so this is called the “edge” step-size rule. Notice from the definition of the weak learner $\mathcal{W}(\cdot)$ and Step (1.) of AdaBoost that

$$|(A^T w^k)_j| = \|A^T w^k\|_\infty = \|\nabla L_l(\hat{\lambda}^k)\|_\infty$$ (6.40)

(where the second equality above uses (6.38)), and therefore the “edge” step-size rule corresponds exactly to the step-size rule (3.8). We therefore can apply Theorem 3.1. Applying (3.9) we obtain:

$$\min_{i \in \{0, \ldots, k\}} \|\nabla L_l(\hat{\lambda}^i)\|_\infty \leq \sqrt{\frac{2(L_l(\hat{\lambda}^0) - L_l^*)}{k + 1}},$$

since in particular we have $L = 1$, and where $L_l^* := \inf_{\lambda} L_l(\lambda)$. Then notice that $L_l(\hat{\lambda}^0) = L_l(0) = 0$. In the non-separable data case we also have from (6.35) that $L_l^* \geq -\ln(m) - \inf_{\lambda} p(\lambda) \geq -\ln(m)$, since the non-separability implies that $p(\lambda) \leq 0$ for any $\lambda$. Substituting these inequalities yields

$$\min_{i \in \{0, \ldots, k\}} \|\nabla L_l(\hat{\lambda}^i)\|_\infty \leq \sqrt{\frac{2\ln(m)}{k + 1}} .$$

This bound shows that AdaBoost with the “edge” step-size rule is driving the gradient of the log-exponential loss function to zero, with a specific computational guarantee, and therefore if the sequence $\{\hat{\lambda}^k\}$ is either bounded or has a convergent subsequence, then a subsequence of the iterates of AdaBoost are converging to an optimal solution of the log-exponential loss problem. We formally record this bound in the fourth column the first row of Table 6.2.
which presents, in summary form, computational guarantees for various step-size rules and for the separable data and non-separable data cases that we now further describe, develop, and discuss.

<table>
<thead>
<tr>
<th>Step-Size Strategy</th>
<th>Separable Data</th>
<th>Non-Separable Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Margin Bound</td>
<td>Gradient Bound</td>
</tr>
<tr>
<td></td>
<td>$\rho^* - p(\lambda^{k+1})$</td>
<td>$\min_{i \in {0, ..., k}} | \nabla L_\ell(\hat{\lambda}<em>i) |</em>\infty$</td>
</tr>
<tr>
<td>“edge rule:”</td>
<td>$\alpha_k = | \nabla L_\ell(\hat{\lambda}^k) |_\infty$</td>
<td>$\sqrt{\frac{2\ln(m)}{k+1}}$</td>
</tr>
<tr>
<td>“line-search:”</td>
<td>$\alpha_k = \frac{1}{2} \ln \left( \frac{1+r_k}{1-r_k} \right)$</td>
<td>$\sqrt{\frac{2\ln(m)}{k+1}}$</td>
</tr>
<tr>
<td>“constant:”</td>
<td>$\alpha_i := \sqrt{\frac{2\ln(m)}{k+1}}$ for $i = 0, \ldots, k$</td>
<td>$\sqrt{\frac{2\ln(m)}{k+1}}$</td>
</tr>
<tr>
<td>“adaptive:”</td>
<td>$\alpha_k = \sqrt{\frac{2\ln(m)}{k+1}} \sqrt{\frac{\ln(m)}{2} \left[ 2+\ln(k+1) \right]} \frac{2(\sqrt{k}+2-1)}{2(\sqrt{k}+2-1)}$</td>
<td>$\sqrt{\frac{2\ln(m)}{k+1}} \sqrt{\frac{\ln(m)}{2} \left[ 2+\ln(k+1) \right]} \frac{2(\sqrt{k}+2-1)}{2(\sqrt{k}+2-1)}$</td>
</tr>
</tbody>
</table>

**Table 6.2:** Computational Guarantees for Various Step-size Strategies. Note that the line-search rule pertains only to the case when the base classifiers are not confidence-rated, i.e., when $y_i \in \{-1, 1\}$.

We now turn to the issue of bounding the log-exponential loss optimality gap of the iterates of AdaBoost. Let $L_\ell^*$ denote the optimal objective value of (6.36). Continuing to analyze AdaBoost through the lens of greedy coordinate descent, we would like to invoke the bound (3.10) of Theorem 3.1 and assert that $L_\ell(\hat{\lambda}^k) - L_\ell^* \leq \frac{\text{Dist}_0^2}{k}$, which would be a bound on the optimality gap of the log-exponential loss optimality gap for the iterates of AdaBoost, and would show that the loss values converge to $L_\ell^*$ at the given rate. However, in order to invoke this bound we must first demonstrate that $L_\ell^*$ is finite and also demonstrate that Dist$_0$ is finite. Let us now see how this can be accomplished.

For this purpose, it will be useful to define a non-separability measure for the log-
exponential loss that is analogous to the nonseparability measure for the logistic loss function (6.19). Recall the notion of strict non-separability. We say that data example \( i \) is \textit{strictly mis-classified} by the classifier \( H = \sum_{j=1}^{n} \lambda_j h_j \) if \( (A\lambda)_i < 0 \). We say that the data examples are \textit{strictly non-separable} if it holds that:

\[
A\lambda \neq 0 \Rightarrow A\lambda \nless 0 .
\] (6.41)

In other words, strict non-separability means that any “non-null” classifier strictly misclassifies at least one example. Put yet another way, strict non-separability means that if \( A\lambda \geq 0 \), then \( A\lambda = 0 \). Let \( \mathcal{S} := \{ \lambda \in \mathbb{R}^n : A\lambda = 0 \} \) denote the null space of \( A \) and let \( \mathcal{S}^\perp \) denote its orthogonal subspace. We define the following measure of non-separability:

\[
\text{NSEP}_i^* := \min_{\lambda \in \mathbb{R}^n} \left[ -p(\lambda) = \max_{i \in \{1, \ldots, m\}} (A\lambda)_i \right] \\
\text{s.t.} \quad \|\lambda\|_1 = 1 \\
\lambda \in \mathcal{S}^\perp .
\] (6.42)

Now recall the distance metric introduced in (3.6), which in the context of the log-exponential loss function is:

\[
\text{Dist}_0 = \max_{L_i(\lambda) \leq 0} \left\{ \min_{L_i(\lambda^*) = L_i^*} \|\lambda - \lambda^*\|_1 \right\} ,
\] (6.43)

since in particular it holds that \( L_i(\hat{\lambda}^0) = L_i(0) = 0 \). Note that if \( L_i^* \) is not finite, i.e., \( L_i^* = -\infty \), then \( \{\lambda^* : L_i(\lambda^*) = L_i^*\} \) is empty and hence \( \text{Dist}_0 = \infty \). The following proposition shows that the log-exponential loss problem (6.36) is well-behaved when the data is strictly non-separable.

**Proposition 6.3.** When the data is strictly non-separable, then \( L_i^* \) is finite, (6.36) attains
its optimum, and
\[
\text{Dist}_0 \leq \frac{2\ln(m) + L_1^*}{\text{NSEP}_l^*} \leq \frac{2\ln(m)}{\text{NSEP}_l^*}.
\] (6.44)

**Proof:** First note that when the data is not separable, then \(p(\lambda) \leq 0\) for any \(\lambda\), therefore from part \((ii)\) of Lemma D.1 it holds that \(L_l(\lambda) \geq -p(\lambda) - \ln(m) \geq -\ln(m)\). It follows that \(L_1^* \geq -\ln(m)\) and hence is finite.

By the definition of \(\text{NSEP}_l^*\) in (6.42), it holds that \(-p(\lambda) \geq \text{NSEP}_l^*||\lambda||_1\) for all \(\lambda \in \mathcal{S}^\perp\). Therefore, again using part \((ii)\) of Lemma D.1, it holds that:

\[
||\lambda||_1 \leq \frac{L_l(\lambda) + \ln(m)}{\text{NSEP}_l^*} \quad \text{for all } \lambda \in \mathcal{S}^\perp.
\]

Now let \(\lambda\) satisfy \(L_l(\lambda) \leq L_l(0) = 0\) and write \(\lambda = \lambda_1 + \lambda_2\) where \(\lambda_1 \in \mathcal{S}\) and \(\lambda_2 \in \mathcal{S}^\perp\), whereby it follows from the above that \(||\lambda_2||_1 \leq \frac{L_l(\lambda_2) + \ln(m)}{\text{NSEP}_l^*} \leq \frac{\ln(m)}{\text{NSEP}_l^*}\). Therefore \(\lambda_2\) lies in the compact set:

\[
V := \left\{ \hat{\lambda} \in \mathbb{R}^n : ||\hat{\lambda}||_1 \leq \frac{\ln(m)}{\text{NSEP}_l^*}, \hat{\lambda} \in \mathcal{S}^\perp \right\},
\]

and it further follows from compactness and continuity that (6.36) attains its optimum at some point \(\hat{\lambda}^* \in V \subset \mathcal{S}^\perp\).

Now notice that \(\hat{\lambda}^* + \lambda_1\) is also an optimal solution of (6.36), whereby

\[
\min_{L_l(\lambda^*)=L_l^*} ||\lambda - \lambda^*|| \leq ||\lambda - (\hat{\lambda}^* + \lambda_1)||_1 = ||\lambda_2 - \hat{\lambda}^*||_1 \leq ||\lambda_2||_1 + ||\hat{\lambda}^*||_1 \leq \frac{\ln(m)}{\text{NSEP}_l^*} + \frac{L_1^* + \ln(m)}{\text{NSEP}_l^*},
\]

from which (6.44) follows. \(\blacksquare\)

It therefore follows from the interpretation of AdaBoost as greedy coordinate descent and bound (3.10) of Theorem 3.1 that if the data is strictly non-separable, then

\[
L_l(\hat{\lambda}^k) - L_l^* \leq \frac{2\text{Dist}_0^2}{k} \leq \frac{8\ln(m)^2}{(\text{NSEP}_l^*)^2k}.
\]

This result is presented in the last column of the second row of Table 6.2. In particular, under strict non-separability of the data, this bounds the optimality gap of the log-exponential loss.
for the iterates of AdaBoost, and shows that the loss values converge to \( L^*_l \) at the given rate.

Continuing with the interpretation of AdaBoost as greedy coordinate descent, it then seems natural to determine the step-size \( \alpha_k \) at iteration \( k \) of AdaBoost by exact line-search of the loss function, i.e., determine:

\[
\alpha_k \leftarrow \arg \min_{\alpha} L_l(\hat{\lambda}^k + \alpha e_{j_k}).
\]

When the base classifiers are not confidence-rated, so that \( h_j(x_i) \in \{-1, +1\} \), then it is well-known that the solution to the above line-search problem is \( \alpha_k \leftarrow \frac{1}{2} \ln \left( \frac{1 + r_{j_k}}{1 - r_{j_k}} \right) \) where recall from above that \( r_{j_k} := (A^T w^k)_{j_k} \) is the “edge” of classifier \( j_k \) under the empirical distribution \( w^k \). (See, for example, Mukherjee et al. [70] for an exposition.) Because the computational guarantees for greedy coordinate descent also hold when an exact line-search is used (see Remark 3.1), the bounds for the “edge” step-size rule also hold when an exact line-search is used to determine the step-sizes. These results are summarized in Table 6.2.

