UNIVERSITY OF CALIFORNIA

Los Angeles

Analytic methods for large-scale data

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Denali Marie Molitor

2021

ABSTRACT OF THE DISSERTATION

Analytic methods for large-scale data

by

Denali Marie Molitor
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2021
Professor Deanna Hunter, Chair

Methods that analyze large-scale data and make predictions based on data are increasingly prevalent in a variety of industries. These methods are often complex, rely on a variety of subroutines and are applied in settings for which they lack theoretical guarantees. Additionally, storage requirements and computational speed are crucial to the feasibility of methods at large-scales.

Here, we derive theoretical guarantees for machine learning and data analytic subroutines such as matrix completion and optimization. We also extend the development of classification methods for binary, compressed data. More specifically, we consider the following. We analyze a regularized variant of the standard nuclear-norm minimization problem for low-rank matrix completion for settings in which smaller entries are more likely to be missing. We derive convergence guarantees for a gradient descent method for solving support vector machines and compare the derived convergence guarantees to those of existing strategies. We analyze adaptive sampling strategies for sketch-and-project methods for solving large-scale least-squares problems. We develop hierarchical and iterative extensions to the simple classification method for binary data introduced in [NSW17].

The dissertation of Denali Marie Molitor is approved.

Stanley Osher

Guido Montufar

Arash Amini

Deanna Hunter, Committee Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. Deanna Needell for her guidance and support during my graduate career. I am thankful to the member's of Prof. Needell's research group for creating a supportive and collaborative research community. I am thankful to Prof. Chris Anderson for his incredible teaching and advising during my early years at UCLA. I would also like to thank my collaborators.

Chapter 2 is a version of [MN18b], Chapter 5 is a version of [MN18a], and Chapter 6 is a version of [MN20]. These works are joint with Prof. Deanna Needell, who advised and proposed the projects.

Chapter 3 is a version of [MNW19] and is joint work with Prof. Rachel Ward and Prof. Deanna Needell, who proposed the project and contributed the initial mathematical investigation.

Chapter 4 is a version of [GMM19] and is joint work with Prof. Robert Gower, Jacob Moorman, and Prof. Deanna Needell. Prof. Robert Gower proposed the project and made significant contributions to the mathematical analysis. Jacob Moorman and I implemented numerical experiments, contributed to the mathematical analysis and manuscript.

I also had the pleasure of working with numerous students as an assistant mentor for the UCLA Computational and Applied Mathematics Research Experience for Undergraduates. I am thankful to Jamie Haddock for serving as the lead mentor for the program and Deanna Needell for organizing, proposing interesting projects and advising throughout the program.

I am grateful to the UCLA Graduate Division (Eugene V. Cota-Robles Fellowship), UCLA Office of the Deans (Charles E. and Sue K. Young Graduate Student Fellowship Award), and National Science Foundation (NSF CAREER DMS #1348721 and NSF BIGDATA DMS #1740325) for financial support during my PhD.

VITA

2014	B.A. (Mathematics), Colorado College.
2014-2015	Mathematics Paraprofessional, Colorado College.
2016-2018	Teaching Assistant, Mathematics Department, UCLA. Taught sections of Applied Numerical Methods (MATH151A), Machine Learning (MATH156) and Introduction to Computing (PIC10A).
2018	Assistant Mentor, Research Experience for Undergraduates, Mathematics Department, UCLA
2018	Teaching Assistant, Representations of High Dimensional Data, Mathematical Sciences Research Institute
2018	M.S. (Mathematics), UCLA.
2019	Research Intern, Google LLC.
2018-Present	Coordinator, Women in Mathematics and Women in Mathematics Mentorship Program, UCLA.
2018-Present	Research Assistant, Mathematics Department, UCLA.

PUBLICATIONS

R. Gower, D. Molitor, J. Moorman, and D. Needell. “Adaptive sketch-and-project methods for solving linear systems.” arXiv preprint arXiv:1909.03604 Sept. 2019

D. Molitor, D. Needell, R. Ward. “Bias of gradient descent for the hinge loss.” Applied Mathematics and Optimization (2020). https://doi.org/10.1007/s00245-020-09656-5

M. Gao, J. Haddock, D. Molitor, D. Needell, E. Sadovnik, T. Will, R. Zhang. “Neural nonnegative matrix factorization for hierarchical multilayer topic modeling.” Proc. IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2019.

J. Haddock, D. Molitor, D. Needell, S. Sambandam, J. Song, and S. Sun. “On inferences from completed data.” Proc. Information Theory and Approximation Workshop, Feb. 2019.

J. Moorman, T. Tu, D. Molitor, D. Needell. “Randomized Kaczmarz with averaging.” Proc. Information Theory and Approximation Workshop, Feb. 2019.

D. Molitor, D. Needell. “An iterative method for classification of binary data.” Information and Inference. To appear.

G. Plumb, D. Molitor, A. Talwalkar “Supervised local modeling for interpretability.” Proc. Neural Information Processing Systems (NeurIPS), Dec. 2018.

D. Molitor, D. Needell. “Hierarchical classification using binary data.” AAAI Magazine Special Issue on Deep Models, Machine Learning and Artificial Intelligence Applications in National and International Security, June 2018.

D. Molitor, D. Needell.“Matrix completion for structured observations.” Proc. Information Theory and Approximation, La Jolla CA, Feb. 2018.

R. Strichartz, N. Ott, D. Molitor. “Using peano curves to construct Laplacians on fractals”, Dec. 2015, Fractals, Vol. 23, No. 4.

D. Molitor, M. Steel, A. Taylor, “The structure of symmetric N-player games when influence and independence collide,” Jan. 2015, Advances in Applied Mathematics, Vol. 62, 15-40.

D. Maxin, L. Berec, A. Bingham, D. Molitor, J. Pattyson, “Is more better? Higher sterilization of infected hosts need not result in reduced pest population,” June 2014, J. Math. Biol.

CHAPTER 1

Introduction

As the magnitude of data collection and the influence that data has on our lives increases, understanding methods for analyzing, processing and manipulating data is critical. For example, decisions based on surveillance data and facial recognition, financial data and credit or loan approvals, and medical data and treatment plans can have significant impacts on lives. Here, we study data analytic subroutines and methods for classification. More specifically, we consider methods for inferring entries of incomplete data, extensions to a classification method for binary data and iterative methods for solving large linear systems of equations.

Developing a solid understanding of the assumptions and consequences of algorithmic choices in data analysis is imperative. Gaps in understanding can lead to unexpected outputs and results which could have extreme negative consequences. Often, methods are applied in settings that lack performance guarantees. Two such settings are matrix completion and optimization.

Datasets often include missing entries, whose values are unknown. A popular method for inferring missing entries assumes that the matrix to be recovered is approximately low-rank. Recovery guarantees for low-rank matrix completion typically assume that the missing entries occur uniformly at random. This assumption is highly unrealistic for many applications. For example, less popular movies may be less likely to receive viewer ratings and medical patients may be less likely to answer questions about symptoms that they are not experiencing. If there is known structure to the missing entries, we can ideally incorporate this into the recovery method to improve results. In Chapter 2, we discuss matrix completion when entries are missing in a structured manner and propose a regularized variant to accommodate this structure. This work is extended in [HMN19] to consider how the error introduced by various matrix completion strategies affects the conclusions of statistical inferences.

Many machine learning models require solving large-scale optimization problems in order to learn appropriate model parameters. The choice of optimization method can affect the learned parameters and ultimate model.

Gradient descent is a classical method for finding the minimum of differentiable convex functions. In Chapter 3, we propose a gradient descent method for minimizing the support vector machine (SVM) hinge loss [CV95]. We analyze the convergence of the proposed optimization strategy to the maximal separating hyperplane and compare it to existing approaches and their convergence guarantees.

Stochastic gradient descent (SGD) is a variant of gradient descent that is especially useful for optimization problems of the form $F(x) = \sum_{i=1}^{n} f_i(x)$ with n very large. Stochastic gradient descent randomly selects an index i or set of indices and performs a gradient descent update with respect to the selected $f_i(x)$ terms, thus avoiding the expensive computation of the full gradient of F at each iteration. While convergence guarantees for gradient descent and SGD require convexity of the objective function F, these methods are often applied to optimize non-convex functions such as those that arise when training neural networks. Despite the lack of guarantees in the non-convex setting, SGD and its variants have been shown to typically work well in practice.

A key component of SGD is the selection of the random index or indices i at each iteration. In the least-squares setting, SGD with a reweighting component has been shown to be equivalent to the randomized Kaczmarz method (RK) [NSW15]. Randomized Kacmzarz is a specific example of a more general class of methods known as sketch-and-project methods. In Chapter 4, we analyze the convergence of sketch-and-project methods with a variety of sampling strategies for selecting indices i that depend on the current approximation to the solution.

Classification is a common task for machine learning methods. In classification, one aims to assign the correct label to a given data point. For example, correctly labeling images of cats versus images of dogs. [NSW17] introduces a classification method intended for binary data. Such classification methods are advantageous when storing or collecting raw data is infeasible and thus only compressed binary measurements are available. Chapter 5 considers an extension of the method proposed in [NSW17] for data that has hierarchical class structure and Chapter 6 proposes an iterative extension that leads to improved accuracy at the cost of additional computation.

CHAPTER 2

Matrix Completion for Structured Data

2.1 Introduction

Data acquisition and analysis is ubiquitous, but data often contains errors and can be highly incomplete. For example, if data is obtained via user surveys, people may only choose to answer a subset of questions. Ideally, one would not want to eliminate surveys that are only partially complete, as they still contain potentially useful information. For many tasks, such as certain regression or classification tasks, one may require complete or completed data [SG02]. Alternatively, consider the problem of collaborative filtering, made popular by the classic Netflix problem [BL07, BK07, KBV09], in which one aims to predict user ratings for unseen movies based on available user-movie ratings. In this setting, accurate data completion is the goal, as opposed to a data pre-processing task. Viewing users as the rows in a matrix and movies as the columns, we would like to recover unknown entries of the resulting matrix from the subset of known entries. This is the goal in many types of other applications, ranging from systems identification [LV09] to sensor networks [BLW06, Sch86, Sin08]. This task is known as *matrix completion* [Rec11]. If the underlying matrix is low-rank and the observed entries are sampled uniformly at random, one can achieve exact recovery with high probability under mild additional assumptions by using nuclear norm minimization (NNM) [CT10, RFP10, CR09, Gro11, CP10].

For many applications, however, we expect *structural differences* between the observed and unobserved entries, which violate these classical assumptions. By structural differences, we mean that whether an entry is observed or unobserved need not be random or occur by some uniform selection mechanism. Consider again the Netflix problem. Popular, or well-received movies are more likely to have been rated by many users, thus violating the assumption of uniform sampling of observed entries across movies. On the flip side, a missing entry may indicate a user's lack of interest in that particular movie. Similarly, in sensor networks, entries may be missing because of geographic limitations or missing connections; in survey data, incomplete sections may be irrelevant or unimportant to the user. In these settings, it is then reasonable to expect that missing entries have lower values[1] than observed entries.

In this work, we propose a modification to the traditional NNM for matrix completion that still results in a semi-definite optimization problem, but also encourages lower values among the unobserved entries. We show that this method works better than NNM alone under certain sampling conditions.

2.1.1 Nuclear Norm Matrix Completion

Let $M \in \mathbb{R}^{n_1 \times n_2}$ be the unknown matrix we would like to recover and Ω be the set of indices of the observed entries. Let $\mathcal{P}_\Omega : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$, where

$$[\mathcal{P}_\Omega]_{ij} = \begin{cases} M_{ij} & (i,j) \in \Omega \\ 0 & (i,j) \notin \Omega \end{cases}$$

as in [CT10]. In many applications, it is reasonable to assume that the matrix M is low-rank. For example, we expect that relatively few factors contribute to a user's movie preferences as compared to the number of users or number of movies considered. Similarly, for health data, a few underlying features may contribute to many observable signs and symptoms.

The minimization,

$$\widehat{M} = \underset{A}{\operatorname{argmin}} \operatorname{rank}(A) \text{ s.t. } \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(M)$$

recovers the lowest rank matrix that matches the observed entries exactly. Unfortunately,

[1]Of course, some applications will tend to have *higher* values in missing entries, in which case our methods can be scaled accordingly.

this minimization problem is NP-hard, so one typically uses the convex relaxation

$$\widehat{M} = \underset{A}{\operatorname{argmin}} \, ||A||_* \text{ s.t. } \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(M), \tag{2.1}$$

where $||\cdot||_*$ is the nuclear norm, given by the sum of the singular values, i.e. $||X||_* := \sum_i \sigma_i(X)$ [CT10, RFP10, CP10, CR09].

2.1.2 Matrix Completion for Structured Observations

We propose adding a regularization term on the unobserved entries to promote adherence to the structural assumption that we expect these entries to be close to 0. We solve

$$\widetilde{M} = \underset{A}{\operatorname{argmin}} \, ||A||_* + \alpha||\mathcal{P}_{\Omega^C}(A)|| \text{ s.t. } \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(M), \tag{2.2}$$

where $\alpha > 0$ and $|| \cdot ||$ is an appropriate matrix norm. For example, if we expect most of the unobserved entries to be 0, but a few to be potentially large in magnitude, the entrywise L_1 norm $||M||_1 = \sum_{ij} |M_{ij}|$ is a reasonable choice.

2.1.3 Matrix Completion with Noisy Observations

In reality, we expect that our data is corrupted by some amount of noise. We assume the matrix M, that we would like to recover, satisfies

$$\mathcal{P}_\Omega Y = \mathcal{P}_\Omega M + \mathcal{P}_\Omega Z,$$

where $\mathcal{P}_\Omega Y$ are the observed values, M is low-rank and $\mathcal{P}_\Omega Z$ represents the noise in the observed data. In [CP10], Candés and Plan suggest using the following minimization to recover the unknown matrix:

$$\widehat{M} = \underset{A}{\operatorname{argmin}} \, ||A||_* \text{ s. t. } ||\mathcal{P}_\Omega(M - A)||_F < \delta. \tag{2.3}$$

Recall, $||X||_F = \sqrt{\sum_{ij} X_{ij}^2}$. The formulation above is equivalent to

$$\widehat{M} = \underset{A}{\operatorname{argmin}} \, ||\mathcal{P}_\Omega(M - A)||_F + \rho||A||_* \tag{2.4}$$

for some $\rho = \rho(\delta)$. The latter minimization problem is generally easier to solve in practice [CP10].

In order to account for the assumption that the unobserved entries are likely to be close to zero, we again propose adding a regularization term on the unobserved entries and aim to solve

$$\widetilde{M} = \underset{A}{\operatorname{argmin}} ||\mathcal{P}_{\Omega}(M - A)||_F + \rho||A||_* + \alpha||\mathcal{P}_{\Omega^C}(A)||. \tag{2.5}$$

2.2 Numerical Results

2.2.1 Recovery without Noise

We first investigate the performance of Equation (2.2) when the observed entries are exact, i.e. there is no noise or errors in the observed values. In Figure 2.1, we consider low-rank matrices $M \in \mathbb{R}^{30\times 30}$. To generate M of rank r, we take $M = M_L M_R$, where $M_L \in \mathbb{R}^{30\times r}$ and $M_R \in \mathbb{R}^{r\times 30}$ are sparse matrices (with density 0.3 and 0.5, respectively) and whose nonzero entries are uniformly distributed at random between zero and one. We subsample from the zero and nonzero entries of the data matrix at various rates to generate a matrix with missing entries. We compare performance of Equation (2.2) using L_1 regularization on the unobserved entries with standard NNM and report the error ratio $||\widetilde{M} - M||_F / ||\widehat{M} - M||_F$ for various sampling rates, where $\widetilde{M}$ and $\widehat{M}$ are the solutions to Equation (2.2) and Equation (2.1), respectively. The regularization parameter α used is selected optimally from the set $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ (discussed below). Values below one in Figure 2.1 indicate that the minimization with L_1 regularization outperforms standard NNM. Results are averaged over ten trials. As expected, we find that if the sampling of the nonzero entries is high, then the modified method Equation (2.2) is likely to outperform standard NNM.

We choose the parameter α, for the regularization term, to be optimal among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and report the values used in Figure 2.2. For large α, the recovered matrix will approach that for which all unobserved entries are predicted to be zero, and as α becomes close to zero, recovery by Equation (2.2) approaches that of standard NNM.

When the sampling rate of the zero entries is low and the sampling of the nonzero entries is high, in addition to Equation (2.2) outperforming NNM, we also see that a larger value for α is optimal, supporting the claim that regularization improves performance. Higher α values are also sometimes optimal when the nonzero sampling rate is nearly zero. If there are very few nonzero entries sampled then the low-rank matrix recovered is likely to be very close to the zero matrix. In this setting, we expect that even with standard NNM the unobserved entries are thus likely to be recovered as zeros and so a larger coefficient on the regularization term will not harm performance. When α is close to zero, the difference in performance is minimal, as the regularization will have little effect in this case.

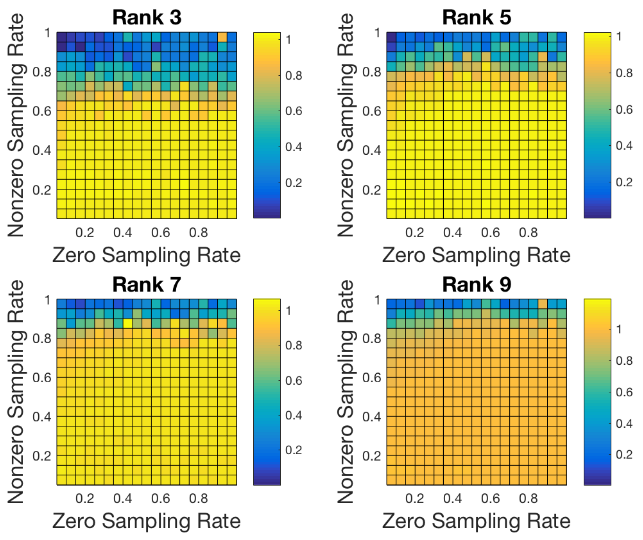


Figure 2.1: For $\widetilde{M}$ and $\widehat{M}$ given by Equation (2.2) and Equation (2.1), respectively, with L_1 regularization on the recovered values for the unobserved entries, we plot $||\widetilde{M} - M||_F / ||\widehat{M} - M||_F$. We consider 30x30 matrices of various ranks and average results over ten trials, with α optimal among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$.

2.2.2 Recovery with Noisy Observed Entries

We generate matrices as in the previous section and now consider the minimization given in Equation (2.4). Suppose the entries of the noise matrix Z are i.i.d. $N(0, \sigma^2)$. We set the

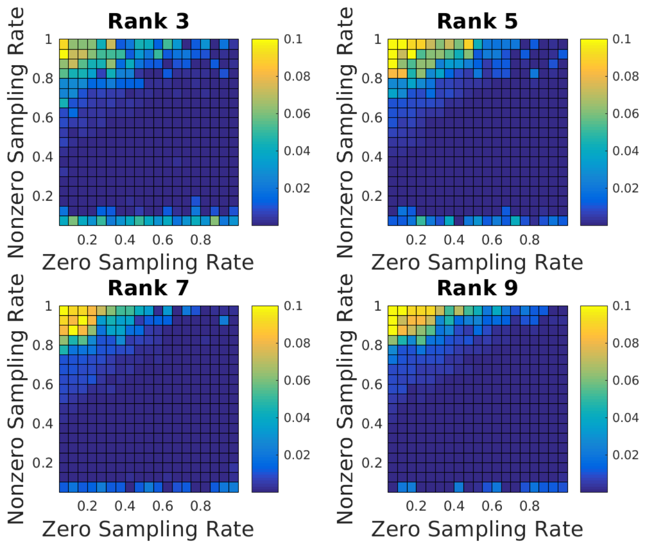


Figure 2.2: Average optimal α value among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ for the minimization given in Equation (2.2) with L_1 regularization on the recovered values for the unobserved entries. The matrices considered here are the same as in Figure 2.1.

parameter ρ, as done in [CP10], to be

$$\rho = (\sqrt{n_1} + \sqrt{n_2})\sqrt{\frac{|\Omega|}{n_1 n_2}}\sigma.$$

We specifically consider low-rank matrices $M \in \mathbb{R}^{30\times 30}$ generated as in the previous section and a noise matrix Z with i.i.d. entries sampled from $N(0, 0.01)$. Thus we set $\rho = 2\sqrt{\frac{|\Omega|}{30}} \cdot 0.1$. We again report $||\widetilde{M} - M||_F / ||\widehat{M} - M||_F$ for various sampling rates of the zero and nonzero entries of M in Figure 2.3. Here, $\widehat{M}$ and $\widetilde{M}$ are given by Equation (2.4) and Equation (2.5) respectively. We see improved performance with regularization when the sampling rate of the zero entries is low and the sampling of the nonzero entries is high.