The second interpretation of AdaBoost that will be of use to us is as an instance of the mirror descent algorithm (Algorithm 6.1) applied directly to a certain optimization problem that is a dual problem of the margin maximization problem (6.33), which we now develop. For any distribution \( w \in \Delta_m \), define the edge of classifier \( h_j \) with respect to \( w \) to be \( w^T A_j \), whereby

\[
f(w) := \max_{j \in \{1, \ldots, n\}} w^T A_j = \max_{\lambda \in \Delta_n} w^T A \lambda
\]

is the maximum edge over all base classifiers; \( f(w) \) is called the edge with respect to \( w \). Recalling from Remark 6.2 that the sequence of weight vectors \( \{w^k\} \) produced by AdaBoost are related to the gradient of the log-exponential loss by \( \nabla L_l(\hat{\lambda}^k) = -A^T w^k \) and using the fact that \( \mathcal{H} \) is closed under negation, we have the following relationship:

\[
f(w^k) = \max_{j \in \{1, \ldots, n\}} (w^k)^T A_j = \max_{j \in \{1, \ldots, n\}} |(A^T w^k)_j| = \|A^T w^k\|_\infty \text{ for each } k \geq 0.
\]

179
The optimization problem of minimizing the edge over all distributions $w$ is:

$$E : f^* := \min_{w \in \Delta_m} f(w), \quad (6.47)$$

and in fact is the (linear programming) dual of the maximum margin problem (6.33). Indeed, (6.45) and (6.47) are precisely in the format of (6.2) and (6.1) with $w$ playing the role of the primal variables $x$, $P = \Delta_m$, and $Q = \Delta_n$. We establish the following equivalence between AdaBoost and mirror descent (Algorithm 6.1) applied to (6.47).

**Theorem 6.4. (AdaBoost as Mirror Descent)** The sequence of weight vectors $\{w^k\}$ in AdaBoost arise as the sequence of primal variables in mirror descent applied to the primal problem (6.47), using the entropy prox function $d(w) := e(w) = \sum_{i=1}^m w_i \ln(w_i) + \ln(m)$, with step-size sequence $\{\alpha_k\}$ and initialized at $w^0 = (1/m, \ldots, 1/m)$. Furthermore, the coefficients $\{\lambda^k\}$ of the normalized classifiers $\tilde{H}_k$ produced by AdaBoost arise as the sequence of dual variables in mirror descent.

**Proof:** By definition of the weak learner and (6.45) combined with (6.5), we have for any $w \in \Delta_m$

$$j^* \in \mathcal{W}(w) \iff j^* \in \arg \max_{j \in \{1, \ldots, n\}} w^T A_j \iff e_{j^*} \in \arg \max_{\lambda \in \Delta_n} w^T A\lambda \iff A_{j^*} \in \partial f(w),$$

whereby step (1.) of AdaBoost is identifying a vector $g^k := A_{j^*} \in \partial f(w^k)$. Moreover, since $g^k_i = y_i h_{j^k}(x_i) = A_{i,j^k}$, the construction of $w^{k+1}$ in step (2.) of AdaBoost is exactly setting $w^{k+1} \leftarrow \arg \min_{w \in \Delta_m} \{\alpha_k (g^k)^T x + D(w, w^k)\}$ (where $D(\cdot, \cdot)$ is the Bregman distance function arising from the entropy function), as discussed in Example 6.1. Therefore the sequence $\{w^k\}$ is a sequence of primal variables in Mirror Descent with the entropy prox function. Also notice from step (1.) of AdaBoost and the output of the weak learner $\mathcal{W}(w^k)$ that $e_{j^k} \in \arg \max_{\lambda \in \Delta_n} (w^k)^T A\lambda$, which gives the correspondence $\dot{\lambda}^k = e_{j^k}$ at step (1.) of Mirror Descent. It therefore follows that the coefficients $\{\lambda^k\}$ of the normalized classifiers $\{\tilde{H}_k\}$
satisfy:

\[ \lambda^k := \frac{\sum_{i=0}^{k-1} \alpha_i e_{j_i}}{\sum_{i=0}^{k-1} \alpha_i} = \frac{\sum_{i=0}^{k-1} \alpha_i \tilde{\lambda}^i}{\sum_{i=0}^{k-1} \alpha_i}, \]

which implies that \( \{\lambda^k\} \) is exactly the sequence of dual variables appearing in mirror descent.

The equivalence of AdaBoost to the mirror descent algorithm is alluded to in various previous works that relate machine learning methods to convex optimization, see for example [39, 89], although there is no publication that we are aware of that presents the above result nor the proof thereof. In our view, the equivalence of AdaBoost as an instance of the mirror descent algorithm is by itself not so valuable as the implications that the equivalence has for various convergence and computational guarantees. To the best of our knowledge, these implications have not been previously brought to light.

Armed with the computational guarantees of mirror descent in Theorem 6.1, the equivalence of AdaBoost to mirror descent via Theorem 6.4 can also be used to give precise computational guarantees for AdaBoost, in both the separable and non-separable cases, and for a variety of step-size rules. We emphasize that the algorithm AdaBoost does not know \textit{ex ante} whether the data is separable or not; thus, the derivation of computational guarantees in both cases reflects the robustness of the method. We summarize the convergence statements and computational guarantees in the following theorem.

**Theorem 6.5. (Computational Guarantees for AdaBoost)** Consider the un-normalized and normalized classifiers and their coefficient vector sequences \( \{\tilde{\lambda}^k\} \) and \( \{\lambda^k\} \), respectively, produced by AdaBoost using an arbitrary step-size sequence \( \{\alpha_k\} \). Then, at the end of iteration \( k \) of AdaBoost we have:

\[(i) \ (\text{Separable Data}) \text{ If } \rho^* > 0, \text{ then:} \]

\[ \rho^* - p(\lambda^{k+1}) \leq \frac{\ln(m) + \frac{1}{2} \sum_{i=0}^{k} \alpha_i^2}{\sum_{i=0}^{k} \alpha_i} . \]
(ii) (Non-separable Data) If $\rho^* = 0$, then:

$$
\min_{i \in \{0, \ldots, k\}} \|\nabla L_l(\hat{\lambda}^i)\|_\infty \leq \frac{\ln(m) + \frac{1}{2} \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}. \tag{6.49}
$$

**Proof:** In Lemma D.2 it is shown that the edge function $f(\cdot)$ is 1-Lipschitz with respect to the $\ell_1$ norm, and in Lemma D.3 it is shown that $\max_{w \in \Delta_m} D(w, w^0) = \ln(m)$. Thus, Theorem 6.4 along with (6.9) in Theorem 6.1 imply that:

$$
\min_{i \in \{0, \ldots, k\}} f(w^i) - p(\lambda^{k+1}) \leq \frac{\ln(m) + \frac{1}{2} \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}. \tag{6.50}
$$

In the separable case, weak duality implies that $\rho^* \leq \min_{i \in \{0, \ldots, k\}} f(w^i)$, which together with (6.50) implies (6.48). In the non-separable case, $p(\lambda^{k+1}) \leq \rho^* = 0$, which together with (6.38), (6.46) and (6.50) yields:

$$
\min_{i \in \{0, \ldots, k\}} \|\nabla L_l(\hat{\lambda}^i)\|_\infty = \min_{i \in \{0, \ldots, k\}} \|A^T w^i\|_\infty = \min_{i \in \{0, \ldots, k\}} f(w^i) \leq \min_{i \in \{0, \ldots, k\}} f(w^i) - p(\lambda^{k+1}) \leq \frac{\ln(m) + \frac{1}{2} \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}.
$$

In the case of separable data, Theorem 6.5 provides an exact computational guarantee that bounds the optimal margin gap $\rho^* - p(\lambda^k)$ of the classifier $\hat{H}_k$ produced by AdaBoost, for an arbitrary step-size sequence. Similarly, in the case of non-separable data, Theorem 6.5 provides an exact computational guarantee for the $\ell_\infty$ norm of the gradient of $L_l(\cdot)$, thereby guaranteeing the extent to which the classifiers $H_k$ (equivalently $\hat{\lambda}^k$) produced by AdaBoost satisfy the first-order optimality condition for minimizing $L_l(\cdot)$. These bounds apply to an arbitrary step-size sequence $\{\alpha_k\}$.

Utilizing Theorem 6.5 which interprets AdaBoost as an instance of the mirror descent algorithm, we can also obtain various specific computational guarantees for a given specification of a step-size sequence. We now show two illustrative examples.
Consider the “constant” step-size strategy which is described in the third row of Table 6.2. In this strategy, the number of iterations $k$ of AdaBoost is decided in advance, and for each of iteration $i = 0, \ldots, k$ we use the step-size $\alpha_i := \sqrt{\frac{2 \ln(m)}{k+1}}$. Then substituting this constant value into the formula for the margin bound in (6.48) and for the gradient bound in (6.49) yields the value $\sqrt{\frac{2 \ln(m)}{k+1}}$, which is shown in the third row of Table 6.2.

Let us now consider a simple non-constant step-size strategy. The “constant” step-size strategy just discussed is premised on a priori choosing the number of iterations of AdaBoost. A more practical step-size strategy that is a variant of the constant step-size strategy is $\alpha_k := \sqrt{\frac{2 \ln(m)}{k+1}}$, and which is described in the fourth row of Table 6.2. This step-size rule is “adaptive” since the step-size depends on the iteration counter itself. Substituting this step-size sequence into the formula for the margin bound in (6.48) and for the gradient bound in (6.49) yields the bound:

$$\frac{\ln(m) + \frac{1}{2} \sum_{i=0}^{k} \alpha_i^2}{\sum_{i=0}^{k} \alpha_i} = \frac{\sqrt{\frac{\ln(m)}{2}} \left(1 + \sum_{i=0}^{k} \frac{1}{i+1}\right)}{\sum_{i=0}^{k} \frac{1}{\sqrt{i+1}}} \leq \frac{\sqrt{\frac{\ln(m)}{2}} \left[2 + \ln(k+1)\right]}{2(\sqrt{k+2} - 1)},$$

where the last inequality uses the integral bounds:

$$1 + \sum_{i=0}^{k} \frac{1}{i+1} \leq 2 + \int_{1}^{k+1} \frac{1}{t} dt = 2 + \ln(k+1) \quad \text{and} \quad \sum_{i=0}^{k} \frac{1}{\sqrt{i+1}} = \sum_{i=1}^{k+1} \frac{1}{\sqrt{i}} \geq \int_{1}^{k+2} \frac{1}{\sqrt{t}} dt = 2 \sqrt{k+2} - 2.$$

Substituting this upper bound into the inequality for the margin bound in (6.48) and for the gradient bound in (6.49) yields the values shown in the last row of Table 6.2.

### 6.3.1 Regularized Exponential Loss Minimization and its Properties

In the case when the data are not separable, Theorem 6.5 shows that AdaBoost is working on the problem of driving the gradient of the log-exponential loss function to zero. It therefore makes even more sense to consider directly the problem of minimizing the log-exponential...
loss subject to a regularization constraint, as follows:

$$\min_\lambda L_t(\lambda)$$

s.t. \[\|\lambda\|_1 \leq \delta \]
\[\lambda \geq 0\]

(6.51)

where \(\delta > 0\) is the regularization parameter, and let \(L^*_t,\delta\) denote the optimal objective function value of (6.51).

The exponential loss function is \(L_{\exp}(\lambda) := \exp(L_t(\lambda))\) and is a strictly monotone function of the log-exponential loss function, and the problem of minimizing \(L_{\exp}(\lambda)\) has attracted much attention \[66, 70\]. We therefore also consider the following problem which is in a sense equivalent to (6.51):

$$\min_\lambda L_{\exp}(\lambda)$$

s.t. \[\|\lambda\|_1 \leq \delta \]
\[\lambda \geq 0\]

(6.52)

and let \(L^*_\exp,\delta\) denote the optimal objective function value of (6.52).

Algorithm 6.6, which we refer to as the “FW-Boost algorithm,” is an algorithm for solving (6.51), and hence explicitly considers the regularization \(\|\lambda\|_1 \leq \delta\). The FW-Boost algorithm looks on the surface to be very similar to AdaBoost in many respects. The name “FW-Boost” is given in consideration of the Frank-Wolfe algorithm in convex optimization, \[33\], \[63\], and \[80\], of which Algorithm 6.6 is also a specific instantiation.

Note that Algorithm 6.6 is structurally quite similar to AdaBoost: both algorithms maintain a distribution vector \(w^k\) that is used the same way by the weak learner to identify the index \(j_k\) of the classifier to be boosted. Both algorithms share similar sparsity and regularization properties: AdaBoost satisfies (6.31), while the iterates \(\{\lambda^k\}\) of Algorithm 6.6 satisfy \(\|\lambda^k\|_0 \leq k\) (for \(k \geq 1\)) and \(\|\lambda^k\|_1 \leq \delta\) so long as \(\bar{\alpha}_0 = 1\). And both algorithms update the distribution vector \(w^k\) in a similar (but not identical) manner. Furthermore, note that in step (2.) of Algorithm 6.6 we do not actually need to update \(\lambda^k_{j+1}\) for all coordinates.
Algorithm 6.6 FW-Boost Algorithm

Initialize at $\lambda^0 \geq 0$ with $\|\lambda^0\|_1 \leq \delta$, $w_i^0 = \frac{\exp(-(A\lambda^0)_i)}{\sum_{l=1}^m \exp(-(A\lambda^0)_l)}$, $i = 1, \ldots, m$, $k = 0$

At iteration $k$:
1. Compute $j_k \in \mathcal{W}(w^k)$
2. Choose $\bar{\alpha}_k \in [0, 1]$ and set:
   $$
   \begin{align*}
   \lambda_{j_k}^{k+1} &\gets (1 - \bar{\alpha}_k)\lambda_{j_k}^k + \bar{\alpha}_k \delta \\
   \lambda_j^{k+1} &\gets (1 - \bar{\alpha}_k)\lambda_{j_k}^k, j \neq j_k \\
   w_i^{k+1} &\gets \left( w_i^k \right)^{1-\bar{\alpha}_k} \exp(-\bar{\alpha}_k \delta y_i h_{j_k}(x_i)) \quad i = 1, \ldots, m, \text{ and re-normalize } w^{k+1}
   \end{align*}
   $$

and we note that the margin of the classifier $H_k$ is $p(\bar{\lambda}^k)$. Theorem 6.6 states computational guarantees for a version of Algorithm 6.6 with a particular step-size sequence choice.

Theorem 6.6. (Computational Properties of FW-Boost Algorithm) If we use the FW-Boost algorithm (Algorithm 6.6), which corresponds to the Frank-Wolfe method to solve (6.51), with either the fixed step-size rule $\bar{\alpha}_k := \frac{2}{k+2}$ or a line-search to determine $\bar{\alpha}_k$, then for all $k \geq 1$ we have the following inequalities:

$$
L_i(\lambda^k) - L_{*,s}^* \leq \frac{8\delta^2}{k+3}, \quad \rho^* - p(\bar{\lambda}^k) \leq \frac{8\delta}{k+3} + \frac{\ln(m)}{\delta},
$$

where $L_{*,s}^*$ is the optimal objective value of (6.51) and $\rho^*$ is the optimal objective value of (6.51).
Furthermore, for $k \geq \max\{8\delta^2 - 3, 1\}$ we have the following inequality:

$$L_{\exp}(\lambda^k) - L^*_{\exp, \delta} \leq \frac{16\delta^2}{k + 3}, \quad \text{(6.55)}$$

where $L^*_{\exp, \delta}$ is the optimal objective function value of (6.52).
Appendix A

Supplement to Chapter 2

A.1 Frank-Wolfe Bounds in the Presence of Minmax Duality Structure

Proposition A.1. Let $B_w^k$ and $B_m^m$ be as defined in Section 2.2. Suppose that there exists an open set $\hat{Q} \subseteq E$ containing $Q$ such that $\phi(x, \cdot)$ is differentiable on $\hat{Q}$ for each fixed $x \in P$, and that $h(\cdot)$ has the minmax structure (2.4) on $\hat{Q}$ and is differentiable on $\hat{Q}$. Then it holds that:

$$B_w^k \geq B_m^m \geq h^*.$$  

Furthermore, it holds that $B_w^k = B_m^m$ in the case when $\phi(x, \cdot)$ is linear in the variable $\lambda$.

Proof. It is simple to show that $B_m^m \geq h^*$. At the current iterate $\lambda_k \in Q$, define $x_k \in \arg\min_{x \in P} \phi(x, \lambda_k)$. Then from the definition of $h(\lambda)$ and the concavity of $\phi(x_k, \cdot)$ we have:

$$h(\lambda) \leq \phi(x_k, \lambda) \leq \phi(x_k, \lambda_k) + \nabla_{\lambda} \phi(x_k, \lambda_k)^T (\lambda - \lambda_k) = h(\lambda_k) + \nabla_{\lambda} \phi(x_k, \lambda_k)^T (\lambda - \lambda_k), \ (A.1)$$

whereby $\nabla_{\lambda} \phi(x_k, \lambda_k)$ is a subgradient of $h(\cdot)$ at $\lambda_k$. It then follows from the differentiability
of \( h(\cdot) \) that \( \nabla h(\lambda_k) = \nabla \lambda \phi(x_k, \lambda_k) \), and this implies from (A.1) that:

\[
\phi(x_k, \lambda) \leq h(\lambda_k) + \nabla h(\lambda_k)^T(\lambda - \lambda_k) .
\] (A.2)

Therefore we have:

\[
B^m_k = f(x_k) = \max_{\lambda \in Q} \{ \phi(x_k, \lambda) \} \leq \max_{\lambda \in Q} \{ h(\lambda_k) + \nabla h(\lambda_k)^T(\lambda - \lambda_k) \} = B^w_k .
\]

If \( \phi(x, \lambda) \) is linear in \( \lambda \), then the second inequality in (A.1) is an equality, as is (A.2). \( \square \)

### A.2 Smoothness and the Curvature Constant \( C_{h,Q} \)

#### Proposition A.2.

Let \( C_{h,Q} \), \( \text{Diam}_Q \), and \( L_{h,Q} \) be as defined in Section 2.2. Then it holds that \( C_{h,Q} \leq L_{h,Q}(\text{Diam}_Q)^2 \).

**Proof.** Since \( Q \) is convex, we have \( \lambda + \alpha(\bar{\lambda} - \lambda) \in Q \) for all \( \lambda, \bar{\lambda} \in Q \) and for all \( \alpha \in [0, 1] \).

Since the gradient of \( h(\cdot) \) is Lipschitz, from the fundamental theorem of calculus we have:

\[
h(\lambda + \alpha(\bar{\lambda} - \lambda)) = h(\lambda) + \nabla h(\lambda)^T(\alpha(\bar{\lambda} - \lambda)) + \int_0^1 [\nabla h(\lambda + t\alpha(\bar{\lambda} - \lambda)) - \nabla h(\lambda)]^T(\alpha(\bar{\lambda} - \lambda))dt
\]

\[
\geq h(\lambda) + \nabla h(\lambda)^T(\alpha(\bar{\lambda} - \lambda)) - \int_0^1 \|\nabla h(\lambda + t\alpha(\bar{\lambda} - \lambda)) - \nabla h(\lambda)\|_*(\alpha)\|\bar{\lambda} - \lambda\|dt
\]

\[
\geq h(\lambda) + \nabla h(\lambda)^T(\alpha(\bar{\lambda} - \lambda)) - \int_0^1 L_{h,Q}(t\alpha)(\bar{\lambda} - \lambda)\|(\alpha)\|\bar{\lambda} - \lambda\|dt
\]

\[
= h(\lambda) + \nabla h(\lambda)^T(\alpha(\bar{\lambda} - \lambda)) - \frac{\alpha^2}{2} L_{h,Q}(\bar{\lambda} - \lambda)\|^2
\]

\[
\geq h(\lambda) + \nabla h(\lambda)^T(\alpha(\bar{\lambda} - \lambda)) - \frac{\alpha^2}{2} L_{h,Q}(\text{Diam}_Q)^2 ,
\]

whereby it follows that \( C_{h,Q} \leq L_{h,Q}(\text{Diam}_Q)^2 \). \( \square \)
Appendix B

Supplement to Chapter 4

B.1 Remainder of the Proof of Theorem 4.1

Recall that it remains to prove the following inequality:

\[
\frac{1}{f(x_{k+1}) - B_{k+1}} \geq \frac{1}{f(x_k) - B_k} + \frac{1}{2LD^2} \quad \text{for all } k \geq 0. \tag{B.1}
\]

Let us fix some simplifying notation. Let

\[ r_k := f(x_k) - B_k \geq 0 \]

and note that \( B_k \geq B^w_k = f(x_k) - G_k \), so that \( G_k \geq r_k \geq 0 \) for \( k \geq 0 \). Also define \( C_k := \bar{L} \|\tilde{x}_k - x_k\|^2 \), whereby \( C_k \leq \bar{L}D^2 \) and \( \bar{\alpha}_k = \min \left\{ \frac{G_k}{C_k}, 1 \right\} \) for \( k \geq 0 \). With this notation \((B.1)\) can be written as \( 1/r_{k+1} \geq 1/r_k + 1/(2LD^2) \). Substituting \( x = x_k \) and \( y = x_{k+1} = x_k + \bar{\alpha}_k(\tilde{x}_k - x_k) \) in \((4.8)\) and using \( \bar{L} \geq L \) yields:

\[
f(x_{k+1}) \leq f(x_k) + \bar{\alpha}_k \nabla f(x_k)^T(\tilde{x}_k - x_k) + \frac{\bar{L}}{2} \bar{\alpha}_k^2 \|\tilde{x}_k - x_k\|^2 = f(x_k) - \bar{\alpha}_k G_k + \frac{1}{2} \bar{\alpha}_k^2 C_k. \tag{B.2}
\]

Note that if we instead used an exact line-search to determine \( x_{k+1} \), then \((B.2)\) also holds since in that case we have \( f(x_{k+1}) \leq f(x_k + \bar{\alpha}_k(\tilde{x}_k - x_k)) \). We now examine two cases depending on the relative magnitudes of \( G_k \) and \( C_k \). **Case 1:** \( G_k \leq C_k \). In this case \( \bar{\alpha}_k = G_k/C_k \), and substituting this value in the right side of \((B.2)\) yields \( f(x_{k+1}) \leq f(x_k) - \frac{(G_k)^2}{2C_k} \), which shows
that \( f(x_{k+1}) \leq f(x_k) \) as well as \( r_{k+1} \leq r_k \), and also yields:

\[
r_{k+1} = f(x_{k+1}) - B_{k+1} \leq f(x_{k+1}) - B_k \leq f(x_k) - \frac{(G_k)^2}{2C_k} - B_k = r_k - \frac{(G_k)^2}{2C_k} \leq r_k - \frac{r_k r_{k+1}}{2C_k},
\]

where the last inequality uses \( r_{k+1} \leq r_k \leq G_k \). Dividing the above inequality by \( r_{k+1} r_k \) and rearranging yields

\[
\frac{1}{r_{k+1}} \geq \frac{1}{r_k} + \frac{1}{2C_k} \geq \frac{1}{r_k} + \frac{1}{2LD^2},
\]

where the second inequality above uses \( C_k \leq LD^2 \). This shows that \( \text{(B.1)} \) holds in this case.