2.2.3 Matrix recovery of health data

Next, we consider real survey data from 2126 patients responding to 65 particular questions provided by LymeDisease.org. Data used was obtained from the LymeDisease.org patient registry, MyLymeData, Phase 1, June 17, 2017. Question responses are integer values between zero and four and answering all questions was required, that is this subset of the data survey

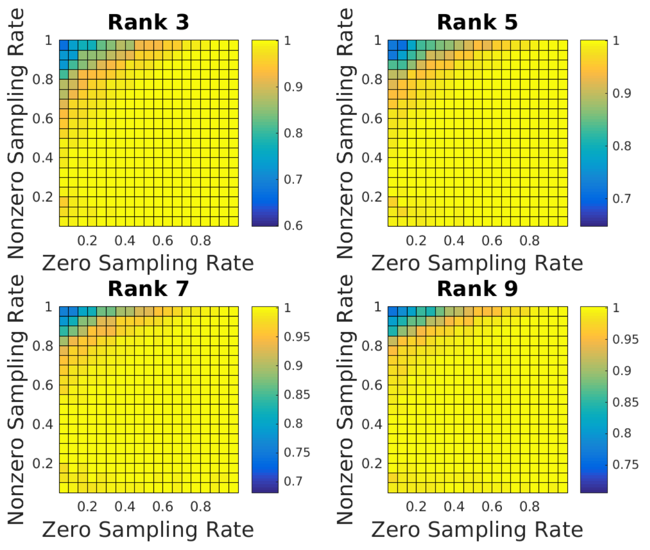


Figure 2.3: For $\widetilde{M}$ and $\widehat{M}$ given by Equation (2.2) and Equation (2.1), respectively, with L_1 regularization on the recovered values for the unobserved entries, we plot $||\widetilde{M} - M||_F/||\widehat{M} - M||_F$. We consider 30x30 matrices of various ranks with normally distributed i.i.d. noise with standard deviation $\sigma = 0.1$ added. We average results over ten trials and with α optimal among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$.

is complete (so we may calculate reconstruction errors). All patients have Lyme disease and survey questions ask about topics such as current and past symptoms, treatments and outcomes. For example, "I would say that currently in general my health is: 0-Poor, 1-Fair, 2-Good, 3-Very good, 4-Excellent." Although, this part of the data considered is complete, we expect that in general, patients are likely to record responses for particularly noticeable symptoms, while a missing response in a medical survey may indicate a lack of symptoms. Thus, in this setting, L_1 regularization of the unobserved entries is a natural choice.

Due to computational constraints, for each of the ten trials executed, we randomly sample 50 of these patient surveys to generate a 50x65 matrix. As in the previous experiments, we subsample from the zero and nonzero entries of the data matrix at various rates to generate a matrix with missing entries. We complete this subsampled matrix with both NNM Equation (2.1) and Equation (2.2) using L_1 regularization on the unobserved entries and report $||\widetilde{M} - M||_F / ||\widehat{M} - M||_F$, averaged over ten trials in Figure 2.4. The parameter α, for the regularization term, is chosen to be optimal among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and we report the values used in Figure 2.5.

The results for the Lyme disease data match closely those found in the synthetic experiments done with and without noise. Regularizing the L_1-norm of the unobserved entries improves performance if the sampling of non-zero entries is sufficiently high and sampling of zero entries is sufficiently low.

2.3 Analytical Remarks

We provide here some basic analysis of the regularization approach. First, in the simplified setting, in which all of the unobserved entries are exactly zero, the modified recovery given in Equation (2.2) will always perform at least as well as traditional NNM.

Proposition 2.3.1. *Suppose $M \in \mathbb{R}^{n_1 \times n_2}$ and Ω gives the set of index pairs of the observed entries. Assume that all of the unobserved entries are exactly zero, i.e. $\mathcal{P}_{\Omega^C}(M) = 0$. Then for*

$$\widehat{M} = \operatorname{argmin} ||A||_* \ s.t.\ \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(M),$$

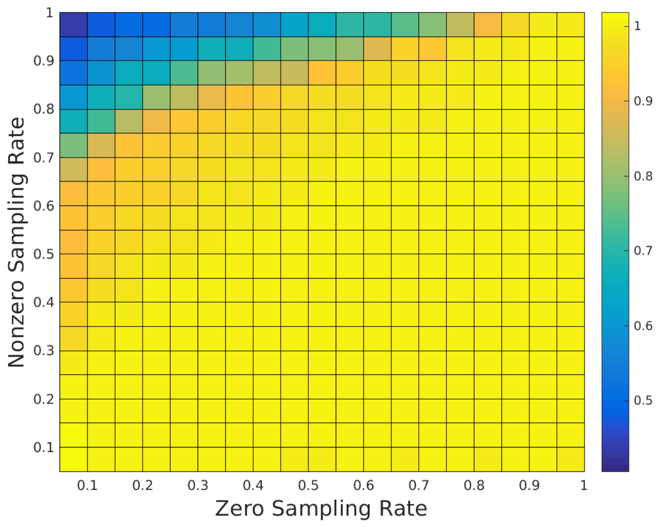


Figure 2.4: For $\widetilde{M}$ and $\widehat{M}$ given by Equation (2.2) and Equation (2.1), respectively, with L_1 regularization on the recovered values for the unobserved entries, we plot $||\widetilde{M} - M||_F / ||\widehat{M} - M||_F$. We consider 50 patient surveys with 65 responses each chosen randomly from 2126 patient surveys. We average results over ten trials and with α optimal among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$.

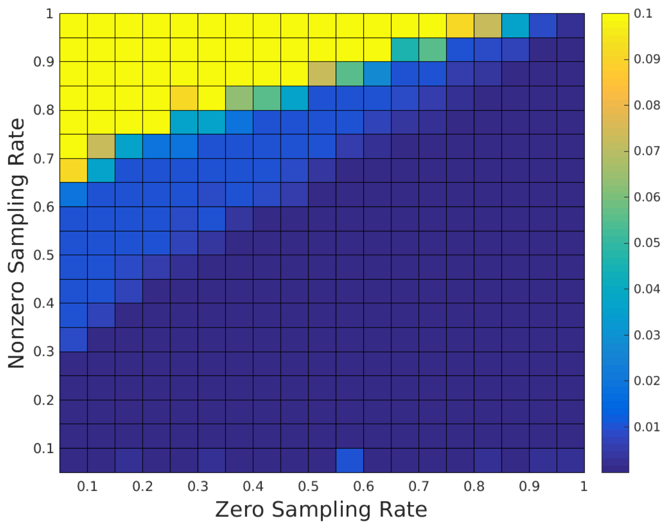


Figure 2.5: Average optimal α value among $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ for the minimization given in Equation (2.2) with L_1 regularization on the recovered values for the unobserved entries in Lyme patient data.

and

$$\widetilde{M} = \operatorname{argmin} ||A||_* + \alpha||\mathcal{P}_{\Omega^C}(A)|| \ s.t. \ \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(M),$$

we have

$$||\widetilde{M} - M|| \leq ||\widehat{M} - M||$$

for any matrix norm $||\cdot||$.

Proof. From the definitions of $\widehat{M}$ and $\widetilde{M}$,

$$||\widehat{M}||_* \leq ||\widetilde{M}||_*.$$

Using the inequality above,

$$\begin{aligned} ||\widetilde{M}||_* + \alpha||\mathcal{P}_{\Omega^C}(\widetilde{M})|| &\leq ||\widehat{M}||_* + \alpha||\mathcal{P}_{\Omega^C}(\widehat{M})|| \\ &\leq ||\widetilde{M}||_* + \alpha||\mathcal{P}_{\Omega^C}(\widehat{M})||. \end{aligned}$$

For $\alpha > 0$, we have

$$||\mathcal{P}_{\Omega^C}(\widetilde{M})|| \leq ||\mathcal{P}_{\Omega^C}(\widehat{M})||.$$

The desired result then follows since

$$\mathcal{P}_\Omega(\widetilde{M}) = \mathcal{P}_\Omega(\widehat{M}) = \mathcal{P}_\Omega(M)$$

and under the assumption that $\mathcal{P}_{\Omega^C}(M) = 0$, as

$$||\widetilde{M} - M|| = ||\mathcal{P}_{\Omega^C}(\widetilde{M})|| \leq ||\mathcal{P}_{\Omega^C}(\widehat{M})|| = ||\widehat{M} - M||.$$

□

2.3.1 Connection to Robust Principal Component Analysis (RPCA)

The program Equation (2.2) very closely resembles the method proposed in [CLM11], called Robust Principal Component Analysis (RPCA). RPCA is a modified version of traditional Principal Component Analysis that is robust to rare corruptions of arbitrary magnitude. In RPCA, one assumes that a low-rank matrix has some set of its entries corrupted. The goal is to recover the true underlying matrix despite the corruptions. More simply, for the observed matrix $Y \in \mathbb{R}^{n_1 \times n_2}$, we have the decomposition

$$Y = L + S,$$

where L is the low-rank matrix we would like to recover and S is a sparse matrix of corruptions. The strategy for finding this decomposition proposed in [CLM11] is

$$\operatorname*{argmin}_{L,S} ||L||_* + \alpha ||S||_1 \text{ s.t. } L + S = Y. \tag{2.6}$$

This method can be extended to the matrix completion setting, in which one would like to recover unobserved values from observed values, of which a subset may be corrupted. In this setting, [CLM11] proposes solving the following minimization problem

$$\operatorname*{argmin}_{L,S} ||L||_* + \alpha ||S||_1 \text{ s.t. } \mathcal{P}_\Omega(L + S) = \mathcal{P}_\Omega(Y).$$

We now return to our original matrix completion problem, in which we assume the observed entries to be exact. Let $M \in \mathbb{R}^{n_1 \times n_2}$ again be the matrix we aim to recover. If we expect the unobserved entries of M to be sparse, that is, only a small fraction of them to be nonzero, we can rewrite the minimization Equation (2.2) in a form similar to RPCA in which we know the support of the corruptions is restricted to the set Ω^C, i.e. $S = \mathcal{P}_{\Omega^C}(S)$. We then have,

$$\operatorname*{argmin}_{A,S} ||A||_* + \alpha ||S||_1 \text{ s.t. } A + S = \mathcal{P}_\Omega(M). \tag{2.7}$$

This strategy differs from traditional RPCA in that we assume the observed data to be free from errors and therefore know that the corruptions are restricted to the set of unobserved entries.

Directly applying Theorem 1.1 from [CLM11], we have the following result.

Proposition 2.3.2. *Suppose $M \in \mathbb{R}^{n_1 \times n_2}$ and $M = U\Sigma V^*$ gives the singular value decomposition of M. Suppose also*

$$\max_i ||U^* e_i||^2 \leq \frac{\mu r}{n_1}, \quad \max_i ||V^* e_i||^2 \leq \frac{\mu r}{n_2},$$

and

$$||UV^*||_\infty \leq \sqrt{\frac{\mu r}{n_1 n_2}},$$

where r is the rank of M, $||X||_\infty = \max_{i,j} |X_{i,j}|$, e_i is the i^{th} standard basis vector and μ is the incoherence parameter as defined in [CLM11]. Suppose that the set of observed entries, Ω, is uniformly distributed among all sets of cardinality of m and the support set of S_0 of non-zero unobserved entries is uniformly distributed among all sets of cardinality s contained in Ω^C. Then there is a numerical constant c such that with probability at least $1 - cn^{-10}$ the minimization in Equation (2.7) *with $\alpha = 1/\sqrt{n}$ achieves exact recovery, provided that*

$$\textit{rank}(L_0) \leq \rho_r n_{(2)} \mu^{-1} (\log n_{(1)})^{-2} \textit{ and } s \leq \rho_s n_{(1)} n_{(2)},$$

where ρ_r and ρ_s are positive numerical constants.

This proposition is a direct application of Theorem 1.1 in [CLM11] to the program given by Equation (2.7). Note that here, the corruptions are exactly the unobserved entries that are nonzero. Thus, if s, the number of nonzero unobserved entries is small, this result may be stronger than corresponding matrix completion results that instead depend on m, the larger, number of missing entries.

The authors of [CLM11] note that RPCA can be thought of as a more challenging version of matrix completion. The reasoning being, that in matrix completion we aim to recover the set of unobserved entries, whose locations are known, whereas in the RPCA setting, we have a set of corrupted entries, whose locations are unknown, and for which we would like to both identify as erroneous and determine their correct values. Figure 1 of [CLM11] provides numerical evidence that in practice RPCA does in fact require more stringent conditions to achieve exact recovery than the corresponding matrix completion problem. In image completion or repair, corruptions are often spatially correlated or isolated to specific regions of an image. In [LRZ12], the authors provide experimental evidence that incorporating an estimate of the support of the corruptions aids in recovery. By the same reasoning, we expect that a stronger result than suggested by Proposition 2.3.2 likely holds, as we do not make use of the fact that we are able to restrict the locations of the corruptions (nonzero, unobserved entries) to a subset of the larger matrix.

2.4 Discussion

For incomplete data in which we expect that unobserved entries are likely to be 0, we find that regularizing the values of the unobserved entries when performing NNM improves performance under various conditions. This improvement in performance holds for both synthetic data, with and without noise, as well as for Lyme disease survey data. We specifically investigate the performance of L_1 regularization on the unobserved entries as it is a natural choice for many applications.

Testing the validity of methods, such as Equation (2.2), on real data is challenging, since this setting hinges on the assumption that unobserved data is structurally different than observed data and would require having access to ground truth values for the unobserved entries. In this paper, we choose to take complete data and artificially partition it into observed and unobserved entries. Another way to manage this challenge is to examine performance of various tasks, such as classification or prediction, based on data that has been completed in different ways. In this setting, relative performance of different completion strategies will likely depend on the specific task considered. However, for many applications, one would like to complete the data in order to use it for a further goal. In this setting, judging the performance of the matrix completion algorithm by its effect on performance of the ultimate goal is very natural.

We offer preliminary arguments as to why we might expect the approach in Equation (2.2) to work well under the structural assumption that unobserved entries are likely to be sparse or small in magnitude, however, stronger theoretical results are likely possible. For example, we show that regularizing the values of the unobserved entries when performing NNM improves performance in the case when all unobserved entries are exactly zero, but based on empirical evidence we expect improved performance under more general conditions.

A range of papers, including [CT10, RFP10, CR09, Gro11], discuss the conditions under which exact matrix completion is possible under the assumption that the observed entries of the matrix are sampled uniformly at random. Under what reasonable structural assumptions on the unobserved entries might we still be able to specify conditions that will lead to exact recovery? We save such questions for future work.

Acknowledgments

The authors would like to thank LymeDisease.org for the use of data derived from MyLymeData to conduct this study. We would also like to thank the patients for their contributions to MyLymeData, and Anna Ma for her guidance in working with this data. In addition, the authors were supported by NSF CAREER DMS #1348721, NSF BIGDATA DMS #1740325, and MSRI NSF DMS #1440140.

CHAPTER 3

Bias of Gradient Descent for the Non-Smooth Hinge Loss

3.1 Introduction

Several recent works suggest that the optimization methods used in training models affect the model's ability to generalize through implicit biases to certain solutions [ZBH17, NTS14, HRS16, HHS17, PKL17, PLM18, HHS17, CCS17, CPS18]. In order to understand the effects of optimization methods in more complex and often non-convex settings such as for neural networks, it is natural to first understand their behavior in simpler settings, such as for least squares regression, logistic regression, and support vector machines (SVM) [SHN18, NLG19, GLS18]. In particular, gradient descent and its many variants, including the subgradient method, are popular choices for optimizing machine learning models and thus warrant careful study.

It was recently shown that gradient descent applied to the (unregularized) logistic regression problem for linearly separable data converges to the solution with maximal margin, while other choices of optimization method converge to different solutions [SHN18]. Convergence to the maximal-margin solution is desirable, as the margin is an important quantity for deriving generalization guarantees [BS99, Vap82, Vap99, VC74, Vap13]. The analysis of Soudry et al [SHN18] extends to additional loss functions, but requires particular properties, including smoothness and monotonicity. These assumptions do not hold, however, for non-differentiable functions such as the hinge loss objective, which is the loss function used in training SVM [CV95].

Here, we analyze the convergence to the maximal margin solution of a homotopic subgradient method applied to the non-smooth hinge loss. In particular we consider a method in which a number of subgradient updates are applied to the hinge loss with decreasing regularization. Although it is well known that the exact solutions of the regularized hinge loss converge to the hard-margin SVM solution as the regularization decreases to zero in the linearly separable case [RZH04, HRT04], we are unaware of results that provide explicit convergence rates for an iterative optimization algorithm, such as the subgradient method, that converges to the hard-margin SVM solution in a single pass of the regularization parameter λ. We provide such an analysis here, and demonstrate that the iterates of an averaged subgradient method applied to the regularized SVM loss with shrinking regularization parameters converge to the max-margin solution at a rate of $O\left(k^{-1/6+\delta}\right)$ for linearly separable data, where δ is any small positive constant.

For linearly separable data there exists $\lambda' > 0$ be such that the solution $\mathbf{w}_\lambda^*$ to the hinge loss with regularization parameter λ is equal to the true, hard-margin solution $\mathbf{w}^*$ for all $\lambda \leq \lambda'$ [RZH04,HRT04]. While λ' is constant for a fixed problem, knowing its value in advance is typically unrealistic. Additionally, if the data is not well separated, λ' can be very small. The homotopic subgradient method analyzed here depends on the value of λ' and converges at a rate of $O\left((\lambda')^{-2}k^{-1/6+\delta}\right)$. If one were to know the appropriate regularization parameter λ' in advance, the averaged subgradient method with appropriate fixed step sizes would converge in L_2 error at a rate of $O\left((\lambda')^{-1}k^{-1/4}\right)$. This rate can be improved to $O\left((\lambda')^{-1}k^{-1/2}\right)$ by using weighted step sizes that depend on λ' [Bub15, LSB12]. Thus, we pay a small price for the shrinking regularization routine and for not knowing the value of λ' in advance. We additionally provide faster convergence guarantees and improved convergence results for the proposed method on small datasets as compared to gradient descent applied to the logistic loss with fixed step sizes [SHN18].

3.1.1 Contributions

While several works analyze the convergence of various optimization methods to the maximal-margin solution for separable data [SHN18, NLG19], we are unaware of any works that provide explicit convergence rates for the fundamental subgradient descent method. Convergence of the subgradient method and stochastic subgradient method have been analyzed for non-smooth convex functions, however these works only provide convergence guarantees in the loss-function values and not the iterates, as, for general convex functions, the minimizer may not be finite and may not be unique [SZ13, Zha04]. In the context of solving the hard-margin SVM, the restriction to linearly separable data guarantees the existence of a minimizer and considering the maximal margin solution ensures uniqueness. Moreover, in the context of general convex functions, previous works often use the projected subgradient method and require knowledge of a bounded domain in which a minimizer exists [SZ13, Bub15]. For solving the hard-margin SVM via gradient descent, we show that such a projection is unnecessary.

Here, we provide explicit convergence guarantees for a homotopic subgradient method for optimizing the non-smooth SVM hinge loss. The proposed method uses decreasing regularization parameters and leads to the hard-margin SVM solution. We study the effects of optimization via this method on the generalization ability of the learned solutions through proved convergence rates to the hard-margin SVM solution in terms of L_2 error as well as difference in angle and margin from the true solution. We additionally show that these convergence rates to the hard-margin SVM solution outpace recent results such as gradient descent with fixed step sizes applied to the logistic loss [SHN18, NLG19]. We demonstrate the convergence of the proposed method on a synthetic dataset.

3.1.2 Organization

In Section 3.2, we introduce the specific problem setting, the notation that will be used throughout, and the proposed optimization scheme, Algorithm 1. Section 3.3 provides the main convergence results for Algorithm 1. An outline for the proof of the main convergence theorem, Theorem 3.3.1, is provided in Section 3.4, with additional details in Section 3.8.

We test convergence properties of Algorithm 1 for a simple synthetic dataset in Section 3.5. Section 3.6 provides additional implementation details for Algorithm 1 as well as possible modifications and extensions.

3.2 Problem Setup

We consider the binary classification problem with data $\{(\mathbf{x}_j, y_j) : j = 1, \ldots, n\}$, where $\mathbf{x}_j \in \mathbb{R}^d$ are the data points and $y_j \in \{-1, 1\}$ their labels. We aim to classify the data via a homogeneous linear SVM. Specifically, we wish to identify a weight vector $\mathbf{w}^*$ that satisfies

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \|\mathbf{w}\| \quad \text{subject to} \quad y_j \mathbf{x}_j^\top \mathbf{w} \geq 1 \,\forall\, j.$$

Throughout, we write $\|\cdot\| = \|\cdot\|_2$. We can equivalently find

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \|\mathbf{w}\| \text{ subject to } \mathcal{L}(\mathbf{w}) = 0, \tag{3.1}$$

where

$$\mathcal{L}(\mathbf{w}) := \frac{1}{n} \sum_{j=1}^{n} h(y_j \mathbf{x}_j^\top \mathbf{w}) \text{ and } h(u) := \max(0, 1-u).$$

The function $h(u)$ is commonly referred to as the *hinge loss*. We assume throughout that the data is linearly separable, i.e. there exists a vector $\mathbf{w}$ satisfying $\mathcal{L}(\mathbf{w}) = 0$ as is done in [SHN18, NLG19, WGC19, BGM18, NSS19, RZH04]. This assumption is common and necessary in order to discuss the margin of the approximated solutions. Minimizing the norm of the solution $\mathbf{w}$ to $\mathcal{L}(\mathbf{w}) = 0$ corresponds to maximizing the margin, that is maximizing the minimal distance between any data point and the separating hyperplane determined by $\mathbf{w}$. In this setting, the solution to Equation (3.1), $\mathbf{w}^*$, is often referred to as the *h*ard-margin SVM solution.

The constrained optimization problem in Equation (3.1) is the primal formulation of an SVM. While solving or approximating the corresponding dual SVM formulation is popular in practice, there are advantages to approximating the primal problem directly [Cha07]. Of particular interest for this work, considering the primal formulation allows for straightforward analysis of the effect of the optimization error on the margin and hyperplane angle.