**Case 2:** \( G_k > C_k \). In this case \( \bar{\alpha}_k = 1 \). Substituting \( x = x_k \) and \( y = x_{k+1} = x_k + \bar{\alpha}_k (\bar{x}_k - x_k) = \bar{x}_k \) in \( \text{(B.2)} \) yields

\[
f(x_{k+1}) \leq f(x_k) - G_k + \frac{1}{2} C_k < f(x_k) - C_k + \frac{1}{2} C_k = f(x_k) - \frac{1}{2} C_k,
\]

which shows that \( f(x_{k+1}) < f(x_k) \) as well as \( r_{k+1} < r_k \), and also yields:

\[
r_{k+1} = f(x_{k+1}) - B_{k+1} \leq f(x_{k+1}) - B_k \leq f(x_k) - G_k + \frac{1}{2} C_k - B_k = r_k - G_k + \frac{1}{2} C_k,
\]

from which we derive:

\[
0 \leq r_{k+1} \leq r_k - G_k + \frac{1}{2} C_k < r_k - G_k + \frac{1}{2} G_k = r_k - \frac{1}{2} G_k,
\]

where the last inequality above uses \( G_k > C_k \). We now consider two sub-cases, one for \( k = 0 \) and another sub-case for \( k \geq 1 \). Let us first consider when \( k = 0 \). Then

\[
G_0 r_0 + G_0 C_0 = G_0 r_0 + \frac{1}{2} G_0 C_0 + \frac{1}{2} G_0 C_0 \geq (r_0)^2 + \frac{1}{2} (C_0)^2 + \frac{1}{2} r_0 C_0,
\]

since \( G_0 \geq C_0 \) and \( G_0 \geq r_0 \), and now add \( r_0 C_0 \) to both sides and rearrange to yield:

\[
r_0 C_0 \geq r_0 C_0 + (r_0)^2 - G_0 r_0 - G_0 C_0 + \frac{1}{2} (C_0)^2 + \frac{1}{2} r_0 C_0 = (r_0 - G_0 + \frac{1}{2} C_0)(r_0 + C_0) \geq r_1 (r_0 + C_0),
\]
where the second inequality uses (B.3) with $k = 0$. Therefore:

\[
\frac{1}{r_1} \geq \frac{r_0 + C_0}{r_0 C_0} = \frac{1}{r_0} + \frac{1}{C_0} \geq \frac{1}{r_0} + \frac{1}{LD^2},
\]

which proves (B.1) for this case for $k = 0$. Last of all, we consider when $k \geq 1$. Taking (B.4) and dividing by $r_k r_{k+1}$ and rearranging yields:

\[
\frac{1}{r_{k+1}} > \frac{1}{r_k} + \frac{G_k}{2r_k r_{k+1}} \geq \frac{1}{r_k} + \frac{1}{2r_{k+1}},
\]

where the second inequality follows since $G_k \geq r_k$. Now notice from (B.4) that $r_{k+1} \leq r_k - G_k + C_k/2 \leq C_k/2$ since $G_k \geq r_k$. Substituting this last inequality into the right-most term above yields:

\[
\frac{1}{r_{k+1}} \geq \frac{1}{r_k} + \frac{1}{C_k} \geq \frac{1}{r_k} + \frac{1}{2C_k} \geq \frac{1}{r_k} + \frac{1}{2LD^2},
\]

which shows (B.1) for this case for $k \geq 1$, and completes the proof. ∎
Appendix C

Supplement to Chapter 5

C.1 Additional Details for Section 5.2

C.1.1 Properties of Convex Quadratic Functions

Consider the following quadratic optimization problem (QP) defined as:

$$h^* := \min_{x \in \mathbb{R}^n} h(x) := \frac{1}{2} x^T Q x + q^T x + q^o,$$

where $Q$ is a symmetric positive semi-definite matrix, whereby $h(\cdot)$ is a convex function. We assume that $Q \neq 0$, and recall that $\lambda_{p\text{min}}(Q)$ denotes the smallest nonzero (and hence positive) eigenvalue of $Q$.

**Proposition C.1.** If $h^* > -\infty$, then for any given $x$, there exists an optimal solution $x^*$ of (QP) for which

$$\|x - x^*\|_2 \leq \sqrt{\frac{2(h(x) - h^*)}{\lambda_{p\text{min}}(Q)}}.$$

Also, it holds that

$$\|\nabla h(x)\|_2 \geq \sqrt{\frac{\lambda_{p\text{min}}(Q) \cdot (h(x) - h^*)}{2}}.$$
Proof: The result is simply manipulation of linear algebra. Let us assume without loss of
generality that \( q^o = 0 \). If \( h^* > -\infty \), then (QP) has an optimal solution \( x^* \), and the set of
optimal solutions are characterized by the gradient condition

\[
0 = \nabla h(x) = Qx + q .
\]

Now let us write the sparse eigendecomposition of \( Q \) as \( Q = PDP^T \) where \( D \) is a diagonal
matrix of non-zero eigenvalues of \( Q \) and the columns of \( P \) are orthonormal, namely \( P^TP = I \).
Because (QP) has an optimal solution, the system of equations \( Qx = -q \) has a solution, and
let \( \tilde{x} \) denote any such solution. Direct manipulation establishes:

\[
PP^Tq = -PP^TQ\tilde{x} = -PP^TPDP^T\tilde{x} = -PDP^T\tilde{x} = -Q\tilde{x} = q .
\]

Furthermore, let \( \hat{x} := -PD^{-1}P^Tq \). It is then straightforward to see that \( \hat{x} \) is an optimal
solution of (QP) since in particular:

\[
Q\hat{x} = -PDP^TPD^{-1}P^Tq = -PP^Tq = -q ,
\]

and hence

\[
h^* = \frac{1}{2}\hat{x}^TQ\hat{x} + q^T\hat{x} = -\frac{1}{2}\hat{x}^TQ\hat{x} = -\frac{1}{2}q^TPD^{-1}P^TPDP^TPD^{-1}P^Tq = -\frac{1}{2}q^TPD^{-1}P^Tq .
\]

Now let \( x \) be given, and define \( x^* := [I - PP^T]x - PD^{-1}P^Tq \). Then just as above it is
straightforward to establish that \( Qx^* = -q \) whereby \( x^* \) is an optimal solution. Furthermore,
it holds that:

\[
\|x - x^*\|_2^2 = (q^T P D^{-1} + x^T P) P^T P (D^{-1} P^T q + P^T x)
\]

\[
= (q^T P D^{-\frac{1}{2}} + x^T P D^{\frac{1}{2}}) D^{-1} (D^{-\frac{1}{2}} P^T q + D^{\frac{1}{2}} P^T x)
\]

\[
\leq \frac{1}{\lambda_{\text{pmin}}(Q)} (q^T P D^{-\frac{1}{2}} + x^T P D^{\frac{1}{2}}) (D^{-\frac{1}{2}} P^T q + D^{\frac{1}{2}} P^T x)
\]

\[
= \frac{1}{\lambda_{\text{pmin}}(Q)} (q^T P D^{-1} P^T q + x^T P D^T x + 2 x^T P P^T q)
\]

\[
= \frac{1}{\lambda_{\text{pmin}}(Q)} (-2h^* + x^T Q x + 2x^T q)
\]

\[
= \frac{2}{\lambda_{\text{pmin}}(Q)} (h(x) - h^*) ,
\]

and taking square roots establishes the first inequality of the proposition.

Using the gradient inequality for convex functions, it holds that:

\[
h^* = h(x^*) \geq h(x) + \nabla h(x)^T (x^* - x)
\]

\[
\geq h(x) - \|\nabla h(x)\|_2 \|x^* - x\|_2
\]

\[
\geq h(x) - \|\nabla h(x)\|_2 \sqrt{\frac{2(h(x) - h^*)}{\lambda_{\text{pmin}}(Q)}} ,
\]

and rearranging the above proves the second inequality of the proposition.

\[\square\]

C.1.2 Proof of Theorem 5.1

We first prove part (i). Since the predictors are all standardized to have unit \(\ell_2\) norm, it follows that the coefficient index \(j_k\) and corresponding step-size \(\tilde{u}_{j_k}\) selected in Step (2.) of
LS-Boost(\(\varepsilon\)) satisfy:

\[
    j_k \in \arg \max_{j \in \{1, \ldots, p\}} |(\hat{r}^k)^T X_j| \quad \text{and} \quad \tilde{u}_{jk} = (\hat{r}^k)^T X_j.
\]  

(C.1)

Combining (5.6) and (C.1), we see that

\[
    |\tilde{u}_{jk}| = |(\hat{r}^k)^T X_j| = n \cdot \|\nabla L_n(\hat{\beta}^k)\|_\infty.
\]  

(C.2)

Using the formula for \(\tilde{u}_{jk}\) in (C.1), we have the following convenient way to express the change in residuals at each iteration of LS-Boost(\(\varepsilon\)):

\[
    \hat{r}_{k+1} = \hat{r}_k - \varepsilon ( (\hat{r}^k)^T X_{jk} ) X_{jk}.
\]  

(C.3)

Utilizing (C.3), we have:

\[
    L_n(\hat{\beta}^{k+1}) = \frac{1}{2n} \|\hat{r}^{k+1}\|_2^2
\]

\[
    = \frac{1}{2n} \|\hat{r}_k - \varepsilon ( (\hat{r}^k)^T X_{jk} ) X_{jk}\|_2^2
\]

\[
    = \frac{1}{2n} \|\hat{r}_k\|_2^2 - \frac{1}{n} \varepsilon ( (\hat{r}^k)^T X_{jk} )^2 + \frac{1}{2n} \varepsilon^2 ( (\hat{r}^k)^T X_{jk} )^2
\]

\[
    = L_n(\hat{\beta}^k) - \frac{1}{2n} \varepsilon (2 - \varepsilon) ((\hat{r}^k)^T X_{jk})^2
\]

(C.4)

\[
    = L_n(\hat{\beta}^k) - \frac{1}{2n} \varepsilon (2 - \varepsilon) n^2 \|\nabla L_n(\hat{\beta}^k)\|_\infty^2,
\]

(\text{where the last equality above uses (C.2)}, which yields:

\[
    L_n(\hat{\beta}^{k+1}) - L_n^* = L_n(\hat{\beta}^k) - L_n^* - \frac{n}{2} \varepsilon (2 - \varepsilon) \|\nabla L_n(\hat{\beta}^k)\|_\infty^2.
\]  

(C.5)
We next seek to bound the right-most term above. We will do this by invoking Proposition C.1, which presents two important properties of convex quadratic functions. Because \( L_n(\cdot) \) is a convex quadratic function of the same format as Proposition C.1 with \( h(\cdot) \leftarrow L_n(\cdot) \), \( Q \leftarrow \frac{1}{n} X^T X \), and \( h^* \leftarrow L^*_n \), it follows from the second property of Proposition C.1 that

\[
\| \nabla L_n(\beta) \|_2 \geq \sqrt{\frac{\lambda_{\text{pmin}}(\frac{1}{n} X^T X)(L_n(\beta) - L^*_n)}{2n}} = \sqrt{\frac{\lambda_{\text{pmin}}(X^T X)(L_n(\beta) - L^*_n)}{2n}}.
\]

Therefore

\[
\| \nabla L_n(\beta) \|_\infty \geq \frac{1}{p} \| \nabla L_n(\beta) \|_2 \geq \frac{\lambda_{\text{pmin}}(X^T X)(L_n(\beta) - L^*_n)}{2np}.
\]

Substituting this inequality into (C.5) yields after rearranging:

\[
L_n(\hat{\beta}^{k+1}) - L^*_n \leq (L_n(\hat{\beta}^k) - L^*_n) \left( 1 - \frac{\varepsilon(2 - \varepsilon)\lambda_{\text{pmin}}(X^T X)}{4p} \right) = (L_n(\hat{\beta}^k) - L^*_n) \cdot \gamma. \tag{C.6}
\]