As an alternative to solving Equation (3.1) directly, one often looks for a solution to an unconstrained, regularized version. Define the functional:

$$F_\lambda(\mathbf{w}) := \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{j=1}^{n} h(y_j \mathbf{x}_j^\top \mathbf{w}). \tag{3.2}$$

For $\lambda > 0$, F_λ is strongly convex with strong convexity parameter λ. We will use ∂F to denote the subgradient of F. The gradient of $F_\lambda(\mathbf{w})$ exists as long as $y_j \mathbf{x}_j^\top \mathbf{w} \neq 1$ for all j and is given by

$$\partial F_\lambda(\mathbf{w}) = \lambda \mathbf{w} - \frac{1}{n} \sum_{j: y_j \mathbf{x}_j^\top \mathbf{w} < 1} y_j \mathbf{x}_j. \tag{3.3}$$

When $y_j \mathbf{x}_j^\top \mathbf{w} = 1$ for some j, the subgradient set $\partial F_\lambda(\mathbf{w})$ contains the point

$$\lambda \mathbf{w} - \frac{1}{n} \sum_{j: y_j \mathbf{x}_j^\top \mathbf{w} < 1} y_j \mathbf{x}_j \in \partial F_\lambda(\mathbf{w}).$$

When the gradient does not exist, we will abuse notation and use Equation (3.3) in the subgradient method update of Equation (3.5).

Let

$$\mathbf{w}_\lambda^* := \operatorname*{argmin}_{\mathbf{w}} F_\lambda(\mathbf{w}). \tag{3.4}$$

We will refer to $\mathbf{w}_\lambda^*$ as the solution to the regularized subproblem of minimizing Equation (3.2). A larger regularization parameter λ encourages a solution $\mathbf{w}_\lambda^*$ with smaller norm at the cost of having some points lie within the margin. For linearly separable data and as λ approaches 0, the regularized solutions $\mathbf{w}_\lambda^*$ converge to the unregularized solution, $\mathbf{w}^*$. Let $\lambda' > 0$ be such that $\mathbf{w}_\lambda^* = \mathbf{w}^*$ for all $\lambda \leq \lambda'$. Such a λ' is guaranteed to exist for linearly separable data [RZH04, HRT04]. This fact suggests solving Equation (3.2) by using the subgradient method for a sufficiently small value of λ. Of course, the value of λ' will typically be unknown.

We use the following assumption and definition of λ' throughout.

Assumption 3.2.1. *The data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ with labels $y_1, \ldots, y_n \in \{-1, 1\}$ are linearly separable, i.e. there exists $\mathbf{w}$ such that for all i, $y_i \mathbf{w}^\top \mathbf{x}_i > 0$. Let $\mathbf{w}^*$ be the hard-margin SVM (i.e. $\mathbf{w}^*$ solves Equation (3.1)) and λ' be such that for all $\lambda \leq \lambda'$, $\mathbf{w}_\lambda^* = \operatorname{argmin} F_\lambda = \mathbf{w}^*$.*

While in practice, one may be satisfied with the solution $\mathbf{w}_\lambda^*$ for λ sufficiently small, we are interested in the convergence to the true hard-margin SVM given by $\mathbf{w}^*$. Thus, we instead propose to use a "homotopic" variant of the subgradient method that iteratively approximates the solution to Equation (3.2) while the regularization parameter λ and accompanying step size η of the subgradient method in Equation (3.5) decay at prescribed rates. Incorporating a piecewise constant decaying step size is commonly used for large-scale minimization problems, especially when using stochastic gradient descent variants [BCN18].

Recall the subgradient method given by the updates:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \partial F_\lambda(\mathbf{w}_k), \tag{3.5}$$

where $\mathbf{w}_k$ is the approximate solution at iteration k and η_k is a step size. For some number of outer iterations $s = 1, \dots, S$, we choose a regularization parameter $\lambda_s > 0$, a step size $\eta_s > 0$ and a number of inner iterations t_s. The regularization parameter λ_s and step size η_s are selected such that they decrease to 0 as s increases. Let $\overline{\mathbf{w}}_{s-1}$ be the current estimate of $\mathbf{w}^*$. We then perform t_s subgradient updates applied to the loss function F_{λ_s} with initial iterate $\overline{\mathbf{w}}_{s-1}$ and step size η_s. The next estimate, $\overline{\mathbf{w}}_s$, is given by the average of the t_s subgradient iterates. This process is detailed in Algorithm 1. For specific choices of λ_s, η_s and t_s, Algorithm 1 converges to the hard-margin SVM solution $\mathbf{w}^*$. Convergence guarantees are detailed in Theorem 3.3.1.

While the strongly convex functions F_λ are not globally Lipschitz, they are Lipschitz functions on bounded domains. Using a projected subgradient method in which iterates are projected onto a bounded domain is a natural strategy for restricting the domain of the iterates. A projection is unnecessary in this setting, however, as the regularization parameter $\lambda > 0$ naturally promotes solutions of smaller norm. In fact, the iterates produced by the subgradient method in Algorithm 1 remain bounded in norm with a bound that depends on the current regularization parameter λ.

Lemma 3.2.2. *Fix a regularization parameter $\lambda > 0$ and step size $\eta > 0$ such that $\eta\lambda < 1$. Define*

$$B_\lambda := \frac{\sum_{j=1}^n \|\mathbf{x}_j\|}{\lambda n}. \tag{3.6}$$

Algorithm 1 Homotopic subgradient method

```
Input: data $\{\mathbf{x}_j\}$, labels $\{y_j\}$, maximum outer iterations $S$, parameter for initial inner iterations $s_0 > 2$, parameters $1 > p > 0$ and $r > 2p$
Define $\epsilon_0 = \frac{\log(s_0)-\log(s_0-1)}{\log(s_0)}$, $\alpha = \min\left\{\frac{r-2p}{2(1+\epsilon_0)}, 1-p\right\}$, and $C = \max\left\{4, \frac{1}{2}s_0^p(s_0-1)^\alpha\right\}$
Initialize $\overline{\mathbf{w}}_0 = \mathbf{0}$
for $s = 0, 1, \ldots, S-1$ do
    $\lambda_s = (s_0+s)^{-p}$, $t_s = (s_0+s)^r$, and $\eta_s = \frac{C(s_0+s-1)^{-\alpha}}{\sqrt{t_s}}$
    $\mathbf{w}_0 = \overline{\mathbf{w}}_s$
    for $i = 1, 2, \ldots, t_s$ do
        $\mathbf{w}_i = (1-\lambda_s\eta_s)\mathbf{w}_{i-1} + \frac{\eta_s}{n}\sum_{j:y_j\mathbf{x}_j^\top\mathbf{w}_{i-1}\leq 1} y_j\mathbf{x}_j$
    end for
    $\overline{\mathbf{w}}_{s+1} = \frac{1}{t_s}\sum_{i=1}^{t_s}\mathbf{w}_i$
end for
Output $\overline{\mathbf{w}}_S$
```

If the initial iterate $\mathbf{w}_0$ *is such that* $\|\mathbf{w}_0\| \leq B_\lambda$*, then each iterate* $\mathbf{w}_k$ *produced by the subgradient method of Equation* (3.3) *applied to the function* F_λ *of Equation* (3.2) *has* $\|\mathbf{w}_k\| \leq B_\lambda$*. Additionally,* $\|\mathbf{w}^*\| \leq B_\lambda$*.*

In summary, if the initial iterate $\mathbf{w}_0$ is such that $\|\mathbf{w}_0\| \leq B_\lambda$, then the iterates produced by the subgradient method applied to F_λ will also have norm less than or equal to B_λ.

Remark. Using Lemma 3.2.2, one can show that the functionals F_λ are Lipschitz over the domain of iterates produced by Algorithm 1. Specifically, the constant

$$L := \frac{2}{n}\sum_{j=1}^{n}\|\mathbf{x}_j\| \tag{3.7}$$

bounds the Lipschitz constants of each function F_λ restricted to the ball centered at the origin with radius B_λ. Lemma 3.2.2 guarantees that the iterates produced when applying the subgradient method to F_λ and for sufficiently small initial iterate remain with this domain. Note that the bound on the Lipschitz constants L is independent of the regularization parameter λ.

3.3 Main Results

We now provide explicit rates of convergence to the hard-margin SVM solution for Algorithm 1. We provide convergence rates in terms of the L_2 error, difference in angle, and difference in margin between the approximation $\overline{\mathbf{w}}_S$ and the true hard-margin solution. The convergence results are stated in terms of k, the total number of subgradient updates required. Recall that the approximations $\overline{\mathbf{w}}_s$ are only updated at increments of t_s subgradient updates. Let $\mathbf{z}_k = \overline{\mathbf{w}}_s$, so that $\mathbf{z}_k$ is the approximation after $k = \sum_{i=1}^{s} t_i$ subgradient calculations.

Theorem 3.3.1 provides a convergence guarantee for the L_2 error of the iterates produced by Algorithm 1. This result will be used to additionally derive convergence guarantees for the angle and margin of the solution in Lemma 3.3.4. The parameter p determines the rate of decay of the regularization λ_s and the parameter r determines the number of steps t_s used at each fixed level of regularization. The constant L is as defined in Equation (3.7) and is an upper bound on the Lipschitz constants of the functions F_λ restricted to the domain of the iterates produced by the subgradient method applied to F_λ (Lemma 3.2.2).

Theorem 3.3.1. *Consider Algorithm 1 with parameters* r *and* p *such that* $0 < p < 1$ *and* $r > 2p$*. Choose an initial number of inner iterations* $s_0^r \in \mathbb{N}$ *with* $s_0 > 2$*. Let* $L = 2\frac{\sum_{j=1}^{n}\|\mathbf{x}_j\|}{n}$ *as defined in Equation* (3.7)*. Define*

$$C = \max\left\{4, \tfrac{1}{2}s_0^p(s_0-1)^\alpha\right\} \quad \textit{and} \quad \alpha = \min\left(\frac{r-2p}{2(1+\epsilon_0)}, 1-p\right),$$

with $\epsilon_0 = \frac{\log(s_0)-\log(s_0-1)}{\log(s_0)}$*. Let* $\mathbf{z}_k$ *be the average of the* t_s *subgradient descent updates calculated to minimize the function* F_{λ_s} *with step size* $\eta_s = \frac{C(s_0+s-1)^{-\alpha}}{\sqrt{t_s}}$*, where* k *is the total number of subgradient descent updates calculated. Then for data and* λ' *satisfying Assumption 3.2.1,*

$$\|\mathbf{z}_k - \mathbf{w}^*\| \leq CL\left((r+1)k\right)^{\frac{-\alpha(1-\epsilon_0)}{r+1}} + \frac{L}{2(\lambda')^2}\left((r+1)k\right)^{\frac{-p}{r+1}}. \tag{3.8}$$

Let $c = \min\left(\frac{\alpha(1-\epsilon_0)}{r+1}, \frac{p}{r+1}\right)$*. Then*

$$\|\mathbf{z}_k - \mathbf{w}^*\| \leq \left(C + \frac{1}{2(\lambda')^2}\right)L(r+1)^{-c}k^{-c}.$$

An outline for a proof of Theorem 3.3.1 can be found in Section 3.4 with additional details in Section 3.8. Note that, for small ϵ_0, the two terms in the bound of Equation (3.8) will decrease at approximately the same rate if $r = 2$ and $p = 1/2$. Corollary 3.3.2 gives a simpler, explicit rate of convergence by making this specification and setting $s_0 = 10$.

Corollary 3.3.2. *Consider Algorithm 1 with parameters $r = 2$, $p = 1/2$ and an initial number of inner iterations $s_0^r = s_0^2 \in \mathbb{N}$ with $s_0 > 2$. Let $L = 2\frac{\sum_{j=1}^n \|\mathbf{x}_j\|}{n}$. Define*

$$C = \max\left\{4, \tfrac{1}{2}s_0^p(s_0-1)^\alpha\right\} \quad \textit{and} \quad \alpha = \min\left(\frac{r-2p}{2(1+\epsilon_0)}, 1-p\right),$$

with $\epsilon_0 = \frac{\log(s_0)-\log(s_0-1)}{\log(s_0)}$. Let $\mathbf{z}_k$ be the average of the t_s subgradient descent updates calculated for F_{λ_s} with step size $\eta_s = \frac{C(s_0+s-1)^{-\alpha}}{\sqrt{t_s}}$, where k is the total number of subgradient descent updates calculated. Then for data and λ' satisfying Assumption 3.2.1,

$$\|\mathbf{z}_k - \mathbf{w}^*\| \le CL\,(3k)^{\frac{-1}{6}\frac{(1-\epsilon_0)}{(1+\epsilon_0)}} + \frac{L\,(3k)^{-1/6}}{2(\lambda')^2}.$$

Choosing $s_0 = 10$, we have $\epsilon_0 < 0.046$, $C < 4.9$ and arrive at the convergence rate

$$\|\mathbf{z}_k - \mathbf{w}^*\| \le 4.17Lk^{\frac{-0.913}{6}} + \frac{0.42Lk^{-1/6}}{(\lambda')^2}.$$

At least theoretically, sending $s_0 \to \infty$ leads to the best convergence rate guarantee. In fact, the convergence rate provided by Theorem 3.3.1 can be made arbitrarily close to $O\left(k^{-1/6}\right)$ by choosing $r = 2$, $p = 1/2$, and s_0 sufficiently large. As we will see in Section 3.5, using s_0 extremely large becomes impractical as the number of iterations for each fixed-λ subproblem becomes extremely large.

For strongly-convex, Lipschitz functions with strong-convexity parameter λ, one can achieve convergence in $\|\mathbf{w} - \mathbf{w}^*\|$ at a rate of $O\left(\lambda^{-1}k^{-1/4}\right)$, using projected averaged gradient descent with fixed step sizes (Theorem 3.2 [Bub14]). Using weighted step sizes, and knowledge of the strong convexity parameter, this rate can be improved to $O\left(\lambda^{-1}k^{-1/2}\right)$ (Theorem 3.9 [Bub14], originally from [LSB12]). A challenge of solving for the hard-margin SVM is that we do not optimize a strongly convex function. While one could fix a regularization parameter λ leading to a strongly convex function, there is no guarantee that the minimizer of this function F_λ will correspond to the true solution $\mathbf{w}^*$. Since the convergence rate of

Algorithm 1 can be made arbitrarily close to $O\left((\lambda')^{-2}k^{-1/6}\right)$ we lose very little, only a factor of $O\left((\lambda')^{-1}k^{1/12+\delta}\right)$ compared to the convergence rate of projected averaged gradient descent with fixed step sizes, for not knowing λ' in advance and instead incorporating decreasing explicit regularization.

Additionally, in designing Algorithm 1, we aimed for a simple algorithm as opposed to optimizing all possible parameters. One could possibly improve on the rates given here by further optimizing these parameters.

3.3.1 Convergence rates for angle and margin gaps

The convergence rate in Theorem 3.3.1 can be used to derive rates of convergence to the angle and margin of the optimal separating hyperplane $\mathbf{w}^*$.

Definition 3.3.3. For the hard margin SVM solution $\mathbf{w}^*$ and a vector $\mathbf{w}$, define

$$\textit{angle gap} := 1 - \frac{\mathbf{w}^\top \mathbf{w}^*}{\|\mathbf{w}\|\|\mathbf{w}^*\|}$$

and

$$\textit{margin gap} := \frac{1}{\|\mathbf{w}^*\|} - \min_i \frac{y_i \mathbf{x}_i^\top \mathbf{w}}{\|\mathbf{w}\|}.$$

While it is natural to consider the L_2 error of the derived solution, the angle between the true and derived solutions as well as the difference in the size of the margins give a more intuitive interpretation of the effect of that error. For example, an approximate solution $\mathbf{w}$ that is off by a constant factor, that is $\mathbf{w} = c\mathbf{w}^*$, will have an angle gap of zero and non-zero margin gap if $c \neq 1$. If an approximate solution $\mathbf{w}$ has a nonzero angle gap, but negligible margin gap, this suggests that the derived solution $\mathbf{w}$ still separates the data reasonably well.

Convergence rates of Algorithm 1 in terms of the angle and margin gaps are stated in Lemma 3.3.4 and compared to other recently obtained convergence rates in Table 3.1. The rates of convergence in these metrics can be derived from Theorem 3.3.1. These arguments are included in Section 3.8.

	Algorithm 1	[SHN18]
Angle gap	$O\left(k^{-1/3+2\delta}\right)$	$O\left(\left(\frac{\log\log(k)}{\log(k)}\right)^2\right)$
Margin gap	$O\left(k^{-1/6+\delta}\right)$	$O\left(\frac{1}{\log(k)}\right)$

Table 3.1: Comparison of convergence rates for Algorithm 1 with those of [SHN18] for gradient descent with fixed step sizes applied to the logistic loss.

Lemma 3.3.4. *Let*

$$c = \min\left(\frac{(r-2p)(1-\epsilon_0)}{2(r+1)(1+\epsilon_0)}, \frac{(1-p)(1+\epsilon_0)}{r+1}, \frac{p}{r+1}\right),$$

where p, r, s_0, and ϵ_0 are as given in Theorem 3.3.1 so that c is the exponent in the convergence rate of Theorem 3.3.1. Let δ be such that $c = 1/6 - \delta$. The value of δ is positive and can be made arbitrarily close to 0 by choosing s_0 sufficiently large and setting $p = 1/2$ and $r = 2$. Then for the angle gap,

$$1 - \frac{\mathbf{w}_k^\top \mathbf{w}^*}{\|\mathbf{w}_k\|\|\mathbf{w}^*\|} = O\left(k^{-1/3+2\delta}\right).$$

For the margin gap,

$$\frac{1}{\|\mathbf{w}^*\|} - \min_i \frac{y_i \mathbf{x}_i^\top \mathbf{w}_k}{\|\mathbf{w}_k\|} = O\left(k^{-1/6+\delta}\right).$$

The convergence guarantees for the angle and margin gaps for Algorithm 1 are significantly faster than those given in Soudry et al [SHN18] for gradient descent with fixed step sizes applied to the logistic loss (see Table 3.1). Nacson et al [NLG19] demonstrate that using aggressive adaptive step sizes for gradient descent applied to the logistic loss leads to a faster convergence rate of $O\left(\frac{\log(t)}{\sqrt{t}}\right)$. While the convergence guarantees for Algorithm 1 are slower, as $c \leq 1/6$, in this paper, we are interested in analyzing convergence guarantees for gradient descent applied to the non-smooth hinge loss.

3.4 Proof of Theorem 3.3.1

We prove Theorem 3.3.1 through a series of lemmas, which are stated in Subsection 3.4.1 and whose proofs are contained in Section 3.8. The proof of Theorem 3.3.1 is contained in Subsection 3.4.2.

We briefly summarize each of the lemmas for convenience. Lemma 3.4.1 provides a modified convergence guarantee for the averaged subgradient method applied to the functions F_λ. Lemma 3.4.2 bounds the distance between minimizers of F_λ for different regularization parameters λ. This result allows for the incorporation of the decreasing regularization in Algorithm 1. Lemma 3.4.3 makes use of Lemma 3.4.1 and Lemma 3.4.2 to bound the initial error $\|\overline{\mathbf{w}}_s - \mathbf{w}^*_\lambda\|$ of each regularized subproblem as given in Equation (3.4).

3.4.1 Useful lemmas

Lemma 3.4.1 is a modified version of a standard convergence analysis of the averaged subgradient method for convex Lipschitz functions (Theorem 3.2 of [Bub15]). This result bounds the distance between the average of the subgradient descent iterates $\overline{\mathbf{w}}$ and the minimizer $\mathbf{w}^*_\lambda$ of the functional F_λ for a fixed regularization parameter λ.

Lemma 3.4.1. *Let*

$$F_\lambda(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{j=1}^{n}\max(0, 1 - y_j\mathbf{x}_j^\top \mathbf{w})$$

and $L = 2\frac{\sum_{j=1}^n \|\mathbf{x}_j\|}{n}$. *Let the initial iterate* $\mathbf{w}_0$ *be such that* $\|\mathbf{w}_0\| \leq \frac{L}{2\lambda}$ *and let* $\mathbf{w}^*_\lambda$ *minimize* F_λ. *Suppose* $\|\mathbf{w}_0 - \mathbf{w}^*_\lambda\| \leq R$, *so that* $\mathbf{w}^*_\lambda$ *is contained in a ball of radius* R *and center* $\mathbf{w}_0$. *Let* $\overline{\mathbf{w}} = \frac{1}{t}\sum_{s=1}^{t}\mathbf{w}_s$ *be the average of* t *subgradient method iterates with initial iterate* $\mathbf{w}_0$ *and step size* $\eta = \frac{R}{L\sqrt{t}}$. *Then*

$$0 \leq F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}^*_\lambda) \leq \frac{RL}{\sqrt{t}} - \frac{\lambda}{2}\|\overline{\mathbf{w}} - \mathbf{w}^*_\lambda\|^2.$$

Note that Lemma 3.4.1 also guarantees that

$$\|\overline{\mathbf{w}} - \mathbf{w}^*_\lambda\|^2 \leq \frac{2RL}{\lambda\sqrt{t}}.$$

The next lemma bounds the distance between the minimizers $\mathbf{w}_\lambda^*$ and $\mathbf{w}_{\widetilde{\lambda}}^*$ of the functions F_λ and $F_{\widetilde{\lambda}}$ and shows that distance from $\mathbf{w}_\lambda^*$ to the true hard-margin solution $\mathbf{w}^*$, $\|\mathbf{w}_\lambda^* - \mathbf{w}^*\|$, is proportional to the regularization parameter λ.