Now note that \( L_n(\hat{\beta}^0) = L_n(0) = \frac{1}{2n} \| y \|_2^2 \) and

\[
L_n(\hat{\beta}^0) - L^*_n = \frac{1}{2n} \| y \|_2^2 - \frac{1}{2n} \| y - X\hat{\beta}_{\text{LS}} \|_2^2 = \frac{1}{2n} \| y \|_2^2 - \frac{1}{2n}(\| y \|_2^2 - 2y^T X\hat{\beta}_{\text{LS}} + \| X\hat{\beta}_{\text{LS}} \|_2^2) = \frac{1}{2n} \| X\hat{\beta}_{\text{LS}} \|_2^2,
\]

where the last equality uses the normal equations (5.5). Then (i) follows by using elementary induction and combining the above with (C.6):

\[
L_n(\hat{\beta}^k) - L^*_n \leq (L_n(\hat{\beta}^0) - L^*_n) \cdot \gamma^k = \frac{1}{2n} \| X\hat{\beta}_{\text{LS}} \|_2^2 \cdot \gamma^k.
\]

To prove (ii), we invoke the first inequality of Proposition C.1, which in this context states that

\[
\| \hat{\beta}^k - \hat{\beta}_{\text{LS}} \|_2 \leq \sqrt{\frac{2(L_n(\hat{\beta}^k) - L^*_n)}{\lambda_{\text{pmin}}(\frac{1}{n} X^T X)}} = \sqrt{\frac{2n(L_n(\hat{\beta}^k) - L^*_n)}{\lambda_{\text{pmin}}(X^T X)}}.
\]

Part (ii) then follows by substituting the bound on \( (L_n(\hat{\beta}^k) - L^*_n) \) from (i) and simplifying terms. Similarly, the proof of (iii) follows from the observation that \( \| X\hat{\beta}^k - X\hat{\beta}_{\text{LS}} \|_2 = \).
\[
\sqrt{2n(L_n(\hat{\beta}^k) - L_n^*)} \text{ and then substituting the bound on } (L_n(\hat{\beta}^k) - L_n^*) \text{ from (i) and simplifying terms.}
\]

To prove (iv), define the point \( \tilde{\beta}^k := \hat{\beta}^k + \tilde{u}_{jk} e_{jk} \). Then using similar arithmetic as in \((C.4)\) one obtains:

\[
L_n^* \leq L_n(\tilde{\beta}^k) = L_n(\hat{\beta}^k) - \frac{1}{2n} \tilde{u}_{jk}^2,
\]

where we recall that \( \tilde{u}_{jk} = (\tilde{r}^k)^T X_{jk} \). This inequality then rearranges to

\[
|\tilde{u}_{jk}| \leq \sqrt{2n(L_n(\hat{\beta}^k) - L_n^*)} \leq \|X_{LS}\|_2 \cdot \gamma^k / 2^k \text{,} \tag{C.7}
\]

where the second inequality follows by substituting the bound on \( (L_n(\hat{\beta}^i) - L_n^*) \) from (i). Recalling (5.6) and (C.2), the above is exactly part (iv).

Part (v) presents two distinct bounds on \( \|\hat{\beta}^k\|_1 \), which we prove independently. To prove the first bound, let \( \hat{\beta}_{LS} \) be any least-squares solution, which therefore satisfies (5.5). It is then elementary to derive using similar manipulation as in \((C.4)\) that for all \( i \) the following holds:

\[
\|X(\hat{\beta}^{i+1} - \hat{\beta}_{LS})\|_2^2 = \|X(\hat{\beta}^i - \hat{\beta}_{LS})\|_2^2 - (2\varepsilon - \varepsilon^2) \tilde{u}_{ji}^2 \tag{C.8}
\]

which implies that

\[
(2\varepsilon - \varepsilon^2) \sum_{i=0}^{k-1} \tilde{u}_{ji}^2 = \|X(\hat{\beta}^0 - \hat{\beta}_{LS})\|_2^2 - \|X(\hat{\beta}^k - \hat{\beta}_{LS})\|_2^2 = \|X\hat{\beta}_{LS}\|_2^2 - \|X(\hat{\beta}^k - \hat{\beta}_{LS})\|_2^2 \text{.} \tag{C.9}
\]

Then note that

\[
\|\hat{\beta}^k\|_1 \leq \|(\varepsilon \tilde{u}_{jo}, \ldots, \varepsilon \tilde{u}_{jk-1})\|_1 \leq \sqrt{k}\varepsilon\|((\tilde{u}_{jo}, \ldots, \tilde{u}_{jk-1})\|_2 = \sqrt{k} \varepsilon \sqrt{\frac{\varepsilon}{2-\varepsilon}} \sqrt{\|X\hat{\beta}_{LS}\|_2^2 - \|X\hat{\beta}_{LS} - X\hat{\beta}^k\|_2^2},
\]

where the last equality is from \((C.9)\).

To prove the second bound in (v), noting that \( \hat{\beta}^k = \sum_{i=0}^{k-1} \varepsilon \tilde{u}_{ji} e_{ji} \), we bound \( \|\hat{\beta}^k\|_1 \) as
follows:

\[ \|\hat{\beta}^k\|_1 \leq \varepsilon \sum_{i=0}^{k-1} |\bar{u}_i| \leq \varepsilon \|X\hat{\beta}_{LS}\|_2 \sum_{i=0}^{k-1} \gamma_i^{i/2} \]
\[ = \frac{\varepsilon \|X\hat{\beta}_{LS}\|_2}{1 - \sqrt{\gamma}} (1 - \gamma^{k/2}) , \]

where the second inequality uses (C.7) for each \(i \in \{0, \ldots, k - 1\}\) and the final equality is a geometric series, which completes the proof of (v). Part (vi) is simply the property of \(LS-\text{Boost}(\varepsilon)\) that derives from the fact that \(\hat{\beta}_0 := 0\) and at every iteration at most one coordinate of \(\beta\) changes status from a zero to a non-zero value. \(\square\)

### C.2 Additional Details for Section 5.3

#### C.2.1 An Elementary Sequence Process Result, and a Proof of Proposition 5.1

Consider the following elementary sequence process: \(x^0 \in \mathbb{R}^n\) is given, and \(x^{i+1} \leftarrow x^i - \alpha_i g^i\) for all \(i \geq 0\), where \(g^i \in \mathbb{R}^n\) and \(\alpha_i\) is a nonnegative scalar, for all \(i\). For this process there are no assumptions on how the vectors \(g^i\) might be generated.

**Proposition C.2.** For the elementary sequence process described above, suppose that the \(\{g^i\}\) are uniformly bounded, namely \(\|g^i\|_2 \leq G\) for all \(i \geq 0\). Then for all \(k \geq 0\) and for any \(x \in \mathbb{R}^n\) it holds that:

\[ \frac{1}{\sum_{i=0}^{k} \alpha_i} \sum_{i=0}^{k} \alpha_i (g^i)^T (x^i - x) \leq \frac{\|x^0 - x\|_2^2 + G^2 \sum_{i=0}^{k} \alpha_i^2}{2 \sum_{i=0}^{k} \alpha_i} . \]  
\[ \text{(C.10)} \]

Indeed, in the case when \(\alpha_i = \varepsilon\) for all \(i\), it holds that:

\[ \frac{1}{k+1} \sum_{i=0}^{k} (g^i)^T (x^i - x) \leq \frac{\|x^0 - x\|_2^2}{2(k+1)\varepsilon} + \frac{G^2\varepsilon}{2} . \]  
\[ \text{(C.11)} \]
Proof. Elementary arithmetic yields the following:

\[
\|x^{i+1} - x\|^2_2 = \|x^i - \alpha_i g^i - x\|^2_2 \\
= \|x^i - x\|^2_2 + \alpha_i^2 \|g^i\|^2_2 + 2\alpha_i (g^i)^T (x - x^i) \\
\leq \|x^i - x\|^2_2 + G^2 \alpha_i^2 + 2\alpha_i (g^i)^T (x - x^i).
\]

Rearranging and summing these inequalities for \(i = 0, \ldots, k\) then yields:

\[
2 \sum_{i=0}^k \alpha_i (g^i)^T (x^i - x) \leq G^2 \sum_{i=0}^k \alpha_i^2 + \|x^0 - x\|^2_2 - \|x^{k+1} - x\|^2_2 \leq G^2 \sum_{i=0}^k \alpha_i^2 + \|x^0 - x\|^2_2,
\]

which then rearranges to yield (C.10). (C.11) follows from (C.10) by direct substitution. \(\square\)

Proof of Proposition 5.1: Consider the subgradient descent method (5.14) with arbitrary step-sizes \(\alpha_i\) for all \(i\). We will prove the following inequality:

\[
\min_{i \in \{0, \ldots, k\}} f(x^i) \leq f^* + \frac{\|x^0 - x^*\|^2_2 + G^2 \sum_{i=0}^k \alpha_i^2}{2 \sum_{i=0}^k \alpha_i}, \tag{C.12}
\]

from which the proof of Proposition 5.1 follows by substituting \(\alpha_i = \alpha\) for all \(i\) and simplifying terms. Let us now prove (C.12). The subgradient descent method (5.14) is applied to instances of problem (3.1) where \(f(\cdot)\) is convex, and where \(g^i\) is subgradient of \(f(\cdot)\) at \(x^i\), for all \(i\). If \(x^*\) is an optimal solution of (3.1), it therefore holds from the subgradient inequality that

\[
f^* = f(x^*) \geq f(x^i) + (g^i)^T (x - x^i).
\]
Substituting this inequality in (C.10) for the value of $x = x^*$ yields:

$$\frac{\|x_0 - x^*\|_2^2}{2} + G^2 \sum_{i=0}^{k} \alpha_i^2 \geq \frac{1}{\sum_{i=0}^{k} \alpha_i} \sum_{i=0}^{k} \alpha_i (g^i)^T (x^i - x^*) \geq \frac{1}{\sum_{i=0}^{k} \alpha_i} \sum_{i=0}^{k} \alpha_i (f(x^i) - f^*) \geq \min_{i \in \{0, \ldots, k\}} f(x^i).$$

\[\Box\]

### C.2.2 Proof of Theorem 5.2

We first prove part (i). Note that item (i) of Proposition 5.2 shows that $\text{FS}_\varepsilon$ is a specific instance of subgradient descent to solve problem (5.16), using the constant step-size $\varepsilon$. Therefore we can apply the computational guarantees associated with the subgradient descent method, particularly Proposition 5.1, to the $\text{FS}_\varepsilon$ algorithm. Examining Proposition 5.1 we need to work out the corresponding values of $f^*$, $\|x_0 - x^*\|_2$, $\alpha$, and $G$ in the context of $\text{FS}_\varepsilon$ for solving the CM problem (5.16). Note that $f^* = 0$ for problem (5.16). We bound the distance from the initial residuals to the optimal least-squares residuals as follows:

$$\|\hat{r}_0 - r^*\|_2 = \|\hat{r}_L S\|_2 = \|y - (y - X\hat{\beta}_{LS})\|_2 = \|X\hat{\beta}_{LS}\|_2.$$

From Proposition 5.2 part (i) we have $\alpha = \varepsilon$. Last of all, we need to determine an upper bound $G$ on the norms of subgradients. We have:

$$\|g^k\|_2 = \|\text{sgn}((\hat{r}^k)^T X_{j_k})X_{j_k}\|_2 = \|X_{j_k}\|_2 = 1,$$

since the covariates have been standardized, so we can set $G = 1$. Now suppose algorithm $\text{FS}_\varepsilon$ is run for $k$ iterations. Proposition 5.1 then implies that:

$$\min_{i \in \{0, \ldots, k\}} \|X^T \hat{r}^i\|_\infty = \min_{i \in \{0, \ldots, k\}} f(\hat{r}^i) \leq f^* + \frac{\|\hat{r}_0 - r^*\|_2^2}{2\alpha(k+1)} + \frac{\alpha G^2}{2} = \frac{\|X\hat{\beta}_{LS}\|_2^2}{2\varepsilon(k+1)} + \frac{\varepsilon}{2}. \quad (C.13)$$