Lemma 3.4.2. *Let $\mathbf{w}_\lambda^*$ minimize F_λ as given in Equation* (3.2) *and let $\mathbf{w}^*$ solve Equation* (3.1). *Let $\lambda' > 0$ be such that $\mathbf{w}_\lambda^* = \mathbf{w}^*$ for all $\lambda \leq \lambda'$ and $L = 2\frac{\sum_{j=1}^n \|\mathbf{x}_j\|}{n}$. For $\lambda, \widetilde{\lambda} \geq 0$ and data satisfying Assumption 3.2.1, we have*

$$\|\mathbf{w}_\lambda^* - \mathbf{w}_{\widetilde{\lambda}}^*\| \leq \frac{L}{2}\left|\frac{1}{\lambda} - \frac{1}{\widetilde{\lambda}}\right| \tag{3.9}$$

and

$$\|\mathbf{w}_\lambda^* - \mathbf{w}^*\| \leq \frac{L\lambda}{2\,(\lambda')^2}. \tag{3.10}$$

The final lemma bounds the initial error at each fixed level of regularization for the subgradient updates produced when minimizing F_{λ_s}. In particular, it specifies a bound shrinking in s on the distance between the initial iterate $\overline{\mathbf{w}}_s$ and the minimizer $\mathbf{w}_{\lambda_s}^*$ of the function F_{λ_s}. The fact that the initial error for each regularized subproblem goes to zero is crucial for proving the convergence of Algorithm 1 to the hard margin SVM solution.

Lemma 3.4.3. *Let $L = 2\frac{\sum_{j=1}^n \|\mathbf{x}_j\|}{n}$ and $R_0 = \frac{L}{2\lambda_0}$. For $s_0 \in \mathbb{N}$ with $s_0 > 2$, $p \in (0,1)$, and $r > 2p$, let $\lambda_s = (s_0 + s)^{-p}$ and $t_s = (s_0 + s)^r$. Let*

$$R_s = CL(s_0 + s - 1)^{-\alpha} \quad \textit{for} \quad 0 \leq \alpha \leq \min\left(\frac{r - 2p}{2(1 + \epsilon_0)}, 1 - p\right),$$

with

$$C = \max\left\{4, \frac{1}{2\lambda_0}(s_0 - 1)^\alpha\right\} \ \textit{and}\ \epsilon_0 = \frac{\log(s_0) - \log(s_0 - 1)}{\log(s_0)}.$$

Let $\eta_s = \frac{R_s}{L\sqrt{t_s}}$. Then for the averaged subgradient iterates $\overline{\mathbf{w}}_s$ of Algorithm 1,

$$\|\overline{\mathbf{w}}_s - \mathbf{w}_{\lambda_s}^*\| \leq R_s.$$

Based on Lemma 3.4.3, for $r > 2p$ and $p < 1$ the radii R_s shrink to 0 as s increases.

3.4.2 Proof of Theorem 3.3.1.

We now prove Theorem 3.3.1 using the above lemmas.

Proof. We use the triangle inequality to bound the error as

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*\| \leq \|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| + \|\mathbf{w}^*_{\lambda_s} - \mathbf{w}^*\|. \tag{3.11}$$

We then bound the terms $\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\|$ and $\|\mathbf{w}^*_{\lambda_s} - \mathbf{w}^*\|$ using the lemmas of Subsection 3.4.1.

Let $L = 2\frac{\sum_{j=1}^{n}\|\mathbf{x}_j\|}{n}$ and choose $s_0 \in \mathbb{N}$ with $s_0 > 2$. Let $\lambda_s = (s_0+s)^{-p}$ and $t_s = (s_0+s)^r$. Let

$$R_s = CL(s_0+s-1)^{-\alpha}, \text{ for } \alpha = \min\left(\frac{r-2p}{2(1+\epsilon_0)}, 1-p\right),$$

with

$$C = \max\left\{4, \tfrac{1}{2}s_0^p(s_0-1)^\alpha\right\} \text{ and } \epsilon_0 = \frac{\log(s_0)-\log(s_0-1)}{\log(s_0)}.$$

Let $\eta_s = \frac{R_s}{L\sqrt{t_s}}$. By Lemma 3.4.3, considering the first term in the bound of Equation (3.11),

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq R_s = CL(s_0+s-1)^{-\alpha}.$$

Changing the base,

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq CL(s_0+s)^{-\alpha(1-\epsilon_0)}.$$

We now bound the second term of the bound in Equation (3.11). Let $\lambda' > 0$ be such that $\mathbf{w}^*_\lambda = \mathbf{w}^*$ for all $\lambda \leq \lambda'$. By Lemma 3.4.2,

$$\|\mathbf{w}^*_{\lambda_s} - \mathbf{w}^*\| \leq \frac{L\lambda_s}{2(\lambda')^2} = \frac{L(s_0+s)^{-p}}{2(\lambda')^2}.$$

The total number of updates, k, used to calculate $\overline{\mathbf{w}}_s$ is bounded by

$$k = \sum_{i=0}^{s-1} t_i = \sum_{i=s_0}^{s_0+s-1} i^r \leq \int_{s_0+1}^{s+s_0} i^r = \frac{(s+s_0)^{r+1}}{r+1}.$$

Rearranging,

$$((r+1)k)^{\frac{1}{r+1}} \leq s_0 + s.$$

Writing the bounds in terms of the total number of updates, k,

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq CL(s_0+s)^{-\alpha(1-\epsilon_0)} \leq CL\left((r+1)k\right)^{\frac{-\alpha(1-\epsilon_0)}{r+1}}$$

and

$$\|\mathbf{w}^*_{\lambda_s} - \mathbf{w}^*\| \leq \frac{L\left((r+1)k\right)^{\frac{-p}{r+1}}}{2(\lambda')^2}.$$

Combining these,

$$\begin{aligned}\|\overline{\mathbf{w}}_s - \mathbf{w}^*\| &\leq \|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| + \|\mathbf{w}^*_{\lambda_s} - \mathbf{w}^*\| \\ &\leq CL\left((r+1)k\right)^{\frac{-\alpha(1-\epsilon)}{r+1}} + \frac{L\left((r+1)k\right)^{\frac{-p}{r+1}}}{2(\lambda')^2}.\end{aligned}$$

□

In order to optimize the convergence rate given in Theorem 3.3.1, we aim to choose parameters p and r such that

$$p, q = \operatorname*{argmax}_{p,q} \min\left\{\frac{(r-2p)(1-\epsilon_0)}{2(1+\epsilon_0)}, (1-p)(1-\epsilon_0), p\right\}.$$

For ϵ_0 small, $p = 1/2$ and $r = 2$ lead to a nearly optimal converge rate of

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*\| \leq 4L\left((r+1)k\right)^{-\frac{(1-\epsilon)}{6(1+\epsilon)}} + \frac{L\left((r+1)k\right)^{-1/6}}{2(\lambda')^2}.$$

The choices $p = \frac{1}{2}$ and $r = 2$ are considered in Corollary 3.3.2 and an explicit convergence rate is given under these conditions.

3.5 Experimental Results

We demonstrate the convergence of Algorithm 1 through several experiments on a simple synthetic dataset that is shown in Figure 3.1. The experiments aim to explore the differences between convergence in theory versus practice and are not intended to be exhaustive or demonstrate superior performance over existing methods. The data includes four support vectors which occur at $\pm(0.5, 1.5)$ and $\pm(1.5, 0.5)$. The hard-margin SVM solution is given by $\mathbf{w}^* = (0.5, 0.5)$. The maximal regularization parameter λ' such that $\mathbf{w}^*_\lambda = \mathbf{w}^*$ for all $\lambda \leq \lambda'$ is $\lambda' = 0.5$. We fix the parameters $p = 1/2$ and $r = 2$ as are considered in Corollary 3.3.2 and initialize $\mathbf{w}_0 = \mathbf{0}$.

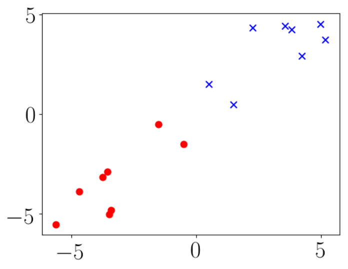


Figure 3.1: Synthetic data considered.

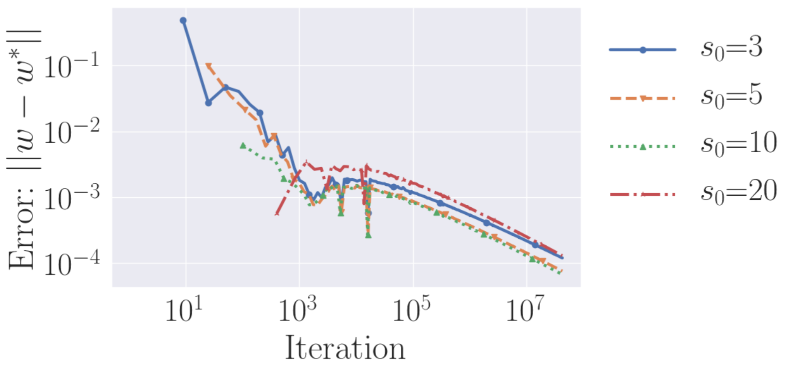


Figure 3.2: Performance of Algorithm 1 applied to data from Figure 3.1 with $p = 1/2$, $r = 2$ and varying s_0.

We measure convergence in terms of the L_2 error as well as the angle and margin gaps of Definition 3.3.3. Convergence results for Algorithm 1 with $p = 1/2$, $r = 2$ and varying s_0 are shown in Figure 3.2. In terms of the L_2 error, for a fixed number of iterations, there appears to be an optimal choice for the parameter s_0, as choosing $s_0 = 10$ performs better than $s_0 = 3, 5$ or 20.

We additionally compare the convergence of Algorithm 1 in terms of the angle gap and margin gap to gradient descent using fixed step sizes applied to the logistic loss. We use step sizes $\eta = \frac{1}{\sigma_{\max}(\mathbf{X})}$, where $\sigma_{\max}(\mathbf{X})$ is the largest singular value of the data matrix $\mathbf{X}$. As can be seen in Figure 3.3, we find significantly faster convergence via Algorithm 1 as compared to minimization of the logistic loss via gradient descent with fixed step sizes as considered in [SHN18, NLG19]. This result is unsurprising, as Algorithm 1 arrives at the SVM solution via controlled explicit regularization as opposed to only implicit regularization via gradient

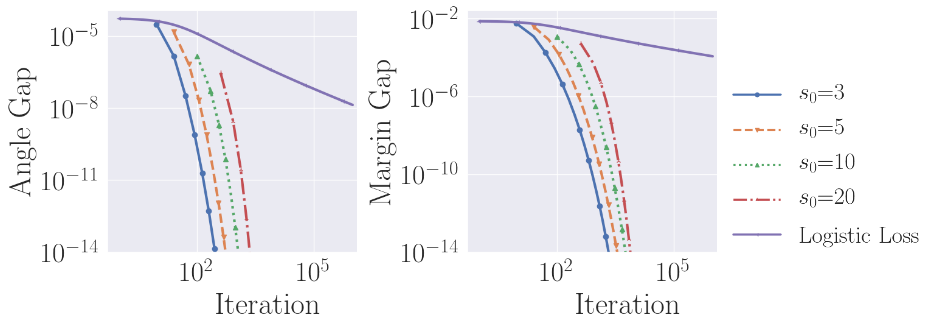


Figure 3.3: Performance of Algorithm 1 applied to data from Figure 3.1 in terms of the angle and margin gaps with $p = 1/2$, $r = 2$ and varying s_0. For comparison, we include gradient descent applied to the logistic loss as described in [SHN18] with step size $\eta = \frac{1}{\sigma_{\max}(\mathbf{X})}$, where $\sigma_{\max}(\mathbf{X})$ is the largest singular value of the data matrix $\mathbf{X}$.

descent.

We additionally consider the performance of Algorithm 1 applied to the data of Figure 3.1 with the y-values of the data multiplied by 20. This leads to a slightly more challenging problem with less symmetric data. The results are shown in Figure 3.4. We find that the convergence of Algorithm 1 is slightly slower in terms of L_2 error. The logistic loss converges significantly slower in terms of both the angle and margin gaps, whereas the effect on the convergence of Algorithm 1 appears to be minimal.

3.6 Implementation remarks

As presented, Algorithm 1 is highly adaptable for different loss functions and settings in which one would like to consider a range of regularization parameters or variable regularization. In this section, we present several potential modifications of interest, including adaptive or gradient based step sizes, amenability to using stochastic subgradients, and alternative updates.

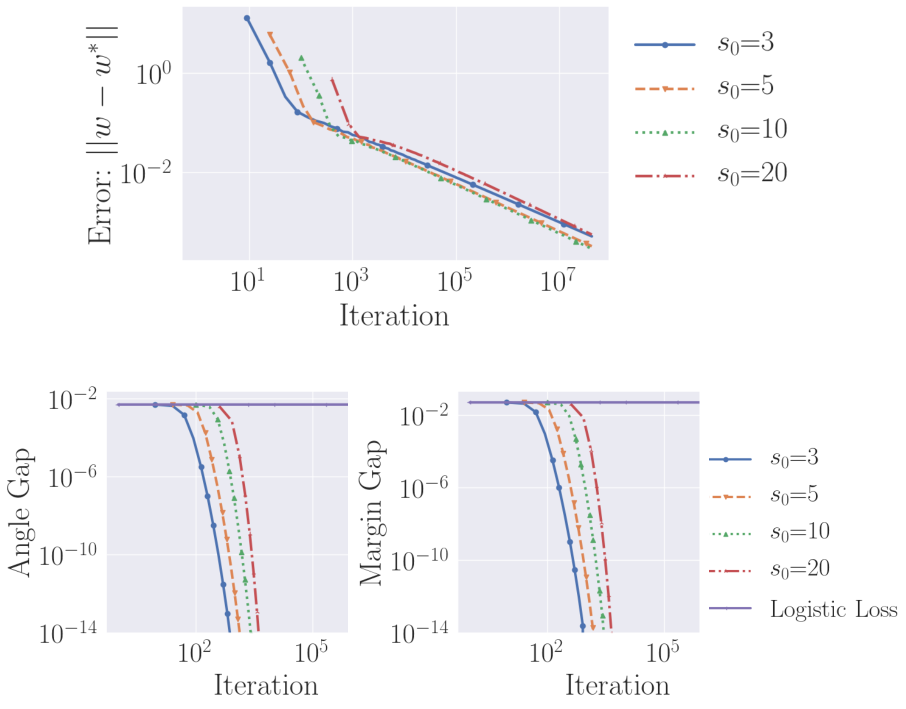


Figure 3.4: Performance of Algorithm 1 applied to the data of Figure 3.1, but with the values of all y-coordinates scaled by 20. The parameters $p = 1/2$, $r = 2$ and varying s_0 are used. For comparison, the angle gap and margin gap plots include gradient descent applied to the logistic loss as described in [SHN18] with step size $\eta = \frac{1}{\sigma_{\max}(\mathbf{X})}$, where $\sigma_{\max}(\mathbf{X})$ is the largest singular value of the data matrix $\mathbf{X}$.

3.6.1 Adaptive step sizes

When the regularization parameter, λ, or the norm of $\mathbf{w}$ are small and close to optimal, if an iterate violates one of the hinge loss constraints, this can increase the magnitude of the gradient of the loss F_λ significantly, leading to a relatively large jump in the next iterate followed by many smaller steps back toward the optimal solution of smaller norm. Using gradient descent with adaptive or loss-dependent step sizes can minimize the effects of these cycles. For example, we could adjust Algorithm 1 to use step sizes that are normalized by the magnitude of the subgradient,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \frac{\nabla F_\lambda(\mathbf{w}_k)}{\|\nabla F_\lambda(\mathbf{w}_k)\|}. \tag{3.12}$$

With this choice, the magnitude of the update is always η_k and is independent of the magnitude of the gradient of F_λ. Cursory experimental results suggest that using adaptive step sizes as in Equation (3.12), leads to slower convergence to the true solution initially and does not lead to improved convergence overall.

One could also potentially increase the convergence rate guarantees for Algorithm 1 by incorporating aggressive loss-dependent step sizes. In [NLG19], the authors show that when using Equation (3.12) with step sizes $\eta_k = \frac{1}{L(\mathbf{w}_k)}$, gradient descent applied to the logistic loss converges at the nearly optimal rate of $O(t^{-1/2}\log t)$. While this strategy provides a faster convergence rate, loss-dependent step sizes are less commonly used in practice as, in the stochastic setting, updating the loss at each iteration is often too expensive. The stochastic setting is discussed further in Subsection 3.6.3.

3.6.2 Regularization decay rate

In Algorithm 1, we consider regularization parameters that decay at a rate of $\lambda_s = O(s^{-p})$ for a constant $p > 0$. One might consider other choices for the decay rate of the regularization parameter λ. For example $\lambda_s = O\left(\frac{1}{\log(s)}\right)$ or $\lambda_s = O(c^s)$ for $c \in (0,1)$. Recall that in bounding the error $\|\overline{\mathbf{w}}_s - \mathbf{w}^*\|$ we use the decomposition

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*\| \leq \|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| + \|\mathbf{w}^*_{\lambda_s} - \mathbf{w}^*\|.$$

The first term converges more quickly when λ is large while the second term converges more quickly when λ is small. The decay rate of $\lambda_s = O(s^{-p})$ was chosen to balance the convergence of these terms.

3.6.3 Stochastic subgradients

Algorithm 1 can be naturally extended to the stochastic subgradient setting, in which one performs updates based on the subgradient of the loss with respect to only a subset of the data points. This is often necessary for large-scale optimization problems. Additionally, although piecewise-constant decaying step sizes are incorporated into Algorithm 1 to account for the introduced regularization, it is also often used in stochastic gradient descent in order to mitigate the effect of noise in the gradient approximation of each update [BCN18]. This commonality suggests that Algorithm 1 may be particularly suited for the stochastic setting.

3.6.4 Alternative updates

Lemma 3.4.1 is the only result that depends on the update given by the fixed-λ subproblem and, in particular, Theorem 3.3.1 applies to any update that satisfies $\|\overline{\mathbf{w}}_s - \mathbf{w}_\lambda^*\| \leq R_s$ for each $s = 1, \ldots, S$. Thus, as opposed to using the average of the iterates from each fixed λ subproblem, one could use alternative updates, such as

$$\widehat{\mathbf{w}}_s = \operatorname*{argmin}_{i=1,\ldots,t_s} F_{\lambda_s}(\mathbf{w}_i),$$

or the iterate that leads to the minimal loss for that subproblem. We refer to this update choice as the best-iterate update and investigate the effects of this choice in Figure 3.5.

We find that the best-iterate update typically leads to significantly faster convergence in terms of the L_2 error. Specifically, choosing the best iterate can alleviate the slow convergence caused by the slow decrease in step size. The convergence of the two strategies, using the averaged iterate and the best iterate, perform comparably in terms of the angle gap. Using the best iterate converges somewhat slower in terms of the margin gap.

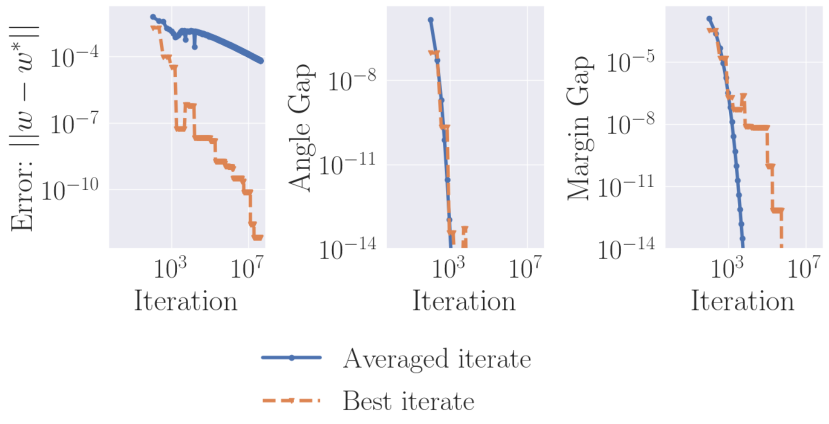


Figure 3.5: Performance of Algorithm 1 applied to data from Figure 3.1 using the averaged iterate versus the best iterate from each regularized subproblem. parameters $p = 1/2$, $r = 2$ and $s_0 = 10$ are used.

3.6.5 Incorporating a bias term

As in [RZH04, SHN18], we consider the case in which the maximal-margin separating hyperplane intersects the origin. One can allow for more general hyperplanes by learning a bias term b for the separating hyperplane. We propose the following method for approximating the bias term b

$$b = -\frac{(\min_{i:y_i=1} \mathbf{x}_i^\top \mathbf{w} + \max_{i:y_i=-1} \mathbf{x}_i^\top \mathbf{w})}{2}, \tag{3.13}$$

which is guaranteed to be close to the true max-margin bias b^* when $\|\mathbf{w} - \mathbf{w}^*\|$ is small. Specifically, one can verify that for the bias b as calculated in Equation (3.13) and b^* the true bias, we have

$$|b - b^*| \leq \max_i \|\mathbf{x}_i\| \|\mathbf{w} - \mathbf{w}^*\|.$$

Initial experiments with a non-trivial bias demonstrate convergence similar to the zero-bias case.

3.7 Conclusion

We have shown that, for linearly separable data, the subgradient method converges to the max-margin SVM solution when minimizing the unconstrained regularized SVM, Equation (3.2), with decreasing regularization parameters, λ. Under the conditions given in Theorem 3.3.1, this convergence can be guaranteed to be $O\left(k^{-1/6+\delta}\right)$ for any $\delta > 0$. We compare convergence rates in several metrics to those provided in [SHN18, NLG19]. In particular, the convergence rate guarantees for Algorithm 1 are faster than those of [SHN18, NLG19] for gradient descent with fixed step sizes. This restriction to fixed or piecewise constant step sizes is a practical choice, especial when working with large-scale optimization problems. We additionally demonstrate the convergence of Algorithm 1 on a simple synthetic dataset.

Although we specifically consider the hinge loss and SVMs, the results and analysis presented here could be extended to more general settings. For example, one could more generally consider settings in which one aims to solve

$$\mathbf{w}^* = \lim_{\lambda \to 0^+} \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2} g(\mathbf{w}) + f(\mathbf{w}),$$

where g is strongly convex and Lipschitz over bounded domains, f is convex and Lipschitz, and the regularization path,

$$\mathbf{w}^*_\lambda = \frac{\lambda}{2} g(\mathbf{w}) + f(\mathbf{w}),$$

is Lipschitz in λ.

3.8 Lemma Proofs

We now present proofs for the lemmas of Sections 3.2 to 3.4.

We first prove Lemma 3.2.2, which gives a bound on the norm of the iterates produced by the subgradient method applied to Equation (3.2).

Proof of Lemma 3.2.2

Proof. Consider the subgradient update for minimizing the function F_λ of Equation (3.2)

$$\mathbf{w}' = (1-\lambda\eta)\mathbf{w} + \frac{\eta}{n}\sum_{j:y_j\mathbf{x}_j^\top\mathbf{w}\le 1} y_j\mathbf{x}_j \tag{3.14}$$

with $\eta\lambda < 1$. Suppose that the iterate $\mathbf{w}$ satisfies $\|\mathbf{w}\| \le \frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\|$. We aim to show that $\mathbf{w}'$ given by the subgradient update also satisfies $\|\mathbf{w}\| \le \frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\|$. Taking the norm on both sides of Equation (3.14),

$$\begin{aligned}
\|\mathbf{w}'\| &= \left\|(1-\eta\lambda)\mathbf{w} + \frac{\eta}{n}\sum_{j:y_j\mathbf{x}_j^\top\mathbf{w}\le 1} y_j\mathbf{x}_j\right\| \\
&\le (1-\eta\lambda)\|\mathbf{w}\| + \frac{\eta}{n}\left\|\sum_{j:y_j\mathbf{x}_j^\top\mathbf{w}\le 1} y_j\mathbf{x}_j\right\| \\
&\le (1-\eta\lambda)\frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\| + \frac{\eta}{n}\sum_{j=1}^n \|\mathbf{x}_j\| \\
&= \frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\| - \frac{\eta}{n}\sum_{j=1}^n \|\mathbf{x}_j\| + \frac{\eta}{n}\sum_{j=1}^n \|\mathbf{x}_j\| \\
&= \frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\|.
\end{aligned}$$

Thus the norms of all iterates of the subgradient method applied to the function F_λ remain bounded by $\frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\|$ if the initial iterate has norm at most $\frac{1}{\lambda n}\sum_{j=1}^n \|\mathbf{x}_j\|$. The norm of the minimizer $\mathbf{w}_\lambda^*$ of F_λ must also satisfy the bound $\|\mathbf{w}_\lambda^*\| \le \frac{1}{\lambda n}\sum_j \|\mathbf{x}_j\|$ as $0 \in \partial F_\lambda(\mathbf{w}_\lambda^*)$ and so

$$\lambda\|\mathbf{w}_\lambda^*\| \le \frac{1}{n}\left\|\sum_{j:y_j\mathbf{x}_j^\top\mathbf{w}_\lambda^*\le 1} y_j\mathbf{x}_j\right\|.$$

□

Proof of Lemma 3.3.4

Lemma 3.3.4 uses Theorem 3.3.1 to derive bounds for the angle and margin gaps.

Proof. To derive a convergence rate for the angle gap, we use the decomposition

$$\begin{aligned}
\|\mathbf{w}_k - \mathbf{w}^*\|^2 &= \|\mathbf{w}_k\|^2 + \|\mathbf{w}^*\|^2 - 2\mathbf{w}_k^\top\mathbf{w}^* \\
&= (\|\mathbf{w}_k\| - \|\mathbf{w}^*\|)^2 + 2\|\mathbf{w}_k\|\|\mathbf{w}^*\| - 2\mathbf{w}_k^\top\mathbf{w}^*.
\end{aligned}$$

Dividing by $2\|\mathbf{w}_k\|\|\mathbf{w}^*\|$,

$$\begin{aligned}1-\frac{\mathbf{w}_k^\top \mathbf{w}^*}{\|\mathbf{w}_k\|\|\mathbf{w}^*\|} &= \frac{\|\mathbf{w}_k-\mathbf{w}^*\|^2-(\|\mathbf{w}_k\|-\|\mathbf{w}^*\|)^2}{2\|\mathbf{w}_k\|\|\mathbf{w}^*\|}\\ &\leq \frac{\|\mathbf{w}_k-\mathbf{w}^*\|^2}{2\|\mathbf{w}_k\|\|\mathbf{w}^*\|}.\end{aligned}$$

Since $\|\mathbf{w}^*\|$ is necessarily bounded away from 0 since $y_i\mathbf{x}_i^\top\mathbf{w}^* \geq 1$ for all i. We can bound $\|\mathbf{w}_k\|$ away from 0 for t large using the convergence of $\mathbf{w}_k$ to $\mathbf{w}^*$ guaranteed by Theorem 3.3.1. Let

$$c = \min\left(\frac{(r-2p)(1-\epsilon_0)}{2(r+1)(1+\epsilon_0)}, \frac{(1-p)(1+\epsilon_0)}{r+1}, \frac{p}{r+1}\right),$$

be the exponent in the convergence rate of $\|\mathbf{w}-\mathbf{w}^*\|$ and p, r, and ϵ_0 be defined as in Theorem 3.3.1. Since

$$(\|\mathbf{w}_k\|-\|\mathbf{w}^*\|)^2 \leq \|\mathbf{w}_k-\mathbf{w}^*\|^2 \leq Ak^{-2c}$$

for constants $A, c > 0$ by Theorem 3.3.1, then $\|\mathbf{w}_k\| \geq \|\mathbf{w}^*\| - Ak^{-c}$. Thus for k sufficiently large, we can bound $\|\mathbf{w}\|$ away from 0 and have

$$1-\frac{\mathbf{w}_k^\top \mathbf{w}^*}{\|\mathbf{w}_k\|\|\mathbf{w}^*\|} = O\left(k^{-2c}\right). \tag{3.15}$$

We now consider the margin bound. Let $j = \operatorname{argmin}_{i=1,\dots n} \frac{y_i\mathbf{x}_i^\top\mathbf{w}_k}{\|\mathbf{w}_k\|}$. Since $y_i\mathbf{x}_i^\top\mathbf{w}^* \geq 1$ for all $i = 1,\dots,n$, we have that

$$\begin{aligned}0 \leq \frac{1}{\|\mathbf{w}^*\|} - \frac{y_j\mathbf{x}_j^\top\mathbf{w}_k}{\|\mathbf{w}_k\|} &\leq \frac{y_j\mathbf{x}_j^\top\mathbf{w}^*}{\|\mathbf{w}^*\|} - \frac{y_j\mathbf{x}_j^\top\mathbf{w}_k}{\|\mathbf{w}_k\|}\\ = y_j\mathbf{x}_j^\top\left(\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}-\frac{\mathbf{w}_k}{\|\mathbf{w}_k\|}\right) &\leq \|\mathbf{x}_j\|\left\|\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}-\frac{\mathbf{w}_k}{\|\mathbf{w}_k\|}\right\|.\end{aligned}$$

Note that

$$\left\|\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}-\frac{\mathbf{w}_k}{\|\mathbf{w}_k\|}\right\|^2 = 2\left(1-\frac{\mathbf{w}_k^\top\mathbf{w}^*}{\|\mathbf{w}_k\|\|\mathbf{w}^*\|}\right).$$

Assuming the data is finite and linearly separable, by Equation (3.15) we then have

$$\frac{1}{\|\mathbf{w}^*\|} - \min_i \frac{y_i\mathbf{x}_i^\top\mathbf{w}_k}{\|\mathbf{w}_k\|} = O\left(k^{-c}\right).$$

□

Proof of Lemma 3.4.1

Lemma 3.4.1 provides a modified convergence guarantee for the averaged subgradient method applied to the functions F_λ [Bub14].

Proof. Let F_λ be a strongly convex function with strong convexity parameter λ and Lipschitz constant L on the bounded domain considered. Let $\mathbf{w}_0$ be an initial iterate and $\mathbf{w}_\lambda^*$ be the minimizer of F_λ. Suppose $\|\mathbf{w}_0 - \mathbf{w}_\lambda^*\| \le R$, so that $\mathbf{w}_\lambda^*$ is contained in a ball of radius R and center $\mathbf{w}_0$. Let $\overline{\mathbf{w}} = \frac{1}{t}\sum_{i=1}^t \mathbf{w}_i$ be the average of t subgradient descent iterates with initial iterate $\mathbf{w}_0$ and step size $\eta = \frac{R}{L\sqrt{t}}$. We aim to show that

$$0 \le F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}_\lambda^*) \le \frac{RL}{\sqrt{t}} - \frac{\lambda}{2}\|\overline{\mathbf{w}} - \mathbf{w}_\lambda^*\|^2.$$

The following proof relies heavily on Theorem 3.2 of [Bub14] (See also [Bub15]).

Since $\mathbf{w}_\lambda^*$ is the minimizer of F_λ, the inequality

$$F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}_\lambda^*) \ge 0$$

is immediate. Let $g(\mathbf{w}) = F_\lambda(\mathbf{w}) - \frac{\lambda}{2}||\mathbf{w}||^2$. Since $g(\mathbf{w})$ is convex,

$$g(\overline{\mathbf{w}}) \le \frac{1}{t}\sum_{i=1}^t g(\mathbf{w}_i)$$

and thus

$$F_\lambda(\overline{\mathbf{w}}) - \frac{\lambda}{2}||\overline{\mathbf{w}}||^2 \le \frac{1}{t}\sum_{i=1}^t \left(F_\lambda(\mathbf{w}_i) - \frac{\lambda}{2}||\mathbf{w}_i||^2 \right).$$

Reorganizing and subtracting $F_\lambda(\mathbf{w}_\lambda^*)$,

$$\begin{aligned} &F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}_\lambda^*) \\ &\le \frac{1}{t}\sum_{i=1}^t \left(F_\lambda(\mathbf{w}_i) - F_\lambda(\mathbf{w}_\lambda^*) - \frac{\lambda}{2}\left(||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2\right) \right). \end{aligned} \tag{3.16}$$

Using the strong convexity of F_λ and the proof of Theorem 3.2 of [Bub14],

$$\begin{aligned} &F_\lambda(\mathbf{w}_i) - F_\lambda(\mathbf{w}_\lambda^*) \\ &\le \partial F_\lambda(\mathbf{w}_i)^\top(\mathbf{w}_i - \mathbf{w}_\lambda^*) - \frac{\lambda}{2}\|\mathbf{w}_i - \mathbf{w}_\lambda^*\|^2 \\ &= \frac{1}{2\eta}\left(\|\mathbf{w}_i - \mathbf{w}^*\|^2 - \|\mathbf{w}_{i+1} - \mathbf{w}^*\|^2\right) + \frac{\eta}{2}\|\partial F_\lambda(\mathbf{w}_i)\|^2 - \frac{\lambda}{2}\|\mathbf{w}_i - \mathbf{w}_\lambda^*\|^2 \\ &\le \frac{1}{2\eta}\left(\|\mathbf{w}_i - \mathbf{w}^*\|^2 - \|\mathbf{w}_{i+1} - \mathbf{w}^*\|^2\right) + \frac{\eta L^2}{2} - \frac{\lambda}{2}\|\mathbf{w}_i - \mathbf{w}_\lambda^*\|^2. \end{aligned}$$

Making this substitution into Equation (3.16),

$$\begin{aligned}
&F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}^*_\lambda) \\
&\quad \le \frac{1}{2t\eta}\left(\|\mathbf{w}_1 - \mathbf{w}^*\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right) + \frac{\eta L^2}{2} \\
&\qquad - \frac{\lambda}{2t}\sum_{i=1}^{t}\left(\|\mathbf{w}_i - \mathbf{w}^*_\lambda\|^2 + ||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2\right) \\
&\quad \le \frac{R^2}{2t\eta} + \frac{\eta L^2}{2} - \frac{\lambda}{2t}\sum_{i=1}^{t}\left(\|\mathbf{w}_i - \mathbf{w}^*_\lambda\|^2 + ||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2\right) \\
&\quad \le \frac{RL}{\sqrt{t}} - \frac{\lambda}{2t}\sum_{i=1}^{t}\left(||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2 + \|\mathbf{w}_i - \mathbf{w}^*_\lambda\|^2\right).
\end{aligned}$$

Decomposing the sum,

$$\begin{aligned}
\frac{1}{t}\sum_{i=1}^{t}\|\mathbf{w}_i - \mathbf{w}^*_\lambda\|^2 &= \frac{1}{t}\sum_{i=1}^{t}\left(||\mathbf{w}_i||^2 - 2\mathbf{w}_i^\top\mathbf{w}^*_\lambda + ||\mathbf{w}^*_\lambda||^2\right) \\
&= \frac{1}{t}\sum_{i=1}^{t}\left(||\mathbf{w}_i||^2\right) - 2\overline{\mathbf{w}}^\top\mathbf{w}^*_\lambda + ||\mathbf{w}^*_\lambda||^2 \\
&= \frac{1}{t}\sum_{i=1}^{t}\left(||\mathbf{w}_i||^2\right) - ||\overline{\mathbf{w}}||^2 + ||\overline{\mathbf{w}}||^2 - 2\overline{\mathbf{w}}^\top\mathbf{w}^*_\lambda + ||\mathbf{w}^*_\lambda||^2 \\
&= \frac{1}{t}\sum_{i=1}^{t}\left(||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2\right) + ||\overline{\mathbf{w}} - \mathbf{w}^*_\lambda||^2.
\end{aligned}$$

Making this substitution,

$$\begin{aligned}
&F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}^*_\lambda) \\
&\quad \le \frac{RL}{\sqrt{t}} - \frac{\lambda}{t}\sum\left(||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2\right) - \frac{\lambda}{2}||\overline{\mathbf{w}} - \mathbf{w}^*_\lambda||^2.
\end{aligned}$$

Since $||\mathbf{w}||^2$ is convex, $\frac{\lambda}{t}\sum\left(||\mathbf{w}_i||^2 - ||\overline{\mathbf{w}}||^2\right) \ge 0$ and

$$F_\lambda(\overline{\mathbf{w}}) - F_\lambda(\mathbf{w}^*_\lambda) \le \frac{RL}{\sqrt{t}} - \frac{\lambda}{2}||\overline{\mathbf{w}} - \mathbf{w}^*_\lambda||^2$$

as desired. □

Proof of Lemma 3.4.2

We now prove Lemma 3.4.2, which bounds the distance between minimizers of F_λ for different regularization parameters λ.

Proof. Let $\mathbf{w}_\lambda^*$ minimize F_λ as given in Equation (3.2). Let $\lambda' > 0$ be such that $\mathbf{w}_\lambda^* = \mathbf{w}^*$ for all $\lambda \leq \lambda'$. For $\lambda, \widetilde{\lambda} \geq 0$ and data satisfying Assumption 3.2.1, we aim to show that

$$\|\mathbf{w}_\lambda^* - \mathbf{w}_{\widetilde{\lambda}}^*\| \leq \frac{L}{2}\left|\frac{1}{\lambda} - \frac{1}{\widetilde{\lambda}}\right|$$

and

$$\|\mathbf{w}_\lambda^* - \mathbf{w}_{\widetilde{\lambda}}^*\| \leq \frac{\max_j \|\mathbf{x}_j\|}{(\lambda')^2}|\lambda - \widetilde{\lambda}|.$$

The proof of Lemma 3.4.2 makes use of Lemma 8 of [LS18], which is also stated below.

Lemma 3.8.1. *(Perturbation of strongly convex functions I [LS18]). Let $f(\mathbf{z})$ be a non-negative, α^2-strongly convex function. Let $g(\mathbf{z})$ be a L-Lipschitz non-negative convex function. For any $\beta \geq 0$, let $\mathbf{z}[\beta]$ be the minimizer of $f(\mathbf{z}) + \beta g(\mathbf{z})$, then we have,*

$$\left\|\frac{d\mathbf{z}[\beta]}{d\beta}\right\| \leq \frac{L}{\alpha^2}.$$

Let $f(\mathbf{w}) = \|\mathbf{w}\|^2$ and $g(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^n \max\{0, 1 - y_j\mathbf{x}_j^\top \mathbf{w}\}$. Then f is strongly convex with strong convexity parameter 2 and g is Lipschitz with a Lipschitz constant bounded by $\frac{1}{n}\sum_{j=1}^n \|\mathbf{x}_j\|$. Note that

$$\begin{aligned} F_\lambda(\mathbf{w}) &= \frac{\lambda}{2}f(\mathbf{w}) + g(\mathbf{w}) = \frac{\lambda}{2}\left[f(\mathbf{w}) + \frac{2}{\lambda}g(\mathbf{w})\right] \\ &= \frac{\lambda}{2}\left[f(\mathbf{w}) + \beta(\lambda)g(\mathbf{w})\right] \end{aligned}$$

for $\beta(\lambda) = \frac{2}{\lambda}$. Applying Lemma 8 of [LS18],

$$\begin{aligned} \left\|\frac{d\mathbf{w}[\lambda]}{d\lambda}\right\| &= \left\|\frac{d\mathbf{w}[\lambda]}{d\beta(\lambda)} \cdot \frac{d\beta(\lambda)}{d\lambda}\right\| \\ &\leq \frac{1}{2n}\sum_{j=1}^n \|\mathbf{x}_j\| \cdot |\beta'(\lambda)| = \frac{\frac{1}{n}\sum_{j=1}^n \|\mathbf{x}_j\|}{\lambda^2}. \end{aligned}$$

Integrating, for any $\tilde{\lambda} \geq \hat{\lambda} > 0$, we have

$$\begin{aligned}
\|\mathbf{w}^*_{\tilde{\lambda}} - \mathbf{w}^*_{\hat{\lambda}}\| &= \left\| \int_{\hat{\lambda}}^{\tilde{\lambda}} \frac{d\mathbf{w}[\lambda]}{d\lambda} d\lambda \right\| \\
&\leq \int_{\hat{\lambda}}^{\tilde{\lambda}} \left\| \frac{d\mathbf{w}[\lambda]}{d\lambda} \right\| d\lambda \\
&\leq \int_{\hat{\lambda}}^{\tilde{\lambda}} \frac{\frac{1}{n}\sum_j \|\mathbf{x}_j\|}{\lambda^2} d\lambda \\
&= \tfrac{1}{n} \sum_j \|\mathbf{x}_j\| \left| \frac{1}{\tilde{\lambda}} - \frac{1}{\hat{\lambda}} \right|.
\end{aligned}$$

As the regularization parameter λ approaches zero, we will use the following bound. Since for all $\lambda < \lambda'$, $\mathbf{w}[\lambda] = \mathbf{w}[\lambda'] = \mathbf{w}^*$, then for $\lambda < \lambda'$, $\|\frac{d\mathbf{w}[\lambda]}{d\lambda}\| = 0$. Thus

$$\left\| \frac{d\mathbf{w}[\lambda]}{d\lambda} \right\| \leq \frac{\max_j \|\mathbf{x}_j\|}{(\lambda')^2} \quad \forall\, \lambda > 0.$$

This gives the second bound,

$$\|\mathbf{w}^*_{\tilde{\lambda}} - \mathbf{w}^*_{\hat{\lambda}}\| \leq \int_{\hat{\lambda}}^{\tilde{\lambda}} \frac{\frac{1}{n}\sum_j \|\mathbf{x}_j\|}{\lambda^2} d\lambda \leq \int_{\hat{\lambda}}^{\tilde{\lambda}} \frac{\frac{1}{n}\sum_j \|\mathbf{x}_j\|}{\lambda'^2} d\lambda \leq \frac{\max_j \|\mathbf{x}_j\|}{(\lambda')^2} |\widetilde{\lambda} - \hat{\lambda}|.$$

□

Proof of Lemma 3.4.3 We finally prove Lemma 3.4.3, which makes use of Lemma 3.4.1 and Lemma 3.4.2 to bound the initial error $\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda}\|$ of each regularized subproblem given in Equation (3.4).

Proof. We aim to show $\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq R_s$ with R_s defined below and proceed by induction. For $s_0 \in \mathbb{N}$ with $s_0 > 2$, $p \in (0, 1)$, and $r > 2p$, let $\lambda_s = (s_0 + s)^{-p}$, $t_s = (s_0 + s)^r$. Recall that $L = \frac{2}{n}\sum_{j=1}^n \|\mathbf{x}_j\|$. For some parameter $\alpha > 0$, let

$$R_s = CL(s_0 + s - 1)^{-\alpha} \text{ with } C = \max\left\{4, \frac{1}{2\lambda_0}(s_0 - 1)^{\alpha}\right\}.$$

By Lemma 3.2.2, and since $\overline{\mathbf{w}}_0 = \mathbf{0}$, we have $\|\overline{\mathbf{w}}_0 - \mathbf{w}^*_{\lambda_0}\| \leq \frac{L}{2\lambda_0}$. Note that $R_0 \geq \frac{L}{2\lambda_0}$ and thus the base case, $\|\overline{\mathbf{w}}_0 - \mathbf{w}^*_{\lambda_0}\| \leq R_0$ is satisfied.