201
The above inequality provides a bound on the best (among the first $k$ residual iterates) empirical correlation between between the residuals $\hat{r}^i$ and each predictor variable, where the bound depends explicitly on the learning rate $\varepsilon$ and the number of iterations $k$. Furthermore, invoking \[5.6\], the above inequality implies the following upper bound on the norm of the gradient of the least squares loss $L_n(\cdot)$ for the model iterates $\{\hat{\beta}^i\}$ generated by $\text{FS}_\varepsilon$:

\[
\min_{i \in \{0, \ldots, k\}} \|\nabla L_n(\hat{\beta}^i)\|_\infty \leq \frac{\|X\hat{\beta}_{LS}\|_2^2}{2n\varepsilon(k + 1)} + \frac{\varepsilon}{2n}. \tag{C.14}
\]

Let $i$ be the index where the minimum is attained on the left side of the above inequality. In a similar vein as in the analysis in Section 5.2, we now use Proposition C.1 which presents two important properties of convex quadratic functions. Because $L_n(\cdot)$ is a convex quadratic function of the same format as Proposition C.1 with $h(\cdot) \leftarrow L_n(\cdot)$, $Q \leftarrow \frac{1}{n}X^TX$, and $h^* \leftarrow L_n^*$, it follows from the second property of Proposition C.1 that

\[
\|\nabla L_n(\hat{\beta}^i)\|_2 \geq \sqrt{\frac{\lambda_{p\min}(\frac{1}{n}X^TX)(L_n(\hat{\beta}^i) - L_n^*)}{2}} = \sqrt{\frac{\lambda_{p\min}(X^TX)(L_n(\hat{\beta}^i) - L_n^*)}{2n}}.
\]

where recall that $\lambda_{p\min}(X^TX)$ denotes the smallest non-zero (hence positive) eigenvalue of $X^TX$. Therefore

\[
\|\nabla L_n(\hat{\beta}^i)\|_\infty \geq \frac{1}{p}\|\nabla L_n(\hat{\beta}^i)\|_2 \geq \frac{\lambda_{p\min}(X^TX)(L_n(\hat{\beta}^i) - L_n^*)}{2np}.
\]

Substituting this inequality into (C.14) for the index $i$ where the minimum is attained yields after rearranging:

\[
L_n(\hat{\beta}^i) - L_n^* \leq \frac{p}{2n\lambda_{p\min}(X^TX)} \left[\frac{\|X\hat{\beta}_{LS}\|_2^2}{\varepsilon(k + 1)} + \varepsilon\right]^2, \tag{C.15}
\]
which proves part (i). The proof of part (ii) follows by noting from the first inequality of Proposition \[C.1\] that there exists a least-squares solution \(\hat{\beta}^*\) for which:

\[
\|\hat{\beta}^* - \hat{\beta}_i\|_2 \leq \sqrt{2(\delta_n(\hat{\beta}_i) - L_n^*) \lambda_{\text{pmin}}(\frac{1}{n}X^TX) \frac{1}{\varepsilon(k + 1)} + \varepsilon},
\]

where the second inequality in the above chain follows using (C.15). The proof of part (iii) follows by first observing that \(\|X(\hat{\beta}_i - \hat{\beta}_\text{LS})\|_2 = \sqrt{2n(\delta_n(\hat{\beta}_i) - L_n^*)}\) and then substituting the bound on \((\delta_n(\hat{\beta}_i) - L_n^*)\) from part (i) and simplifying terms. Part (iv) is a restatement of inequality (C.13). Finally, parts (v) and (vi) are simple and well-known structural properties of FS\(\varepsilon\) that are re-stated here for completeness.

\[
C.3 \quad \text{Additional Details for Section 5.4}
\]

\textbf{C.3.1 \quad Duality Between Regularized Correlation Minimization and the LASSO}

In this section, we precisely state the duality relationship between the RCM problem (5.19) and the LASSO. We first prove the following property of the least squares loss function that will be useful in our analysis.

\textbf{Proposition C.3.} \textit{The least squares loss function} \(L_n(\cdot)\) \textit{has the following max representation:}

\[
L_n(\beta) = \max_{\tilde{\beta} \in P_{\text{res}}} \{-\tilde{\beta}^T(\frac{1}{n}X)\beta - \frac{1}{2n}\|\tilde{\beta} - y\|_2^2 + \frac{1}{2n}\|y\|_2^2\}, \tag{C.16}
\]

where \(P_{\text{res}} := \{r \in \mathbb{R}^n : r = y - X\beta\ \text{for some} \ \beta \in \mathbb{R}^p\}\). Moreover, the unique optimal solution (as a function of \(\beta\)) to the subproblem in (C.16) is \(\tilde{\beta} := y - X\beta\).

\textit{Proof.} For any \(\beta \in \mathbb{R}^p\), it is easy to verify through optimality conditions (setting the gradient
with respect to \( \tilde{r} \) equal to 0) that \( \bar{r} \) solves the subproblem in (C.16), i.e.,

\[
\bar{r} = \arg \max_{\tilde{r} \in P_{\text{res}}} \left\{ -\tilde{r}^T \left( \frac{1}{n} X \right) \beta - \frac{1}{2n} \| \tilde{r} - y \|_2^2 + \frac{1}{2n} \| y \|_2^2 \right\} .
\]

Thus, we have

\[
\max_{\tilde{r} \in P_{\text{res}}} \left\{ -\tilde{r}^T \left( \frac{1}{n} X \right) \beta - \frac{1}{2n} \| \tilde{r} - y \|_2^2 + \frac{1}{2n} \| y \|_2^2 \right\} = \frac{1}{n^2} \left( \frac{1}{2} \| y \|_2^2 - y^T X \beta + \frac{1}{2} \| X \beta \|_2^2 \right) = \frac{1}{2n} \| y - X \beta \|_2^2 .
\]

The following result demonstrates that RCM \((5.19)\) has a direct interpretation as a (scaled) dual of the LASSO problem \((5.1)\). Moreover, in part (iii) of the below Proposition, we give a bound on the optimality gap for the LASSO problem in terms of a quantity that is closely related to the objective function of RCM.

**Proposition C.4. (Duality Equivalence of LASSO and RCM\(_{\delta}\), and Optimality Bounds)** The LASSO problem \((5.1)\) and the regularized correlation minimization problem RCM\(_{\delta}\) \((5.19)\) are dual optimization problems modulo the scaling factor \(n^2\). In particular:

(i) **(Weak Duality)** If \(\beta\) is feasible for the LASSO problem \((5.1)\), and if \(\tilde{r}\) is feasible for the regularized correlation minimization problem RCM\(_{\delta}\) \((5.19)\), then

\[
L_n(\beta) + \frac{\delta}{n} f_\delta(\tilde{r}) \geq \frac{1}{2n^2} \| y \|_2^2 .
\]

(ii) **(Strong Duality)** It holds that:

\[
L_{n,\delta}^* + \frac{\delta}{n} f_\delta^* = \frac{1}{2n^2} \| y \|_2^2 .
\]

Moreover, for any given parameter value \(\delta \geq 0\), there is a unique vector of residuals \(\hat{r}_\delta^*\) associated with every LASSO solution \(\hat{\beta}_\delta^*\), i.e., \(\hat{r}_\delta^* = y - X \hat{\beta}_\delta^*\), and \(\hat{r}_\delta^*\) is the unique
optimal solution to the RCMₜ problem \((5.19)\).

(iii) (Optimality Condition for LASSO) If \(\beta\) is feasible for the LASSO problem \((5.1)\) and \(r = y - X\beta\), then

\[
\omega_\delta(\beta) := \|X^T r\|_\infty - \frac{r^T X \beta}{\delta} \geq 0 ,
\]

(C.17)

and

\[
L_n(\beta) - L^*_{n,\delta} \leq \frac{\delta}{n} \cdot \omega_\delta(\beta) .
\]

Hence, if \(\omega_\delta(\beta) = 0\), then \(\beta\) is an optimal solution of the LASSO problem \((5.1)\). \(\Box\)

**Proof.** Let us first construct the problem RCMₜ using basic constructs of minmax duality. As demonstrated in Proposition \(C.3\), the least-squares loss function \(L_n(\cdot)\) has the following max representation:

\[
L_n(\beta) = \max_{\tilde{r} \in P_{\text{res}}} \left\{ -\tilde{r}^T \left( \frac{1}{n} X \right) \beta - \frac{1}{2n} \|\tilde{r} - y\|_2^2 + \frac{1}{2n} \|y\|_2^2 \right\}.
\]

Therefore the LASSO problem \((5.1)\) can be written as

\[
\min_{\beta \in B_\delta} \max_{\tilde{r} \in P_{\text{res}}} \left\{ -\tilde{r}^T \left( \frac{1}{n} X \right) \beta - \frac{1}{2n} \|\tilde{r} - y\|_2^2 + \frac{1}{2n} \|y\|_2^2 \right\}
\]

where \(B_\delta := \{ \beta \in \mathbb{R}^p : \|\beta\|_1 \leq \delta \}\). We construct a dual of the above problem by interchanging the min and max operators above, yielding the following dual optimization problem:

\[
\max_{\tilde{r} \in P_{\text{res}}} \min_{\beta \in B_\delta} \left\{ -\tilde{r}^T \left( \frac{1}{n} X \right) \beta - \frac{1}{2n} \|\tilde{r} - y\|_2^2 + \frac{1}{2n} \|y\|_2^2 \right\} .
\]

After negating, and dropping the constant term \(\frac{1}{2n} \|y\|_2^2\), the above dual problem is equivalent to:

\[
\min_{\tilde{r} \in P_{\text{res}}} \max_{\beta \in B_\delta} \left\{ \tilde{r}^T \left( \frac{1}{n} X \right) \beta \right\} + \frac{1}{2n} \|\tilde{r} - y\|_2^2 .
\]

(C.18)
Now notice that
\[
\max_{\beta \in B_\delta} \left\{ \tilde{r}^T \left( \frac{1}{n} X \right) \beta \right\} = \frac{\delta}{n} \left( \max_{j \in \{1, \ldots, p\}} |\tilde{r}^T X_j| \right) = \frac{\delta}{n} \|X^T \tilde{r}\|_\infty , \tag{C.19}
\]
from which it follows after scaling by \( \frac{n}{\delta} \) that (C.18) is equivalent to (5.19).

Let us now prove item (i). Let \( \beta \) be feasible for the LASSO problem (5.1) and \( \tilde{r} \) be feasible for the regularized correlation minimization problem \( \text{RCM}_\delta \) (5.19), and let \( r = y - X \beta \) and let \( \tilde{\beta} \) be such that \( \tilde{r} = y - X \tilde{\beta} \). Then direct arithmetic manipulation yields the following equality:

\[
L_n(\beta) + \frac{\delta}{n} f_\delta(\tilde{r}) = \frac{1}{2n} \|y\|_2^2 + \frac{1}{2n} \|r - \tilde{r}\|_2^2 + \frac{\delta}{n} \left( \|X^T \tilde{r}\|_\infty - \frac{\tilde{r}^T X \beta}{\delta} \right) , \tag{C.20}
\]
from which the result follows since \( \|r - \tilde{r}\|_2^2 \geq 0 \) and \( \tilde{r}^T X \beta \leq \|X^T \tilde{r}\|_\infty \|\beta\|_1 \leq \delta \|X^T \tilde{r}\|_\infty \), which implies that the last term above is also nonnegative.

To prove item (ii), notice that both the LASSO and \( \text{RCM}_\delta \) can be re-cast as optimization problems with a convex quadratic objective function and with linear inequality constraints. That being the case, the classical strong duality results for linearly-constrained convex quadratic optimization apply, see [2] for example.