Suppose that $\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq R_s$. By the triangle inequality,

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq \|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_{s-1}}\| + \|\mathbf{w}^*_{\lambda_{s-1}} - \mathbf{w}^*_{\lambda_s}\|.$$

For $\overline{\mathbf{w}}_s$ generated as in Algorithm 1, Lemma 3.4.1 along with the inductive assumption gives that

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_{s-1}}\| \leq \left(\frac{2R_{s-1}L}{\lambda_{s-1}\sqrt{t_{s-1}}}\right)^{1/2} = \frac{\sqrt{2C}L(s_0+s-2)^{-\alpha/2}}{(s_0+s-1)^{r/4-p/2}}.$$

From Equation (3.9) of Lemma 3.4.2,

$$\begin{aligned}\|\mathbf{w}^*_{\lambda_{s-1}} - \mathbf{w}^*_{\lambda_s}\| &\leq \tfrac{L}{2}\left(\tfrac{1}{\lambda_s} - \tfrac{1}{\lambda_{s-1}}\right)\\ &= \tfrac{L}{2}\left((s_0+s)^p - (s_0+s-1)^p\right)\\ &\leq \tfrac{Lp}{2}(s_0+s-1)^{p-1}.\end{aligned}$$

Combining these

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq \frac{\sqrt{2C}L(s_0+s-2)^{-\alpha/2}}{(s_0+s-1)^{r/4-p/2}} + \frac{L}{2}p(s_0+s-1)^{p-1}.$$

Applying a change of base via $\epsilon \geq \frac{\log(s_0+s-1)-\log(s_0+s-2)}{\log(s_0+s-1)}$,

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq \sqrt{2C}L(s_0+s-1)^{p/2-\alpha/2(1-\epsilon)-r/4} + \frac{Lp}{2}(s_0+s-1)^{p-1}.$$

To simplify the analysis and remove the dependence of ϵ on the iteration number s, we use $\epsilon_0 = \frac{\log(s_0)-\log(s_0-1)}{\log(s_0)}$. Now, for

$$0 \leq \alpha \leq \min\left(\frac{r-2p}{2(1+\epsilon_0)}, 1-p\right)$$

and $p < 1$, we have

$$\|\overline{\mathbf{w}}_s - \mathbf{w}^*_{\lambda_s}\| \leq L\left(\sqrt{2C} + \frac{p}{2}\right)(s_0+s-1)^{-\alpha} \leq CL(s_0+s-1)^{-\alpha} = R_s.$$

Note that allowing the first term in the upper bound on α to increase with s leads to smaller bounds R_s. This choice, however, complicates the analysis.

□

CHAPTER 4

Adaptive Sketch-and-Project Methods

4.1 Introduction

We consider the fundamental problem of finding an approximate solution to the linear system

$$\mathbf{A}x = b, \tag{4.1}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Given the possibility of multiple solutions, we set out to find a least-norm solution given by

$$x^* \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \tfrac{1}{2}\|x\|_{\mathbf{B}}^2 \quad \text{subject to} \quad \mathbf{A}x = b, \tag{4.2}$$

where $\mathbf{B} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $\|x\|_{\mathbf{B}}^2 \overset{\text{def}}{=} \langle \mathbf{B}x, x\rangle$. Here, we consider consistent systems, for which there exists an x that satisfies Equation (4.1).

When the dimensions of $\mathbf{A}$ are large, direct methods for solving Equation (4.2) can be infeasible, and iterative methods are favored. In particular, Krylov subspace iterative methods including the conjugate gradient algorithms [HS52] are the industrial standard so long as one can afford full matrix vector products and the system matrix fits in memory. On the other hand, if a single matrix vector product is considerably expensive, or $\mathbf{A}$ is too large to fit in memory, then randomized iterative methods such as the randomized Kaczmarz [Kac37, SV09] and coordinate descent method [MNR15b, LL10] are effective.

4.1.1 Randomized Kacmarz

The randomized Kaczmarz method is typically used to solve linear systems of equations in the large data regime, i.e., when the number of samples m is much larger than the dimension n.

The Kaczmarz method was originally proposed in 1937 and has seen applications in computer tomography (CT scans), signal processing, and other areas [Kac37, SV09, GBH70, Nat01]. In each iteration k, the current iterate x^k is projected onto the solution space of a selected row of the linear system of Equation (4.1). Specifically, at each iteration

$$x^{k+1} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \|x - x^k\|^2 \quad \text{subject to} \quad \mathbf{A}_{i_k:} x = b_{i_k},$$

where $\mathbf{A}_{i_k:}$ is the row of $\mathbf{A}$ selected at iteration k. Let $\mathbf{A}_{i_k:}^\top$ denote the transpose of this row. The Kaczmarz update can be written explicitly as

$$x^{k+1} = x^k + \frac{b_{i_k} - \langle \mathbf{A}_{i_k:}, x^k \rangle}{\|\mathbf{A}_{i_k:}\|^2} \mathbf{A}_{i_k:}^\top. \tag{4.3}$$

4.1.2 Coordinate descent

Coordinate descent is commonly used for optimizing general convex optimization functions when the dimensions are extremely large, since at each iteration only a single coordinate (or dimension) is updated [RT14, RT13]. Here, we consider coordinate descent applied to Equation (4.2). In this setting, it is sometimes referred to as randomized Gauss-Seidel [MNR15b, LL10].

At iteration k an index $i \in \{1, \ldots, n\}$ is selected and the coordinate x_i^k of the current iterate x^k is updated such that the least-squares objective $\|b - \mathbf{A}x\|^2$ is minimized. More formally,

$$x^{k+1} = \underset{x \in \mathbb{R}^n,\, \lambda \in \mathbb{R}}{\operatorname{argmin}} \|b - \mathbf{A}x\|^2 \quad \text{subject to} \quad x = x^k + \lambda\, e^i,$$

where e^i is the ith coordinate vector. Let $\mathbf{A}_{:i}$ denote the ith column of $\mathbf{A}$ and $\mathbf{A}_{i_k:}^\top$ denote the transpose of this column. The explicit update for coordinate descent applied to Equation (4.2) is given by

$$x^{k+1} = x^k - \frac{\mathbf{A}_{:i_k}^\top (\mathbf{A}x^k - b)}{\|\mathbf{A}_{:i_k}\|^2} e^{i_k}. \tag{4.4}$$

4.1.3 Sketch-and-project methods

Sketch-and-project is a general archetypal algorithm that unifies a variety of randomized iterative methods including both randomized Kaczmarz and coordinate descent along with all of their block variants [GR15b]. At each iteration, sketch-and-project methods project the current iterate onto a subsampled or sketched linear system with respect to some norm. Let $\mathbf{B} \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. We will consider the projection with respect to the $\mathbf{B}$–norm given by $\|\cdot\|_{\mathbf{B}} = \sqrt{\langle \cdot, \mathbf{B}\cdot\rangle}$.

Let $\mathbf{S}_i \in \mathbb{R}^{m\times\tau}$ for $i = 1, \dots, q$ be the set of *sketching matrices* where $\tau \in \mathbb{N}$ is the *sketch size*. In general, the set of sketching matrices $\mathbf{S}_i$ could be infinite, however, here, we restrict ourselves to a finite set of $q \in \mathbb{N}$ sketching matrices. At the k^{th} iteration of the sketch-and-project algorithm, a sketching matrix $\mathbf{S}_i$ is selected and the current iterate x^k is projected onto the solution space of the sketched system $\mathbf{S}_{i_k}^\top \mathbf{A}x = \mathbf{S}_{i_k}^\top b$ with respect to the $\mathbf{B}$–norm. Given a selected index $i_k \in \{1, \dots, q\}$ the sketch-and-project update solves

$$x^{k+1} = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \|x - x^k\|_{\mathbf{B}}^2 \quad \text{subject to} \quad \mathbf{S}_{i_k}^\top \mathbf{A}x = \mathbf{S}_{i_k}^\top b. \tag{4.5}$$

The closed form solution to Equation (4.5) is given by

$$x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b), \tag{4.6}$$

where

$$\mathbf{H}_i \overset{\text{def}}{=} \mathbf{S}_i(\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger \mathbf{S}_i^\top, \quad \text{for } i = 1, \dots, q, \tag{4.7}$$

and $\dagger$ denotes the pseudoinverse.

One can recover the randomized Kaczmarz method under the sketch-and-project framework by choosing the matrix $\mathbf{B}$ as the identity matrix and sketches $\mathbf{S}_i = e^i$. If instead $\mathbf{B} = \mathbf{A}^\top\mathbf{A}$ and sketches $\mathbf{S}_i = \mathbf{A}e^i = \mathbf{A}_{:i}$, where $\mathbf{A}_{:i}$ is the i^{th} column of the matrix $\mathbf{A}$, then the resulting method is coordinate descent.

4.1.4 Sampling of indices

An important component of the methods above is the selection of the index i_k at iteration k. Methods often use independently and identically distributed (i.i.d.) indices, as this choice makes the method and analysis relatively simple [SV09, Nes12]. In addition to choosing indices i.i.d. at each iteration, several adaptive sampling methods have also been proposed, which we discuss next. These sampling strategies use information about the current iterate in order to improve convergence guarantees over i.i.d. random sampling strategies at the cost of extra calculation per iteration. Under certain conditions, such strategies can be implemented with only a marginal additional cost per iteration.

4.1.4.1 Sampling for the Kaczmarz method

The original Kaczmarz method cycles through the rows of the matrix $\mathbf{A}$ and makes projections onto the solution space with respect to each row [Kac37]. In 2009, Strohmer and Vershynin suggested selecting rows with probabilities that are proportional to the squared row norms (i.e. $p_i \propto \|\mathbf{A}_{i:}\|_2^2$) and provided the first proof of exponential convergence of the randomized Kaczmarz method [SV09].

Several adaptive selection strategies have also been proposed in the Kaczmarz setting. The max-distance Kaczmarz or Motzkin's method selects the index i_k at iteration k that leads to the largest magnitude update [NSL16, MS54]. In addition to the max-distance selection rule, Nutini et al also consider the greedy selection rule that chooses the row corresponding to the maximal residual component, i.e., $i_k = \text{argmax}_i \, |\mathbf{A}_{i:}x^k - b_i|$ at each iteration, but show that the max-distance Kacmzarz method performs at least as well as this strategy [NSL16]. More complicated adaptive methods have also been suggested for randomized Kaczmarz, such as the capped sampling strategies proposed in [BW18a, BW18b, BW19a] or the sampling Kaczmarz Motzkin's method of [LHN17].

4.1.4.2 Sampling for coordinate descent

For coordinate descent, several works have investigated adaptive coordinate selection strategies [PCJ17, NSL15, Nes12, AG18]. As coordinate descent is not restricted to solving linear systems, these works often consider more general convex loss functions. A common greedy selection strategy for coordinate descent applied to differentiable loss functions is to select the coordinate that corresponds to the maximal gradient component, which is known as the Gauss-Southwell rule [Tse90, LT92, NSL15, Nes12] or adaptively according to a duality gap [CQR15].

4.1.4.3 Sampling for sketch-and-project

The problem of determining the optimal fixed probabilities with which to select the index i_k at each iteration k was shown in Section 5.1 of [GR15b] to be a convex semi-definite program, which is often a harder problem than solving the original linear system. The problem of determining the optimal adaptive probabilities is even harder as one must consider the effects of the current index selection on the future iterates. Here, instead, we present adaptive sampling rules that are not necessarily optimal, but can be efficiently implemented and are proven to converge faster than the fixed non-adaptive rules.

4.1.5 Choosing the sketches and preconditioning

Another key question is how we should choose the set of sketching matrices. This question has been partially answered in Section 5.2 of [GR17], wherein the authors show that if a preconditioned $\mathbf{A}$ were available, then the set of sketching matrices should be drawn from row partitions or column partitions of this preconditioner. This strategy can be combined with any index sampling rule for an overall faster algorithm. Here, we will assume a set of sketching matrices has been provided, and focus only on the index sampling rule.

4.1.6 Additional related works

Various related works consider extensions to solving Equation (4.2) in the randomized Kaczmarz, coordinate descent and sketch-and-project settings. The following summary of related works is not exhaustive. While we consider consistent linear systems, others have analyzed and extended sketch-and-project methods to handle inconsistent linear systems [PP16, ZF13, Pop99, MNR15a, Dum15]. An adaptive maximum-residual sampling strategy has also been analyzed for the inconsistent extension [PP16]. The randomized Kaczmarz method has also been studied in the context of solving systems of linear inequalities [LL10, MS54, BN15, BW19b]. Block and accelerated variants of randomized Kaczmarz and coordinate descent have also been analyzed [RT14, Nec19, NZZ15, NT14, LX15, NNG17, NS17]. Recent works have considered combining ideas from random sketching methods with those from the sketch-and-project framework [PJM19].

4.2 Contributions

Adaptive sampling strategies have not yet been analyzed for the general sketch-and-project framework. We introduce three different adaptive sampling rules for the general sketch-and-project method: the max-distance sampling rule, the capped-adaptive sampling rule, and proportional sampling probabilities. We prove that each of these methods converge exponentially in mean squared error with convergence guarantees that are strictly faster than the guarantees for sampling indices uniformly.

4.2.1 Key quantity: Sketched loss

As we will see in the general convergence analysis of the sketch-and-project method detailed in Section 4.7, the convergence at each iteration depends on the current iterate x^k and a key quantity known as the sketched loss

$$f_i(x^k) \overset{\text{def}}{=} \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}, \tag{4.8}$$

of the sketch $\mathbf{S}_i$ (recall that $\mathbf{H}_i$, defined in Equation (4.7), is symmetric positive semi-definite and thus $\|\cdot\|_{\mathbf{H}_i} \overset{\text{def}}{=} \sqrt{\langle \cdot, \mathbf{H}_i \cdot \rangle}$ gives a semi-norm). This sketched loss was introduced in [RT17] where the authors show that the sketch-and-project method can be seen as a stochastic gradient method (we expand on this in Section 4.4). We show that using adaptive selection rules based on the sketched losses results in new methods with faster convergence guarantees.

4.2.2 Max-distance rule

We introduce the max-distance sketch-and-project method, which is a generalization of both the max-distance Kaczmarz method (also known as Motzkin's method) [NSL16, MS54, HN19], greedy coordinate descent (Gauss-Southwell rule [NSL15]), and all their possible block variants.

Nutini et al. showed that the max-distance Kaczmarz method performs at least as well as uniform sampling and the non-uniform sampling method of [SV09], in which rows are sampled with probabilities proportional to the squared row norms of $\mathbf{A}$ [NSL16]. We extend this result to the general sketch-and-project setting and also show that the max-distance rule leads to a convergence guarantee that is *strictly* faster than that of any fixed probability distribution.

4.2.3 The capped adaptive rule

A new family of adaptive sampling methods were recently proposed for the Kaczmarz and coordinate descent type methods [BW18a, BW18b, BW19a]. We extend these methods to the sketch-and-project setting, which allows for their application in other settings such as for coordinate descent. While introduced under the names greedy randomized Kaczmarz and relaxed greedy randomized Kaczmarz, we refer to these methods as *capped adaptive* methods because they select indices i whose corresponding sketched losses $f_i(x^k)$ are larger than a capped threshold given by a convex combination of the largest and average sketched losses. These sampling strategies were introduced as 'greedy randomized' sampling rules [BW18a, BW18b, BW19a], however, we rename them here to prevent confusion with the greedy max-distance sampling rule. It was proven in [BW18a] that the convergence guarantee when using the capped adaptive rule is strictly faster than the fixed non-uniform sampling rule given in [SV09]. In Subsection 4.7.5, we generalize this capped adaptive sampling to sketch-and-project methods and prove that the resulting convergence guarantee of this adaptive rule is slower than that of the max-distance rule. Furthermore, in Subsection 4.A.3, we show that the max-distance rule requires less computation at each iteration than the capped adaptive rule.

4.2.4 The proportional adaptive rule

We also present a new and much simpler randomized adaptive rule as compared to the capped adaptive rule discussed above, in which indices are sampled with probabilities that are directly *proportional* to their corresponding sketched losses $f_i(x^k)$. We show that this rule gives a resulting convergence that is at least twice as fast as when sampling the sketches uniformly.

4.2.5 Efficient implementations

Our adaptive methods come with the added cost of computing the sketched loss $f(x^k)$ of Equation (4.8) at each iteration. Fortunately, the sketched loss can be computed efficiently with certain precomputations as discussed in Section 4.8. We show how the sketched losses can be maintained efficiently via an auxiliary update, leading to reasonably efficient implementations of the adaptive sampling rules. We demonstrate improved performance of the adaptive methods over uniform sampling when solving linear systems with both real and synthetic matrices per iteration and in terms of the flops required.

4.2.6 Consequences and future work

Our results on adaptive sampling have consequences on many other closely related problems. For instance, an analogous sampling strategy to our proportional adaptive rule has been proposed for coordinate descent in the primal-dual setting for optimizing regularized loss functions [PCJ17]. Also a variant of adaptive and greedy coordinate descent has been shown to speed-up the solution of the matrix scaling problem [AG18]. The matrix scaling problem is equivalent to an entropy-regularized version of the optimal transport problem which has numerous applications in machine learning and computer vision [AG18, Cut13]. Thus the adaptive methods proposed here may be extended to these other settings such as adaptive coordinate descent for more general smooth optimization [PCJ17]. The adaptive methods and the analysis proposed in this paper may also provide insights toward adaptive sampling for other classes of optimization methods such as stochastic gradient, since the randomized Kaczmarz method can be reformulated as stochastic gradient descent applied to the least-squares problem [NSW15].

4.3 Notation

We now introduce notation that will be used throughout. Let Δ_q denote the simplex in $\mathbb{R}^q$, that is

$$\Delta_q \overset{\text{def}}{=} \{p \in \mathbb{R}^q \ : \ \sum_{i=1}^{q} p_i = 1, \ p_i \geq 0, \ \text{for } i = 1, \ldots, q\}.$$

For probabilities $p \in \Delta_q$ and values x_i depending on an index $i = 1, \ldots, q$, we denote $\mathbb{E}_{i \sim p}[x_i] \overset{\text{def}}{=} \sum_{i=1}^{q} p_i x_i$, where $i \sim p$ indicates that i is sampled with probability p_i. At the k^{th} iteration of the sketch-and-project algorithm, a sketching matrix $\mathbf{S}_{i_k}$ is sampled with probability

$$\mathbb{P}[\mathbf{S}_{i_k} = \mathbf{S}_i \mid x^k] = p_i^k, \quad \text{for } i = 1, \ldots, q, \tag{4.9}$$

where $p^k \in \Delta_q$ and we use $p^k \overset{\text{def}}{=} (p_1^k, \ldots, p_q^k)$ to denote the vector containing these probabilities. We drop the superscript k when the probabilities do not depend on the iteration.

For any symmetric positive semi-definite matrix $\mathbf{G}$ we write the semi-norm induced by $\mathbf{G}$ as $\|\cdot\|_{\mathbf{G}}^2 \overset{\text{def}}{=} \langle \cdot, \mathbf{G} \cdot \rangle$, while $\|\cdot\|$ denotes the standard 2-norm ($\|\cdot\|_2$). For any matrix $\mathbf{M}$, $\|\mathbf{M}\|_F \overset{\text{def}}{=} \sqrt{\sum_{i,j} \mathbf{M}_{ij}^2}$. We use

$$\lambda_{\min}^{+}(\mathbf{G}) \overset{\text{def}}{=} \min_{v \in \operatorname{range} \mathbf{G}, v \neq 0} \frac{\|v\|_{\mathbf{G}}^2}{\|v\|_2^2},$$

to denote the smallest non-zero eigenvalue of $\mathbf{G}$.

4.3.1 Organization

The remainder of the paper is organized as follows. Sections 4.4 and 4.5 provide additional background on the sketch-and-project method and motivation for adaptive sampling in this setting. Section 4.4 explains how the sketch-and-project method can be reformulated as stochastic gradient descent. The sampling of the sketches can then be seen as importance sampling in the context of stochastic gradient descent. Section 4.5 provides geometric intuition for the sketch-and-project method and motivates why one would expect adaptive sampling strategies that depend on the sketched losses $f_i(x^k)$ to perform well.

Section 4.6 introduces the various sketch selection strategies considered throughout the paper, while Section 4.7 provides convergence guarantees for each of the resulting methods. In Section 4.8, we discuss the computational costs of adaptive sketch-and-project for the sketch selection strategies of Section 4.6 and suggest efficient implementations of the methods. Section 4.9 discusses convergence and computational cost for the special subcases of randomized Kaczmarz and coordinate descent. Performance of adaptive sketch-and-project methods are demonstrated in Section 4.10 for both synthetic and real matrices.

4.4 Reformulation as importance sampling for stochastic gradient descent

The sketch-and-project method can be reformulated as a stochastic gradient method, as shown in [RT17]. We use this reformulation to motivate our adaptive sampling as a variant of importance sampling.

Let $p \in \Delta_q$. Consider the stochastic program

$$\min_{x\in\mathbb{R}^d} F(x) \overset{\text{def}}{=} \mathbb{E}_{i\sim p}\left[f_i(x)\right] = \mathbb{E}_{i\sim p}\left[\|\mathbf{A}x-b\|^2_{\mathbf{H}_i}\right]. \tag{4.10}$$

Objective functions $F(x)$ such as the one in Equation (4.10) are common in machine learning, where $f_i(x)$ often represents the loss with respect to a single data point.

When $\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]$ is invertible, solving Equation (4.10) is equivalent to solving the linear system Equation (4.1). This invertibility condition on $\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]$ can be significantly relaxed by using the following technical exactness assumption on the probability p and the set of sketches introduced in [RT17].

Assumption 4.4.1. *Let $p \in \Delta_q$, $\Sigma \overset{def}{=} \{S_1,\ldots,S_q\}$ be a set of sketching matrices and $\mathbf{H}_i$ as defined in Equation (4.7). We say that the exactness assumption holds for (p,Σ) if*

$$\mathit{Null}(\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]) \subset \mathit{Null}\left(\mathbf{A}^\top\right).$$

This exactness assumption guarantees[1] that

$$\text{Null}\left(\mathbf{A}\right) = \text{Null}\left(\mathbf{A}^\top\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}\right). \tag{4.11}$$

[1]This can be shown by applying Lemma 4.B.1 in Section 4.B with with $\mathbf{G} = \mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]$ and $\mathbf{W} = \mathbf{A}$.

This in turn guarantees that the expected sketched loss of the point x is zero if and only if $\mathbf{A}x = b$. Indeed, by taking the derivative of (4.10) and setting it to zero we have that

$$\nabla F(x) \;=\; \mathbf{A}^\top \mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right](\mathbf{A}x - b) \;=\; \mathbf{A}^\top \mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}(x - x^*) \;=\; 0.$$

Thus, every minimizer x of Equation (4.10) is such that

$$x - x^* \;\in\; \mathrm{Null}\left(\mathbf{A}^\top \mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}\right) \overset{(4.11)}{=} \mathrm{Null}\left(\mathbf{A}\right), \tag{4.12}$$

thus $\mathbf{A}(x - x^*) = \mathbf{A}x - b = 0$. As shown in [GR15a] and [RT17] this exactness assumption holds trivially for most practical sketching techniques.

When the number of f_i functions is large, the SGD (stochastic gradient descent) method is typically the method of choice for solving Equation (4.10). To view the sketch-and-project update in Equation (4.6) as a SGD method, we sample an index $i_k \sim p$ at each iteration and take a step

$$x^{k+1} = x^k - \nabla^{\mathbf{B}} f_{i_k}(x^k), \tag{4.13}$$

where $\nabla^{\mathbf{B}} f_{i_k}(x^k)$ is the gradient taken with respect to the $\mathbf{B}$–norm. For $f_i(x^k)$ of Equation (4.8), the exact expression of this stochastic gradient is given by

$$\nabla^{\mathbf{B}} f_{i_k}(x^k) = \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b). \tag{4.14}$$

By plugging Equation (4.14) into Equation (4.13) we can see that the resulting update is equivalent to a the sketch-and-project update in Equation (4.6).

Though the indices $i \in \{1, \dots, q\}$ are often sampled uniformly at random for SGD, many alternative sampling distributions have been proposed in order to accelerate convergence, including adaptive sampling strategies [CR18, JZ13, NSW15, ZZ15, KF18, LH15, ALS15]. Such sampling strategies give more weight to sampling indices corresponding to a larger loss $f_i(x)$ or a larger gradient norm $\|\nabla^{\mathbf{B}} f_i(x)\|_{\mathbf{B}}^2$. In the sketch-and-project setting, it is not hard to show[2] that these two sampling strategies result in similar methods since

$$f_i(x) = \|\mathbf{A}x - b\|_{\mathbf{H}_i}^2 = \|\nabla^{\mathbf{B}} f_i(x)\|_{\mathbf{B}}^2.$$

[2]See Lemma 3.1 in [RT17].

In general, updating the loss and gradient of every $f_i(x)$ at each iteration can be too expensive. Thus many methods resort to using global approximations of these values such as the Lipschitz constant of the gradient [NSW15] that lead to fixed data-dependent sample distributions. For the sketch-and-project setting, we demonstrate in Section 4.8 that the adaptive sample distributions can be calculated efficiently, with a per-iterate cost on the same order as is required for the sketch-and-project update.

4.5 Geometric viewpoint and motivational analysis

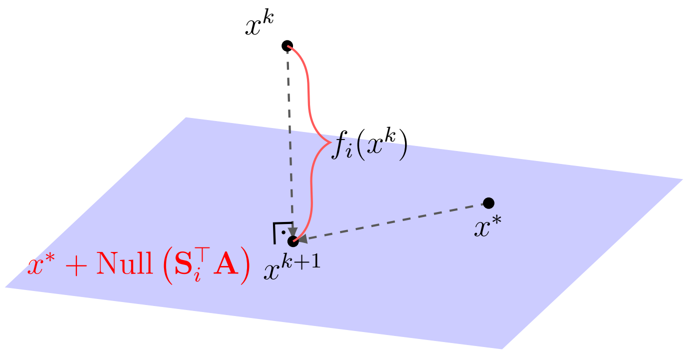

x^k
$f_i(x^k)$
x^*
$x^* + \mathrm{Null}\left(\mathbf{S}_i^\top \mathbf{A}\right)$
x^{k+1}

Figure 4.1: The geometric interpretation of Equation (4.5), as the projection of x^k onto a random affine space that contains x^*. The distance traveled is given by $f_i(x^k) = \|x^{k+1} - x^k\|^2_{\mathbf{B}}$.

The sketch-and-project method given in Equation (4.5) can be seen as a method that calculates the next iterate x^{k+1} by projecting the previous iterate x^k onto a random affine space. Indeed, the constraint in Equation (4.5) can be re-written as

$$\{x \,:\, \mathbf{S}_i^\top \mathbf{A} x = \mathbf{S}_i^\top b\} \quad = \quad x^* + \mathrm{Null}\left(\mathbf{S}_i^\top \mathbf{A}\right). \tag{4.15}$$

In particular, Equation (4.5) is an orthogonal projection of the point x^k onto an affine space that contains x^* with respect to the $\mathbf{B}$–norm. See Figure 4.1 for an illustration. This projection is determined by the following projection operator.

Lemma 4.5.1. *Let*

$$\mathbf{Z}_i \overset{def}{=} \mathbf{B}^{-1/2}\mathbf{A}^\top \mathbf{S}_i(\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1/2} = \mathbf{B}^{-1/2}\mathbf{A}^\top \mathbf{H}_i \mathbf{A}\mathbf{B}^{-1/2}, \tag{4.16}$$

for $i = 1, \ldots, q$, which is the orthogonal projection matrix onto $\operatorname{range} \mathbf{B}^{-1/2}\mathbf{A}^\top\mathbf{S}_i$*. Consequently*

$$\mathbf{Z}_i\mathbf{Z}_i = \mathbf{Z}_i, \quad \textit{and} \quad (\mathbf{I} - \mathbf{Z}_i)\mathbf{Z}_i = 0. \tag{4.17}$$

Furthermore we have that $(\mathbf{I} - \mathbf{Z}_i)$ *gives the projection depicted in Figure 4.1 since*

$$\mathbf{B}^{1/2}(x^{k+1} - x^*) \;=\; (\mathbf{I} - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^*). \tag{4.18}$$

Finally we can re-write the sketched loss as

$$f_i(x) \;=\; \|\mathbf{B}^{1/2}(x - x^*)\|^2_{\mathbf{Z}_i}, \quad \textit{for } i = 1, \ldots, q. \tag{4.19}$$

Proof. The proof of Equation (4.17) relies on standard properties of the pseudoinverse and is given in Lemma 2.2 in [GR15b].

As for the proof of Equation (4.18), subtracting x^* from both sides of Equation (4.6) we have that

$$\begin{aligned} x^{k+1} - x^* &= x^k - x^* - \mathbf{B}^{-1}\mathbf{A}^\top\mathbf{H}_{i_k}(\mathbf{A}x^k - b) \\ &\overset{\mathbf{A}x^*=b}{=} x^k - x^* - \mathbf{B}^{-1/2}\mathbf{B}^{-1/2}\mathbf{A}^\top\mathbf{H}_{i_k}\mathbf{A}\mathbf{B}^{-1/2}\mathbf{B}^{1/2}(x^k - x^*) \\ &\overset{(4.16)}{=} x^k - x^* - \mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*). \end{aligned} \tag{4.20}$$

It now only remains to multiply both sides by $\mathbf{B}^{1/2}$.

Finally the proof of Equation (4.19) follows by using $\mathbf{A}x^* = b$ together with the definitions of $\mathbf{H}_i$ and $\mathbf{Z}_i$ given in Equation (4.7) and Equation (4.16) so that

$$f_i(x) = \|\mathbf{A}(x - x^*)\|^2_{\mathbf{H}_i} = \|x - x^*\|^2_{\mathbf{A}^\top\mathbf{H}_i\mathbf{A}} \overset{(4.16)}{=} \|\mathbf{B}^{1/2}(x - x^*)\|^2_{\mathbf{Z}_i}. \tag{4.21}$$

□

With the explicit expression for the projection operator we can calculate the progress made by a single iteration of the sketch-and-progress method. The convergence proofs later on in Section 4.7 will rely heavily on Lemmas 4.5.2 and 4.5.3.

Lemma 4.5.2. *Let $x^k \in \mathbb{R}^d$ and let x^{k+1} be given by Equation* (4.5)*. Then the squared magnitude of the update is*

$$\|x^{k+1} - x^k\|^2_{\mathbf{B}} = f_{i_k}(x^k), \tag{4.22}$$

and the error from one iteration to the next decreases according to

$$\|x^{k+1} - x^*\|_{\mathbf{B}}^2 = \|x^k - x^*\|_{\mathbf{B}}^2 - f_{i_k}(x^k). \tag{4.23}$$

Proof. We begin by deriving Equation (4.23). Taking the squared norm in Equation (4.18) we have

$$\begin{aligned}
\|x^{k+1} - x^*\|_{\mathbf{B}}^2 &= \|(\mathbf{I} - \mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2})(x^k - x^*)\|_{\mathbf{B}}^2 \\
&= \|(\mathbf{I} - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^*)\|_2^2 \\
&= \left\langle \mathbf{B}^{1/2}(x^k - x^*), (I - \mathbf{Z}_{i_k})(I - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^*)\right\rangle \\
&\overset{(4.17)}{=} \left\langle \mathbf{B}^{1/2}(x^k - x^*), (I - \mathbf{Z}_{i_k})\mathbf{B}^{1/2}(x^k - x^*)\right\rangle \\
&= \|x^k - x^*\|_{\mathbf{B}}^2 - \left\langle \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*), \mathbf{B}^{1/2}(x^k - x^*)\right\rangle \\
&\overset{(4.19)}{=} \|x^k - x^*\|_{\mathbf{B}}^2 - f_i(x^k).
\end{aligned} \tag{4.24}$$

Finally we establish Equation (4.22) by subtracting x^k from both sides of Equation (4.6) so that

$$x^{k+1} - x^k = -\mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*).$$

It now remains to take the squared $\mathbf{B}$–norm and use Equation (4.19). □

Equation (4.22) shows that the distance traveled from x^k to x^{k+1} is given by the sketch residual $f_{i_k}(x^k)$, as we have depicted in Figure 4.1. Furthermore, Equation (4.23) shows that the contraction of the error $x^{k+1} - x^*$ is given by $-f_{i_k}(x^k)$. Consequently Lemma 4.5.2 indicates that in order to make the most progress in one step, or maximize the distance traveled, we should choose i_k corresponding to the largest sketched loss $f_{i_k}(x^k)$. We refer to this greedy sketch selection as the max-distance rule, which we explore in detail in Subsection 4.6.3.

Next we give the expected decrease in the error.

Lemma 4.5.3. *Let $p^k \in \Delta_q$. Consider the iterates of the sketch-and-project method given in Equation* (4.6) *where $i_k \sim p_i^k$ as is done in Algorithm 3. It follows that*

$$\mathbb{E}_{i\sim p^k}\left[\|x^{k+1} - x^*\|_{\mathbf{B}}^2 \mid x^k\right] \;=\; \|x^k - x^*\|_{\mathbf{B}}^2 - \mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right].$$

Proof. The result follows by taking the expectation over Equation (4.23) conditioned on x^k. $\square$

Lemma 4.5.3 suggests choosing adaptive probabilities so that $\mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right]$ is large. This analysis motivates the adaptive methods described in Subsection 4.6.2.

4.6 Selection rules

Motivated by Lemmas 4.5.2 and 4.5.3, we might think that sampling rules that prioritize larger entries of the sketched loss should converge faster. From this point we take two alternatives, 1) choose the i_k that maximizes the decrease (Subsection 4.6.3) or 2) choose a probability distribution that prioritizes the biggest decrease (Subsection 4.6.2). Below, we describe several sketch-and-project sampling strategies (fixed, adaptive, and greedy) and analyze their convergence in Section 4.7. The adaptive and greedy sampling strategies require knowledge of the current sketched loss vector at each iteration. Calculating the sketched loss from scratch is expensive, thus in Section 4.8 we will show how to efficiently calculate the new sketched loss $f(x^{k+1})$ using the previous sketched loss $f(x^k)$.

4.6.1 Fixed sampling

We first recall the standard non-adaptive sketch-and-project method that will be used as a comparison for the greedy and adaptive versions. In the non-adaptive setting the sketching matrices are sampled from a fixed distribution that is independent of the current iterate x^k. For reference, the details of the non-adaptive sketch-and-project method are provided in Algorithm 2.

4.6.2 Adaptive probabilities

Equation (4.23) motivates selecting indices that correspond to larger sketched losses with higher probability. We refer to such sampling strategies as adaptive sampling strategies, as they depend on the current iterate and its corresponding sketched loss values. In the adaptive

Algorithm 2 Non-Adaptive Sketch-and-Project

```
1: input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $p \in \Delta_q$, and a set of sketching matrices $\mathbf{S} = [\mathbf{S}_1, \dots, \mathbf{S}_q]$
2: for $k = 0, 1, 2, \dots$ do
3:     $i_k \sim p_i$
4:     $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$
5: end for
6: output: last iterate $x^{k+1}$
```

setting, we sample indices at the kth iteration with probabilities given by $p^k \in \Delta_q$. Adaptive sketch-and-project is detailed in Algorithm 3.

Algorithm 3 Adaptive Sketch-and-Project

```
1: input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and a set of sketching matrices $\mathbf{S} = [\mathbf{S}_1, \dots, \mathbf{S}_q]$
2: for $k = 0, 1, 2, \dots$ do
3:     $f_i(x^k) = \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}$ for $i = 1, \dots, q$
4:     Calculate $p^k \in \Delta_q$            ▷ Typically based on $f(x^k)$
5:     $i_k \sim p_i^k$
6:     $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$
7: end for
8: output: last iterate $x^{k+1}$
```

4.6.3 Max-distance rule

We refer to the greedy sketch selection rule given by

$$i_k \in \underset{i=1,\dots,q}{\operatorname{argmax}}\, f_i(x^k) = \underset{i=1,\dots,q}{\operatorname{argmax}}\, \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}, \tag{4.25}$$

as the max-distance selection rule. If multiple indices lead to the maximal sketched loss, any of these indices can be chosen. Per iteration, the max-distance rule leads to the best expected decrease in mean squared error. The max-distance sketch-and-project method is described in Algorithm 4. This greedy selection strategy has been studied for several specific choices of $\mathbf{B}$ and sketching methods. For example, in the Kaczmarz setting, this strategy is typically referred to as max-distance Kaczmarz or Motzkin's method [GO12, NSL16, MS54]. For coordinate descent, this selection strategy is the Gauss-Southwell rule [Nes12, NSL15]. We provide a convergence analysis for the general sketch-and-project max-distance selection rule in Theorem 4.7.6. We further show that max-distance selection leads to a convergence rate that is strictly larger than the resulting convergence rate when sampling from any fixed distribution in Theorem 4.7.8.

Algorithm 4 Max-Distance Sketch-and-Project

```
1: input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and a set of sketching matrices $\mathbf{S} = [\mathbf{S}_1, \dots, \mathbf{S}_q]$
2: for $k = 0, 1, 2, \dots$ do
3:     $f_i(x^k) = \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}$ for $i = 1, \dots, q$
4:     $i_k = \arg\max_{i=1,\dots,q} f_i(x^k)$
5:     $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$
6: end for
7: output: last iterate $x^{k+1}$
```

4.7 Convergence

We now present convergence results for the max-distance selection rule, uniform sampling, and adaptive sampling with probabilities proportional to the sketched loss. We summarize the convergence rate guarantees discussed throughout Section 4.7 in Table 4.1. Note that these are worst case convergence guarantees and thus may not reflect the expected performance of each selection rule. Our first step in the analysis is to establish an invariance property of the iterates in the following lemma.[3] In particular, Lemma 4.7.1 guarantees the error vectors $x^k - x^*$ remain in the subspace range $\mathbf{B}^{-1}\mathbf{A}^\top$ for all iterations if $x^0 \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$, which allows for a tighter convergence analysis.

Lemma 4.7.1. *If $x^0 \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$ then $x^k - x^* \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$.*

[3]This lemma was first presented in [GR15a]. We present and prove it here for completeness.

Proof. First note that $x^* \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$. This follows by taking the Lagrangian of Equation (4.2) given by

$$L(x,\lambda) = \tfrac{1}{2}\|x\|_{\mathbf{B}}^2 + \langle \lambda, \mathbf{A}x - b\rangle .$$

Taking the derivative with respect to x, setting to zero and isolating x gives

$$x^* = -\mathbf{B}^{-1}\mathbf{A}^\top \lambda \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top. \tag{4.26}$$

Consequently $x^* - x^0 \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$. Assuming that $x^k - x^* \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$ holds, by induction we have that

$$x^{k+1} - x^* \overset{(4.6)}{=} x^k - x^* - \underbrace{\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k}(\mathbf{S}_{i_k}^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k})^\dagger \mathbf{S}_{i_k}^\top (\mathbf{A}x^k - b)}_{\in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top}. \tag{4.27}$$

Thus $x^{k+1} - x^*$ is the difference of two elements in the subspace $\operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$ and thus $x^{k+1} - x^* \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$. $\square$

We also make use of the following fact. For a symmetric positive semi-definite random matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ drawn from some probability distribution $\mathcal{D}$ and for any vector $v \in \mathbb{R}^n$

$$\mathbb{E}_{\mathcal{D}}\left[\|v\|_{\mathbf{M}}^2\right] = \mathbb{E}_{\mathcal{D}}\left[\langle v, \mathbf{M}v\rangle\right] = \langle v, \mathbb{E}_{\mathcal{D}}\left[\mathbf{M}v\right]\rangle = \|v\|_{\mathbb{E}_{\mathcal{D}}[\mathbf{M}]}^2. \tag{4.28}$$

4.7.1 Important spectral constants

We define two key spectral constants in the following definition that will be used to express our forthcoming rates of convergence.

Definition 4.7.2.

$$\sigma_\infty^2(\mathbf{B},\mathbf{S}) \overset{\text{def}}{=} \min_{v \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|_{\mathbf{Z}_i}^2}{\|v\|_{\mathbf{B}}^2}. \tag{4.29}$$

Let $p \in \Delta_q$ and let

$$\sigma_p^2(\mathbf{B},\mathbf{S}) \overset{\text{def}}{=} \min_{v \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top} \frac{\|\mathbf{B}^{1/2}v\|_{\mathbb{E}_{i\sim p}[\mathbf{Z}_i]}^2}{\|v\|_{\mathbf{B}}^2}. \tag{4.30}$$

Next we show that $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ and $\sigma_p^2(\mathbf{B},\mathbf{S})$ can be used to lower bound $\max_i f_i(x)$ and $\mathbb{E}_{i\sim p}\left[f_i(x)\right]$, respectively. This result will allow us to develop Equation (4.23) and Lemma 4.5.3 into a recurrence later on.

Lemma 4.7.3. *Let $p \in \Delta_q$ and consider the iterates x^k given by Algorithm 3 when using any adaptive sampling rule. The spectral constants Equation* (4.29) *and Equation* (4.30) *are such that*

$$\max_{i=1,\ldots,q} f_i(x^k) \quad \geq \quad \sigma_\infty^2(\mathbf{B},\mathbf{S})\|x^k - x^*\|_{\mathbf{B}}^2, \tag{4.31}$$

$$\mathbb{E}_{i\sim p}\left[f_i(x^k)\right] \quad \geq \quad \sigma_p^2(\mathbf{B},\mathbf{S})\|x^k - x^*\|_{\mathbf{B}}^2. \tag{4.32}$$

Proof. From the invariance provided by Lemma 4.7.1 we have that $x^k - x^* \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$ and consequently

$$\begin{aligned} \frac{\max_{i=1,\ldots,q} f_i(x^k)}{\|x^k - x^*\|_{\mathbf{B}}^2} &\overset{(4.19)}{=} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}(x^k - x^*)\|_{\mathbf{Z}_i}^2}{\|x^k - x^*\|_{\mathbf{B}}^2} \\ &\geq \min_{v \in \operatorname{range}\mathbf{B}^{-1}\mathbf{A}^\top} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|_{\mathbf{Z}_i}}{\|v\|_{\mathbf{B}}^2} \overset{(4.29)}{=} \sigma_\infty^2(\mathbf{B},\mathbf{S}), \quad \forall k. \end{aligned} \tag{4.33}$$

Analogously we have that

$$\begin{aligned} \frac{\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]}{\|x^k - x^*\|_{\mathbf{B}}^2} &\overset{(4.19)}{=} \frac{\mathbb{E}_{i\sim p}\left[\|\mathbf{B}^{1/2}(x^k - x^*)\|_{\mathbf{Z}_i}^2\right]}{\|x^k - x^*\|_{\mathbf{B}}^2} \\ &\geq \min_{v \in \operatorname{range}\mathbf{B}^{-1}\mathbf{A}^\top} \frac{\mathbb{E}_{i\sim p}\left[\|\mathbf{B}^{1/2}v\|_{\mathbf{Z}_i}^2\right]}{\|v\|_{\mathbf{B}}^2} \overset{(4.30)+(4.28)}{=} \sigma_p^2(\mathbf{B},\mathbf{S}). \end{aligned} \tag{4.34}$$

Thus Equation (4.31) and Equation (4.32) follow by re-arranging Equation (4.33) and Equation (4.34) respectively. □

Finally, we show that $\sigma_p^2(\mathbf{B},\mathbf{S})$ and $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ are always less than one, and if the exactness Assumption 4.4.1 holds then they are both strictly greater than zero.