We now prove (iii). Since \( \beta \) is feasible for the LASSO problem, it follows from the Holder inequality that \( r^T X \beta \leq \|X^T r\|_\infty \|\beta\|_1 \leq \delta \|X^T r\|_\infty \), from which it then follows that \( \omega_\delta(\beta) \geq 0 \). Invoking (C.20) with \( \tilde{r} \leftarrow r = y - X \beta \) yields:

\[
L_n(\beta) + \frac{\delta}{n} f_\delta(r) = \frac{1}{2n} \|y\|_2^2 + \frac{\delta}{n} \cdot \omega_\delta(\beta) .
\]

Combining the above with strong duality (ii) yields:

\[
L_n(\beta) + \frac{\delta}{n} f_\delta(r) = L^*_n,\delta + \frac{\delta}{n} f^*_\delta + \frac{\delta}{n} \cdot \omega_\delta(\beta) .
\]
After rearranging we have:

\[
L_n(\beta) - L^*_{n,\delta} \leq \frac{\delta}{n} f^*_\delta - \frac{\delta}{n} f_\delta(r) + \frac{\delta}{n} \cdot \omega_\delta(\beta) \leq \frac{\delta}{n} \cdot \omega_\delta(\beta),
\]

where the last inequality follows since \( f^*_\delta \leq f_\delta(r) \).

---

### C.3.2 Proof of Proposition 5.3

Recall the update formula for the residuals in R-FS\(\varepsilon,\delta\):

\[
\hat{r}^{k+1} \leftarrow \hat{r}^k - \varepsilon \left[ \text{sgn}((\hat{r}^k)^T X_{jk}) X_{jk} + \frac{1}{\delta} (\hat{r}^k - y) \right]. \tag{C.21}
\]

We first show that \( g^k := \text{sgn}((\hat{r}^k)^T X_{jk}) X_{jk} + \frac{1}{\delta} (\hat{r}^k - y) \) is a subgradient of \( f_\delta(\cdot) \) at \( \hat{r}^k \). Recalling the proof of Proposition 5.2, we have that \( \text{sgn}((\hat{r}^k)^T X_{jk}) X_{jk} \) is a subgradient of \( f(r) := \|X^T r\|_\infty \) at \( \hat{r}^k \) since \( j_k \in \arg \max_{j \in \{1,...,p\}} |(\hat{r}^k)^T X_j| \). Therefore, since \( f_\delta(r) = f(r) + \frac{1}{2\delta} \|r - y\|_2^2 \), it follows from the additive property of subgradients (and gradients) that \( g^k = \text{sgn}((\hat{r}^k)^T X_{jk}) X_{jk} + \frac{1}{\delta} (\hat{r}^k - y) \) is a subgradient of \( f_\delta(r) \) at \( r = \hat{r}^k \). Therefore the update (C.21) is of the form \( \hat{r}^{k+1} = \hat{r}^k - \varepsilon g^k \) where \( g^k \in \partial f_\delta(\hat{r}^k) \). Finally note that \( \hat{r}^k - \varepsilon g^k = \hat{r}^{k+1} = y - X_{\beta^{k+1}} \in P_{\text{res}} \), hence \( \Pi_{P_{\text{res}}}(\hat{r}^k - \varepsilon g^k) = \hat{r}^k - \varepsilon g^k \), i.e., the projection step is superfluous here. Therefore \( \hat{r}^{k+1} = \Pi_{P_{\text{res}}}(\hat{r}^k - \varepsilon g^k) \), which shows that (C.21) is precisely the update for the subgradient descent method with step-size \( \alpha_k := \varepsilon \).

---

### C.3.3 Proof of Theorem 5.3

Let us first use induction to demonstrate that the following inequality holds:

\[
\|\hat{\beta}^k\|_1 \leq \varepsilon \sum_{j=0}^{k-1} \left( 1 - \frac{\varepsilon}{\delta} \right)^j \quad \text{for all } k \geq 0. \tag{C.22}
\]
Clearly, (C.22) holds for \( k = 0 \) since \( \hat{\beta}^0 = 0 \). Assuming that (C.22) holds for \( k \), then the update for \( \hat{\beta}^{k+1} \) in step (3.) of algorithm R-FS\( _{\varepsilon,\delta} \) can be written as 
\[
\hat{\beta}^{k+1} = (1 - \frac{\varepsilon}{\delta}) \hat{\beta}^k + \varepsilon \cdot \text{sgn}(\hat{r}^k)^T X_{jk} e_{jk},
\]
from which it holds that 
\[
\|\hat{\beta}^{k+1}\|_1 = \|(1 - \frac{\varepsilon}{\delta}) \hat{\beta}^k + \varepsilon \cdot \text{sgn}(\hat{r}^k)^T X_{jk} e_{jk}\|_1 
\leq (1 - \frac{\varepsilon}{\delta})\|\hat{\beta}^k\|_1 + \varepsilon \|e_{jk}\|_1 
\leq (1 - \frac{\varepsilon}{\delta})\varepsilon \sum_{j=0}^{k-1} (1 - \frac{\varepsilon}{\delta})^j + \varepsilon 
= \varepsilon \sum_{j=0}^{k} (1 - \frac{\varepsilon}{\delta})^j ,
\]
which completes the induction. Now note that (C.22) is a geometric series and we have:
\[
\|\hat{\beta}^k\|_1 \leq \varepsilon \sum_{j=0}^{k-1} (1 - \frac{\varepsilon}{\delta})^j = \delta \left[ 1 - (1 - \frac{\varepsilon}{\delta})^k \right] \leq \delta \quad \text{for all } k \geq 0 . \quad (C.23)
\]

Recall that we developed the algorithm R-FS\( _{\varepsilon,\delta} \) in such a way that it corresponds exactly to an instantiation of the subgradient descent method applied to the RCM problem (5.19). Indeed, the update rule for the residuals given in Step (3.) of R-FS\( _{\varepsilon,\delta} \) is: 
\[
\hat{r}^{k+1} \leftarrow \hat{r}^k - \varepsilon g^k 
\]
where 
\[
g^k = \left[ \text{sgn}(\hat{r}^k)^T X_{jk} X_{jk} + \frac{1}{\delta} (\hat{r}^k - y) \right].
\]
We therefore can apply Proposition (C.2) and more specifically the inequality (C.11). In order to do so we need to translate the terms of Proposition (C.2) to our setting: here the variables \( x \) are now the residuals \( r \), the iterates \( x^i \) are now the iterates \( \hat{r}^i \), etc. The step-sizes of algorithm R-FS\( _{\varepsilon,\delta} \) are fixed at \( \varepsilon \), so we have \( \alpha_i = \varepsilon \) for all \( i \geq 0 \). Setting the value of \( x \) in Proposition (C.2) to be least-squares residual...
value, namely \( x = \hat{r}_{LS} \), the left side of (C.11) is therefore:

\[
\frac{1}{k+1} \sum_{i=0}^{k} (g^i)^T (x^i - x) = \frac{1}{k+1} \sum_{i=0}^{k} \left( X \left[ \text{sgn}((\hat{r}^i)^T X_{j_i}) e_{j_i} - \frac{1}{\delta} \hat{\beta}^i \right] \right)^T (\hat{r}^i - \hat{r}_{LS}) \\
= \frac{1}{k+1} \sum_{i=0}^{k} \left( \text{sgn}((\hat{r}^i)^T X_{j_i}) X_{j_i} - \frac{1}{\delta} (X \hat{\beta}^i) \right)^T \hat{r}^i \\
= \frac{1}{k+1} \sum_{i=0}^{k} \left[ \|X^T\hat{r}^i\|_\infty - \frac{1}{\delta} (\hat{r}^i)^T X \hat{\beta}^i \right] \\
= \frac{1}{k+1} \sum_{i=0}^{k} \omega_\delta(\hat{\beta}^i),
\]

where the second equality uses the fact that \( X^T \hat{r}_{LS} = 0 \) from (5.5) and the fourth equality uses the definition of \( \omega_\delta(\beta) \) from (C.17).

Let us now evaluate the right side of (C.11). We have

\[
\|x^0 - x\|_2 = \|\hat{r}^0 - \hat{r}_{LS}\|_2 = \|y - (y - X \hat{\beta}_{LS})\|_2 = \|X \hat{\beta}_{LS}\|_2.
\]

Also, it holds that

\[
\|g^i\|_2 = \|\text{sgn}((\hat{r}^i)^T X_{j_i}) X_{j_i} - \frac{1}{\delta} (X \hat{\beta}^i)\|_2 \leq \|X_{j_i}\|_2 + \|X (\hat{\beta}^i)\|_2 \leq 1 + \frac{1}{\delta} \|X\|_1 \|\hat{\beta}^i\|_1 \leq 1 + \|X\|_1 \leq 2,
\]

where the third inequality follows since \( \|\hat{\beta}^i\|_1 \leq \delta \) from (C.23) and the second and fourth inequalities follow from the assumption that the columns of \( X \) have been normalized to have unit \( \ell_2 \) norm. Therefore \( G = 2 \) is a uniform bound on \( \|g^i\|_2 \). Combining the above, inequality (C.11) implies that after running R-FS\(_{\varepsilon, \delta}\) for \( k \) iterations, it holds that:

\[
\min_{i \in \{0, \ldots, k\}} \omega_\delta(\hat{\beta}^i) \leq \frac{1}{k+1} \sum_{i=0}^{k} \omega_\delta(\hat{\beta}^i) \leq \frac{\|X \hat{\beta}_{LS}\|_2^2}{2(k+1)\varepsilon} + \frac{2\varepsilon}{2} = \frac{\|X \hat{\beta}_{LS}\|_2^2}{2\varepsilon(k+1)} + 2\varepsilon,
\]

where the first inequality is elementary arithmetic and the second inequality is the application of (C.11). Now let \( i \) be the index obtaining the minimum in the left-most side of the above. Then it follows from part (iii) of Proposition C.4 that

\[
L_n(\hat{\beta}^i) - L_n^{*} \leq \frac{\delta}{n} \omega_\delta(\hat{\beta}^i) \leq \frac{\delta \|X \hat{\beta}_{LS}\|_2^2}{2n\varepsilon(k+1)} + \frac{2\varepsilon}{n},
\]

where \( \delta = \frac{\delta}{n} \) and \( \omega_\delta(\hat{\beta}^i) \) is the function defined in (C.26).
which proves item (i) of the theorem.

To prove item (ii), note first that if $\hat{\beta}^*_\delta$ is a solution of the LASSO problem (5.1), then it holds that $\|\hat{\beta}^*_\delta\|_1 \leq \delta$ (feasibility) and $\omega_\delta(\hat{\beta}^*_\delta) = 0$ (optimality). This latter condition follows easily from the optimality conditions of linearly constrained convex quadratic problems, see [2] for example. Setting $\hat{r}^*_\delta = y - X\hat{\beta}^*_\delta$, the following holds true:

$$\|X\hat{\beta}^i - X\hat{\beta}^*_\delta\|_2^2 = 2n \left( L_n(\hat{\beta}^i) - L_n(\hat{\beta}^*_\delta) + (\hat{r}^*_\delta)^T X (\hat{\beta}^i - \hat{\beta}^*_\delta) \right)$$

$$= 2n \left( L_n(\hat{\beta}^i) - L^*_n,\delta - \delta \|X^T \hat{r}^*_\delta\|_\infty + (\hat{r}^*_\delta)^T X \hat{\beta}^i \right)$$

$$\leq 2n \left( L_n(\hat{\beta}^i) - L^*_n,\delta - \delta \|X^T \hat{r}^*_\delta\|_\infty + \|X^T \hat{r}^*_\delta\|_\infty \|\hat{\beta}^i\|_1 \right)$$

$$\leq 2n \left( L_n(\hat{\beta}^i) - L^*_n,\delta - \delta \|X^T \hat{r}^*_\delta\|_\infty + \delta \|X^T \hat{r}^*_\delta\|_\infty \right)$$

$$= 2n \left( L_n(\hat{\beta}^i) - L^*_n \right)$$

$$\leq \frac{\delta \|X^T \hat{r}^*_\delta\|_\infty^2}{\varepsilon (k+1)} + 4\delta \varepsilon ,$$

where the first equality is from direct arithmetic substitution, the second equality uses the fact that $\omega_\delta(\hat{\beta}^*_\delta) = 0$ whereby $(\hat{r}^*_\delta)^T X \hat{\beta}^*_\delta = \delta \|X^T \hat{r}^*_\delta\|_\infty$, the first inequality follows by applying Holder’s inequality to the last term of the second equality, and the final inequality is an application of (C.26). Item (ii) then follows by taking square roots of the above.

Item (iii) is essentially just (C.23). Indeed, since $i \leq k$ we have:

$$\|\hat{\beta}^i\|_1 \leq \varepsilon \sum_{j=0}^{i-1} \left(1 - \frac{\varepsilon}{\delta}\right)^j \leq \varepsilon \sum_{j=0}^{k-1} \left(1 - \frac{\varepsilon}{\delta}\right)^j = \delta \left[1 - (1 - \frac{\varepsilon}{\delta})^k\right] \leq \delta .$$

(Note that we emphasize the dependence on $k$ rather than $i$ in the above since we have direct control over the number of boosting iterations $k$.) Item (iv) of the theorem is just a
restatement of the sparsity property of $R$-$FS_{\epsilon,\delta}$.
Appendix D

Supplement to Chapter 6

D.1 Useful Properties Regarding LogitBoost

Proposition D.1. Consider the function \( d(\cdot) : [0, 1]^n \rightarrow \mathbb{R} \) given by (6.15). It holds that \( d(\cdot) \) is a \( \sigma := \frac{4}{n} \)-strongly convex function with respect to the Euclidean norm \( \|w\| := \|w\|_2 \).

Proof: Let \( G := [0, 1]^n \), consider any point \( w \in \text{int}G \), and let \( H(w) \) denote the Hessian matrix of \( d(\cdot) \) at \( w \). The off-diagonal components of \( H(w) \) are all zero, and the \( i^{th} \) diagonal component is \( H_{ii}(w) = \frac{1}{n} \frac{1}{w_i(1-w_i)} \geq \frac{4}{n} \), and hence \( v^T[H(w)]v \geq \frac{4}{n} v^Tv \) for any \( v \). Now let \( y \in \text{int}G \). Invoking an intermediate value theorem of calculus, there exists a scalar \( c \in [0, 1] \) for which it holds that:

\[
d(y) = d(w) + \nabla d(w)^T(y - w) + \frac{1}{2} (y - w) H(w + c(y - w))(y - w),
\]

whereby:

\[
d(y) \geq d(w) + \nabla d(w)^T(y - w) + \frac{1}{2} (y - w) \left[ \frac{4}{n} I \right] (y - w) = d(w) + \nabla d(w)^T(y - w) + \frac{1}{2} \frac{4}{n} \|y - w\|_2^2.
\]

This proves that \( d(\cdot) \) is \( \sigma = \frac{4}{n} \)-strongly convex on \( \text{int}G \), and a continuity argument establishes
the result for all of $G$. □

### D.2 Proof of Lemma 6.1

**Proof:** It is easy to verify through optimality conditions that the unique optimal solution to the maximization problem in (6.16) is given by (6.17), and direct substitution and simplification of terms then yields the equality in (6.16), which shows (i) and (ii). The formula for the gradient in (iii) can be derived directly using elementary rules of calculus. Alternatively, given the representation (6.16), it follows from Danskin’s Theorem (also known as the Envelope Theorem), see [9], that

$$\nabla L_n(\beta) = -\frac{1}{n} X^T w^*(\beta).$$

Again using the representation (6.16), Theorem 1 of [74] implies that the Lipschitz constant $L$ of the gradient of $L_n(\cdot)$ is at most $\left[\frac{1}{n} \|X\|_{1,2}^2 \right] / \sigma$ where $\sigma$ is the strong convexity parameter of the function $d(\cdot)$. From Proposition D.1 in the Appendix it holds that $\sigma \geq \frac{4}{n}$, which implies that $L \leq \left[\frac{1}{n} \|X\|_{1,2}^2 \right] / \sigma \leq \left[\frac{1}{4n} \right] \|X\|_{1,2}^2$, which proves (iv). Finally, (v) is a well-known implication that follows from the fundamental theorem of calculus (also see Proposition A.2). □

### D.3 Properties of the Log-Exponential Loss Function

**Lemma D.1. (Properties of the Log-Exponential Loss Function)** Recall that $A$ is the feature matrix defined in section 6.3 and $e(\cdot) : \Delta_m \to \mathbb{R}$ is the entropy function defined by $e(w) := \sum_{i=1}^{m} w_i \ln(w_i) + \ln(m)$. The log-exponential loss function $L_l(\cdot) : \mathbb{R}^n \to \mathbb{R}$, defined by:

$$L_l(\lambda) = \log \left( \frac{1}{m} \sum_{i=1}^{m} \exp \left( -(A\lambda)_i \right) \right)$$

satisfies the following properties:
(i) Conjugate representation:

\[ L_l(\lambda) = \max_{w \in \Delta_m} \{-w^T A \lambda - e(w)\} \quad \text{for any } \lambda \in \mathbb{R}^n. \quad (D.1) \]

Furthermore, for any \( \lambda \in \mathbb{R}^n \), the unique optimal solution to \( \max_{w \in \Delta_m} \{-w^T A \lambda - e(w)\} \) is given by:

\[ w^*(\lambda)_i = \frac{\exp(-(A\lambda)_i)}{\sum_{l=1}^m \exp(-(A\lambda)_l)} \quad i = 1, \ldots, m , \quad (D.2) \]

and

\[ \nabla L_l(\lambda) = -A^T w^*(\lambda) . \quad (D.3) \]

(ii) Relation to the margin:

\[ -p(\lambda) - \ln(m) \leq L_l(\lambda) \leq -p(\lambda) \quad \text{for any } \lambda \in \mathbb{R}^n , \]

where \( p(\cdot) \) is the margin function defined in (6.32).

(iii) Smoothness (Lipschitz gradient):

When \( \mathbb{R}^n \) is considered with the \( \ell_1 \) norm, the Lipschitz constant of the gradient of \( L_l(\cdot) \) is \( \|A\|_1,\infty = \max_{i,j} |A_{ij}| \leq 1 \), i.e., the gradient of \( L_l(\cdot) \) satisfies:

\[ \|\nabla L_l(\lambda) - \nabla L_l(\lambda')\|_\infty \leq \|\lambda - \lambda'\|_1 \quad \text{for any } \lambda, \lambda' \in \mathbb{R}^n , \]

which also implies the following:

\[ L_l(\lambda') \leq L_l(\lambda) + \nabla L_l(\lambda)^T (\lambda' - \lambda) + \frac{1}{2} \|\lambda' - \lambda\|_1^2 \quad \text{for any } \lambda, \lambda' \in \mathbb{R}^n . \]

**Proof:** It is easy to verify through optimality conditions that the unique optimal solution to \( \max_{w \in \Delta_m} \{-w^T A \lambda - e(w)\} \) is given by (D.2). Thus for any \( \lambda \in \mathbb{R}^n \), letting \( Z := \)
\[ \sum_{i=1}^{m} \exp(-(A\lambda)_i) \] be the normalization constant, we have:

\[
\max_{w \in \Delta_m} \left\{ -w^T A\lambda - \epsilon(w) \right\} = -\frac{1}{Z} \sum_{i=1}^{m} \exp(-(A\lambda)_i) (A\lambda)_i - \frac{1}{Z} \sum_{i=1}^{m} \exp(-(A\lambda)_i) (- (A\lambda)_i - \ln(Z)) - \ln(m)
\]

\[
= \frac{\ln(Z)}{Z} \sum_{i=1}^{m} \exp(-(A\lambda)_i) - \ln(m)
\]

\[
= \ln(Z) - \ln(m)
\]

\[
= \ln \left( \frac{\sum_{i=1}^{m} \exp(-(A\lambda)_i)}{m} \right) = L_l(\lambda).
\]

Given the representation (D.1), it follows from Danskin’s Theorem (also known as the Envelope Theorem), see Bertsekas [9], that \( \nabla L_l(\lambda) = -A^T w^*(\lambda) \). This completes the proof of (i).

Again using the representation (D.1), Theorem 1 of [74] implies that the Lipschitz constant of the gradient of \( L_l(\cdot) \) is \( \|A\|_{1,\infty} \leq 1 \). The second part of (iii) is a well-known implication that follows from the fundamental theorem of calculus (also see Proposition A.2).

To prove (ii), note that the minimum of \( \epsilon(\cdot) \) on \( \Delta_m \) is achieved at \( \left( \frac{1}{m}, \ldots, \frac{1}{m} \right) \), thus \( 0 \leq \epsilon(w) \leq \ln(m) \) for all \( w \in \Delta_m \). Therefore for any \( \lambda \in \mathbb{R}^n \):

\[
-p(\lambda) - \ln(m) = - \min_{w \in \Delta_m} \left\{ w^T A\lambda \right\} - \ln(m)
\]

\[
= \max_{w \in \Delta_m} \left\{ -w^T A\lambda - \ln(m) \right\}
\]

\[
\leq \max_{w \in \Delta_m} \left\{ -w^T A\lambda - \epsilon(w) \right\} = L_l(\lambda),
\]

and

\[
L_l(\lambda) = \max_{w \in \Delta_m} \left\{ -w^T A\lambda - \epsilon(w) \right\} \leq \max_{w \in \Delta_m} \left\{ -w^T A\lambda \right\} = -p(\lambda).
\]
D.4 Other Useful Properties Regarding AdaBoost

**Lemma D.2.** The edge function $f(\cdot) : \Delta_m \to \mathbb{R}$ defined in (6.45) is 1-Lipschitz with respect to the $\ell_1$ norm.

**Proof:** Let $w^1, w^2 \in \Delta_m$ and let $j_1 \in \arg\max_{j \in \{1, \ldots, n\}} (w^1)^T A_j$, $j_2 \in \arg\max_{j \in \{1, \ldots, n\}} (w^2)^T A_j$. Then,

$$f(w^1) - f(w^2) = (w^1)^T A_{j_1} - (w^2)^T A_{j_2} \leq (w^1)^T A_{j_1} - (w^2)^T A_{j_1} = (w^1 - w^2)^T A_{j_1} \leq \|A_{j_1}\|_{\infty} \|w^1 - w^2\|_1 \leq \|w^1 - w^2\|_1,$$

where the last inequality uses the fact that all of the entries of $A$ are in $[-1, 1]$. Likewise, we can show that $f(w^2) - f(w^1) \leq \|w^1 - w^2\|_1$, which shows that $|f(w^1) - f(w^2)| \leq \|w^1 - w^2\|_1$. \qed

**Lemma D.3.** Let $e(\cdot) : \Delta_m \to \mathbb{R}$ be the entropy function, defined by $e(w) = \sum_{i=1}^m w_i \ln(w_i) + \ln(m)$, with induced Bregman distance $D(\cdot, \cdot)$, and let $w^0 = (1/m, \ldots, 1/m)$. Then $\max_{w \in \Delta_m} D(w, w^0) = \ln(m)$.

**Proof:** Clearly $e(w^0) = \ln(1/m) + \ln(m) = 0$ and since $\nabla e(w^0)_i = 1 + \ln(1/m) = 1 - \ln(m)$, we have for any $w \in \Delta_m$:

$$\nabla e(w^0)^T (w - w^0) = (1 - \ln(m)) \sum_{i=1}^m (w_i - 1/m) = (1 - \ln(m))(1 - 1) = 0 .$$

Thus we have:

$$D(w, w^0) = e(w) - e(w^0) - \nabla e(w^0)^T (w - w^0) = e(w) = \sum_{i=1}^m w_i \ln(w_i) + \ln(m) \leq \ln(m) .$$
Furthermore, the maximum is achieved at $e_1 = (1, 0, \ldots, 0)$. □
Bibliography