Lemma 4.7.4. *Let $p \in \Delta_q$ and the set of sketching matrices $\{\mathbf{S}_1, \ldots, \mathbf{S}_q\}$ be such that that exactness Assumption 4.4.1 holds. We then have the following relations:*

$$0 < \sigma_p^2(\mathbf{B},\mathbf{S}) \quad = \quad \lambda_{\min}^+\left(\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]\right) \quad \leq \quad \sigma_\infty^2(\mathbf{B},\mathbf{S}) \leq 1.$$

Proof. Using the definition of $\mathbf{Z}_i$ given in Equation (4.16) and the fact that $\mathbf{B}$ is symmetric positive definite, we have

$$\begin{aligned} \operatorname{Null}\left(\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]\right) &\overset{(4.16)}{=} \operatorname{Null}\left(\mathbf{B}^{-1/2}\mathbf{A}^\top\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}\mathbf{B}^{-1/2}\right) \\ &= \operatorname{Null}\left(\mathbf{A}^\top\mathbb{E}_{i\sim p}\left[\mathbf{H}_i\right]\mathbf{A}\mathbf{B}^{-1/2}\right) \overset{\textit{Lemma } 4.B.1}{=} \operatorname{Null}\left(\mathbf{A}\mathbf{B}^{-1/2}\right), \end{aligned}$$

where we applied Lemma 4.B.1 in the appendix with $\mathbf{G} = \mathbb{E}_{i\sim p}[\mathbf{H}_i]$ and $\mathbf{W} = \mathbf{A}$. Taking the orthogonal complement of the above we have that

$$\operatorname{range} \mathbb{E}_{i\sim p}[\mathbf{Z}_i] = \operatorname{range} \mathbf{B}^{-1/2}\mathbf{A}^\top. \tag{4.35}$$

Using the above we then have

$$\begin{aligned}
\sigma_p^2(\mathbf{B},\mathbf{S}) &\overset{(4.30)}{=} \min_{v\in\operatorname{range}\mathbf{B}^{-1}\mathbf{A}^\top} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbb{E}_{i\sim p}[\mathbf{Z}_i]}}{\|v\|^2_{\mathbf{B}}} \\
&\overset{(4.35)}{=} \min_{\mathbf{B}^{1/2}v\in\operatorname{range}\mathbb{E}_{i\sim p}[\mathbf{Z}_i]} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbb{E}_{i\sim p}[\mathbf{Z}_i]}}{\|v\|^2_{\mathbf{B}}} \;=\; \lambda^+_{\min}\left(\mathbb{E}_{i\sim p}[\mathbf{Z}_i]\right) > 0.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\sigma_p^2(\mathbf{B},\mathbf{S}) &\overset{(4.30)}{=} \min_{v\in\operatorname{range}\mathbf{B}^{-1}\mathbf{A}^\top} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbb{E}_{i\sim p}[\mathbf{Z}_i]}}{\|v\|^2_{\mathbf{B}}} \\
&\overset{(4.28)}{=} \min_{v\in\operatorname{range}\mathbf{B}^{-1}\mathbf{A}^\top} \frac{\mathbb{E}_{i\sim p}\left[\|\mathbf{B}^{1/2}v\|^2_{\mathbf{Z}_i}\right]}{\|v\|^2_{\mathbf{B}}} \\
&\leq \min_{v\in\operatorname{range}\mathbf{B}^{-1}\mathbf{A}^\top} \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{Z}_i}}{\|v\|^2_{\mathbf{B}}} \;=\; \sigma_\infty^2(\mathbf{B},\mathbf{S}).
\end{aligned}$$

Finally, using the fact that the matrix $\mathbf{Z}_i$ is an orthogonal projection (Lemma 4.5.1), we have that

$$\sigma_\infty^2(\mathbf{B},\mathbf{S}) \;=\; \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|^2_{\mathbf{Z}_i}}{\|v\|^2_{\mathbf{B}}} \overset{(4.17)}{=} \max_{i=1,\ldots,q} \frac{\|\mathbf{Z}_i\mathbf{B}^{1/2}v\|^2}{\|\mathbf{B}^{1/2}v\|^2} \;\leq\; \max_{i=1,\ldots,q} \frac{\|\mathbf{B}^{1/2}v\|^2}{\|\mathbf{B}^{1/2}v\|^2} = 1.$$

□

4.7.2 Sampling from a fixed distribution

We first present a convergence result for the sketch-and-project method when the sketches are drawn from a fixed sampling distribution. This result will later be used as a baseline for comparison against the adaptive sampling strategies.

Theorem 4.7.5. *Consider Algorithm 2 for some set of probabilities $p \in \Delta_q$. It follows that*

$$\mathbb{E}\|x^k - x^*\|^2_{\mathbf{B}} \leq \left(1 - \sigma_p^2(\mathbf{B},\mathbf{S})\right)^k \|x^0 - x^*\|^2_{\mathbf{B}}.$$

Proof. Combining Lemma 4.5.3 and Equation (4.32) of Lemma 4.7.3 we have that

$$\begin{aligned}\mathbb{E}_{i_k\sim p}\left[\|x^{k+1}-x^*\|_{\mathbf{B}}^2 \,|\, x^k\right] &\overset{\textit{Lemma } 4.5.3}{=} \|x^k-x^*\|_{\mathbf{B}}^2 - \mathbb{E}_{i_k\sim p}\left[f_i(x^k)\right] \\ &\overset{(4.32)}{\leq} \left(1-\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^*\|_{\mathbf{B}}^2.\end{aligned}$$

Taking the full expectation and unrolling the recurrence, we arrive at Theorem 4.7.5. □

There are several natural and previously studied choices for fixed sampling distributions, for example, sampling the indices uniformly at random. Another choice is to pick $p \in \Delta_q$ in order to maximize $\sigma_p^2(\mathbf{B},\mathbf{S})$, but this results in a convex semi-definite program (see Section 5.1 in [GR15b]). The authors of [GR15b] suggest convenient probabilities such that $p_i \sim \|\mathbf{A}^\top \mathbf{S}_i\|_{\mathbf{B}^{-1}}^2$ for which $\sigma_p^2(\mathbf{B},\mathbf{S})$ reduces to the scaled condition number.

4.7.3 Max-distance selection

The following theorem provides a convergence guarantee for the max-distance selection rule of Subsection 4.6.3. To our knowledge, this is the first analysis of the max-distance rule for general sketch-and-project methods.

Theorem 4.7.6. *The iterates of max-distance sketch-and-project method in Algorithm 4 satisfy*

$$\|x^k-x^*\|_{\mathbf{B}}^2 \leq (1-\sigma_\infty^2(\mathbf{B},\mathbf{S}))^k\|x^0-x^*\|_{\mathbf{B}}^2,$$

where $\sigma_\infty(\mathbf{B},\mathbf{S})$ is defined as in Equation (4.29) of Definition 4.7.2.

Proof. Combining Equation (4.23) and Equation (4.31) we have that

$$\begin{aligned}\|x^{k+1}-x^*\|_{\mathbf{B}}^2 &\overset{(4.23)}{=} \|x^k-x^*\|_{\mathbf{B}}^2 - \max_{i=1,\dots,q} f_i(x^k) \\ &\overset{(4.31)}{\leq} \left(1-\sigma_\infty^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^*\|_{\mathbf{B}}^2.\end{aligned}$$

Unrolling the recurrence gives Theorem 4.7.6. □

One obvious disadvantage of sampling from a fixed distribution is that it is possible to sample the same index twice in a row. Since the current iterate already lies in the solution space with respect to the previous sketch, no progress is made in such an update. For adaptive distributions that only assign non-zero probabilities to non-zero sketched loss values, the same index will never be chosen twice in a row since the sketched loss corresponding to the previous iterate will always be zero (Lemma 4.7.7). This fact allows us to derive convergence rates for adaptive sampling strategies that are strictly better than those for fixed sampling strategies.

Lemma 4.7.7. *Consider the sketched losses $f(x^k)$ generated by iterating the sketch-and-project update given in Equation* (4.6)*. We have that*

$$f_{i_k}(x^{k+1}) = 0, \quad \forall\, k \geq 0.$$

Proof. Recall from Equation (4.19), we can write

$$f_{i_k}(x^{k+1}) = \|\mathbf{B}^{1/2}(x^{k+1} - x^*)\|^2_{\mathbf{Z}_{i_k}} = \left\langle \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^{k+1} - x^*), \mathbf{B}^{1/2}(x^{k+1} - x^*)\right\rangle. \tag{4.36}$$

We can show that the above is equal to zero by using Equation (4.18) and Lemma 4.5.1 we have that

$$\begin{aligned}
\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^{k+1} - x^*) &\overset{(4.18)}{=} \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - \mathbf{B}^{-1/2}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*) - x^*) \\
&= \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*) - \mathbf{Z}_{i_k}\mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*)) \\
&\overset{(4.17)}{=} \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*) - \mathbf{Z}_{i_k}\mathbf{B}^{1/2}(x^k - x^*)) \\
&= 0.
\end{aligned}$$

□

We now use Lemma 4.7.7 to additionally show that the convergence guarantee for the greedy method is strictly faster than for sampling with respect to any set of fixed probabilities.

Theorem 4.7.8. *Let $p \in \Delta_q$ where $p_i > 0$ for all $i = 1, \ldots, q$. Let $\sigma_p^2(\mathbf{B}, \mathbf{S})$ be defined as in Equation* (4.30) *of Definition 4.7.2 and define*

$$\gamma \overset{def}{=} \frac{1}{\max_{i=1,\ldots,q} \sum_{j=1,\, j\neq i}^{q} p_j} > 1. \tag{4.37}$$

We then have that the max-distance sketch-and-project method of Algorithm 4 satisfies the following convergence guarantee

$$\|x^{k+1}-x^*\|_{\mathbf{B}}^2 \le (1-\gamma\sigma_p^2(\mathbf{B},\mathbf{S}))\|x^k-x^*\|_{\mathbf{B}}^2. \tag{4.38}$$

Proof. Recall that $f_{i_k}(x^{k+1})=0$ by Lemma 4.7.7. Thus,

$$\begin{aligned}
\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right] &= \sum_{j=1,\, j\neq i_k}^{q} p_j f_j(x^{k+1}) \\
&\le \left(\max_{j=1,\ldots,q} f_j(x^{k+1})\right)\left(\sum_{j=1,\, j\neq i_k}^{q} p_j\right) \\
&\le \left(\max_{j=1,\ldots,q} f_j(x^{k+1})\right)\left(\max_{j=1,\ldots,q}\sum_{j=1,\, j\neq i}^{q} p_j\right) \\
&\overset{(4.37)}{=} \frac{\max_{j=1,\ldots,q} f_j(x^{k+1})}{\gamma}.
\end{aligned} \tag{4.39}$$

From Equation (4.23) we have that

$$\begin{aligned}
\|x^{k+1}-x^*\|_{\mathbf{B}}^2 &\overset{(4.23)}{=} \|x^k-x^*\|_{\mathbf{B}}^2 - \max_{i=1,\ldots,q} f_i(x^k) \\
&\overset{(4.39)}{\le} \|x^k-x^*\|_{\mathbf{B}}^2 - \gamma\mathbb{E}_{i\sim p}\left[f_i(x^k)\right] \\
&\overset{(4.32)}{\le} \left(1-\gamma\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^*\|_{\mathbf{B}}^2.
\end{aligned}$$

□

4.7.4 The proportional adaptive rule

We now consider the adaptive sampling strategy in which indices are sampled with probabilities proportional to the sketched loss values. For this sampling strategy, we derive a convergence rate that is at least twice as fast as that of Theorem 4.7.5 for uniform sampling.

Theorem 4.7.9. *Consider Algorithm 3 with* $p^k = \frac{f(x^k)}{\|f(x^k)\|_1}$. *Let* $u = \left(\frac{1}{q},\ldots,\frac{1}{q}\right)\in\Delta_q$ *and* $\sigma_u^2(\mathbf{B},\mathbf{S})$ *be as defined in Equation* (4.30). *Let* $\mathbb{VAR}_u[\cdot]$ *denote the variance taken with respect to the uniform distribution over indices* $i\in\{1,\ldots,q\}$. *It follows that for* $k\ge 1$,

$$\mathbb{E}\|x^{k+1}-x^*\|_{\mathbf{B}}^2 \,|\, x^k \le \left(1-(1+q^2\mathbb{VAR}_u\left[p_i^k\right])\sigma_u^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^*\|_{\mathbf{B}}^2. \tag{4.40}$$

Furthermore we have that

$$\mathbb{E}\|x^{k+1}-x^*\|^2_{\mathbf{B}} \leq \left(1-2\sigma_u^2(\mathbf{B},\mathbf{S})\right)^k \mathbb{E}\|x^1-x^*\|^2_{\mathbf{B}}. \tag{4.41}$$

Proof. Let $\mathbb{E}_u[\cdot]$ denote the expectation taken with respect to the uniform distribution over indices $i \in \{1,\ldots,q\}$. First note that

$$\mathbb{VAR}_u\left[f_i(x^k)\right] = \mathbb{E}_u\left[(f_i(x^k))^2\right] - \mathbb{E}_u\left[f_i(x^k)\right]^2 = \frac{1}{q}\sum (f_i(x^k))^2 - \frac{1}{q^2}\left(\sum f_i(x^k)\right)^2. \tag{4.42}$$

Given that $p^k = \frac{f(x^k)}{\|f(x^k)\|_1}$,

$$\begin{aligned}
\mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right] &= \sum_{i=1}^q p_i^k f_i(x^k) \\
&= \sum_{i=1}^q \frac{(f_i(x^k))^2}{\sum_{i=1}^q f_i(x^k)} \\
&\overset{(4.42)}{=} \frac{q\mathbb{VAR}_u\left[f_i(x^k)\right] + \frac{1}{q}\left(\sum f_i(x^k)\right)^2}{\sum_{i=1}^q f_i(x^k)} \\
&= \left(q^2\mathbb{VAR}_u\left[\frac{f_i(x^k)}{\sum_{i=1}^q f_i(x^k)}\right]+1\right)\frac{1}{q}\sum_{i=1}^q f_i(x^k). \qquad (4.43)
\end{aligned}$$

Recalling that $p_i^k = \frac{f_i(x^k)}{\sum_{i=1}^q f_i(x^k)}$ and using Lemma 4.5.3 we have that

$$\mathbb{E}\|x^{k+1}-x^*\|^2_{\mathbf{B}} \,|\, x^k \leq \|x^k-x^*\|^2_{\mathbf{B}} - (1+q^2\mathbb{VAR}_u\left[p_i^k\right])\sigma_u^2(\mathbf{B},\mathbf{S})\|x^k-x^*\|^2_{\mathbf{B}}.$$

Furthermore, due to Lemma 4.7.7 we have that $p_{i_k}^{k+1}=0$. Therefore

$$\begin{aligned}
\mathbb{VAR}_u\left[p_i^{k+1}\right] &= \frac{1}{q}\sum_{i=1}^q\left(p_i^{k+1} - \frac{1}{q}\sum_{s=1}^q p_s^{k+1}\right)^2 \\
&= \frac{1}{q}\sum_{i=1}^q\left(p_i^{k+1}-\frac{1}{q}\right)^2 \geq \frac{1}{q}\left(p_{i_k}^{k+1}-\frac{1}{q}\right)^2 = \frac{1}{q^2}.
\end{aligned}$$

This lower bound on the variance gives the following upper bound on Equation (4.40)

$$\mathbb{E}\|x^{k+1}-x^*\|^2_{\mathbf{B}} \,|\, x^k \leq \left(1-2\sigma_u^2(\mathbf{B},\mathbf{S})\right)\|x^k-x^*\|^2_{\mathbf{B}}.$$

Taking the expectation and unrolling the recursion gives Equation (4.41). □

Thus by sampling proportional to the sketched losses the sketch-and-project method enjoys a strictly faster convergence rate as compared to sampling uniformly. How much faster depends on the variance of the adaptive probabilities through $1 + q^2 \mathbb{VAR}_u \left[p_i^k\right]$ which in turn depends on the variance of the sketched losses.

This same variance term is used in [PCJ17] to analyze the convergence of an adaptive sampling strategy based on the dual residuals for coordinate descent applied to regularized loss functions and in [OAL16] for adaptive sampling in the block-coordinate Frank-Wolfe algorithm for optimizing structured support vector machines.

4.7.5 Capped adaptive sampling

We now extend the capped adaptive sampling method and convergence guarantees of [BW18a, BW18b, BW19a] for the randomized Kaczmarz and coordinate descent settings to the general sketch-and-project setting, see Algorithm 5. Let $p \in \Delta_q$ be a fixed reference probability. At each iteration k an index set $\mathcal{W}_k$ is constructed on line 4 of Algorithm 5 that contains indices whose sketched losses are sufficiently close to the maximal sketched loss and that are at least as large as $\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]$. At each iteration, the adaptive probabilities p_i^k are zero for all indices that are not included in the set $\mathcal{W}_k$. The input parameter $\theta \in [0,\,1]$ controls how aggressive the sampling method is. In particular, if $\theta = 1$, the method reduces to max-distance sampling. As θ approaches 0, the sampling method remains adaptive, as only indices corresponding to sketched losses larger than $\mathbb{E}_{i\sim p}\left[f_i(x^k)\right]$ are sampled with non-zero probability. Bai and Wu originally introduced an adaptive randomized Kaczmarz method with $\theta = 1/2$ [BW18a] and generalized this to allow for the more general choice of $\theta \in [0,\,1]$ [BW18b].

Algorithm 5 generalizes and improves upon the methods proposed in [BW18a, BW18b, BW19a] in several ways. We generalize the methods from the randomized Kaczmarz setting to the more general sketch-and-project setting. We additionally allow for the use of any fixed reference probability distribution $p \in \Delta_q$, whereas the methods of [BW18a, BW18b, BW19a] use a specific reference probability when identifying the set of indices that will be selected with nonzero probability. Lastly, we allow for the use of any adaptive sampling strategy such that the probabilities p_i^k are zero outside of the set $\mathcal{W}_k$ whereas the methods proposed in [BW18a,BW18b,BW19a] specify that a specific adaptive probability be used. However, this restriction is unnecessary in proving the accompanying convergence result Theorem 4.7.11.

Below, we provide two convergence guarantees for Algorithm 5. Theorem 4.7.10 provides a convergence guarantee in terms of the spectral constants $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ and $\sigma_p^2(\mathbf{B},\mathbf{S})$ of Definition 4.7.2 and the parameter θ. Theorem 4.7.11 provides a generalization of the convergence rate derived in [BW18b].

Algorithm 5 Capped Adaptive Sketch-and-Project

```
1: input: $x^0 \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $p \in \Delta_q$, $\theta \in [0,\, 1]$ and a set of sketching matrices
   $\{\mathbf{S}_1, \ldots, \mathbf{S}_q\}$
2: for $k = 0, 1, 2, \ldots$ do
3:     $f_i(x^k) = \|\mathbf{A}x^k - b\|^2_{\mathbf{H}_i}$ for $i = 1, \ldots, q$.
4:     $\mathcal{W}_k = \left\{ i \mid f_i(x^k) \geq \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]\right\}$
5:     Choose $p^k \in \Delta_q$ such that support$(p^k) \subset \mathcal{W}_k$
6:     $i_k \sim p^k$
7:     $x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{H}_{i_k}(\mathbf{A}x^k - b)$
8: end for
9: output: last iterate $x^{k+1}$
```

Theorem 4.7.10. *Consider Algorithm 5. Let* $p \in \Delta_q$ *be a fixed* reference probability *and* $\theta \in [0,1]$. *Let*

$$\mathcal{W}_k = \left\{ i \mid f_i(x^k) \geq \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]\right\}. \tag{4.44}$$

It follows that

$$\mathbb{E}\|x^k - x^*\|^2_{\mathbf{B}} \leq \left(1 - \theta\sigma_\infty^2(\mathbf{B},\mathbf{S}) - (1-\theta)\sigma_p^2(\mathbf{B},\mathbf{S})\right)^k \|x^0 - x^*\|^2_{\mathbf{B}}. \tag{4.45}$$

Proof. First note that $\mathcal{W}_k$ is not empty since

$$\max_{j=1,\ldots,q} f_j(x^k) \geq \mathbb{E}_{j\sim p}\left[f_j(x^k)\right],$$

and thus $\arg\max_{j=1,\ldots,q} f_j(x^k) \in \mathcal{W}_k$. Since $p_i^k = 0$ for all $i \notin \mathcal{W}_k$, Lemma 4.5.3 gives that

$$\mathbb{E}_{i\sim p^k}\left[\|x^{k+1} - x^*\|_{\mathbf{B}}^2 \,|\, x^k\right] = \|x^{k+1} - x^*\|_{\mathbf{B}}^2 - \sum_{i\in\mathcal{W}_k} p_i^k f_i(x^k). \tag{4.46}$$

We additionally have

$$\begin{aligned}\sum_{i\in\mathcal{W}_k} f_i(x^k)p_i^k &\overset{(4.44)}{\geq} \sum_{i\in\mathcal{W}_k}\left(\theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]\right)p_i^k \\ &= \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right] \qquad (4.47)\\ &\overset{Lemma\ 4.7.3}{\geq} \left(\theta\sigma_\infty^2(\mathbf{B},\mathbf{S}) + (1-\theta)\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^k - x^*\|_{\mathbf{B}}^2. \qquad (4.48)\end{aligned}$$

Using Equation (4.48) to bound Equation (4.46) and taking the expectation gives the result. □

The resulting convergence rate is a convex combination of the spectral constant $\sigma_\infty^2(\mathbf{B},\mathbf{S})$ which corresponds to the max-distance convergence rate guarantee and $\sigma_p^2(\mathbf{B},\mathbf{S})$ corresponding to the convergence rate guarantee for the fixed reference probabilities p. This convex combination is in terms of the parameter θ and we can see that as θ approaches 1 the method and convergence guarantee approach that of max-distance. When θ is close to 0, the convergence guarantee approaches that of a fixed distribution, but still filters out sketches with sketched losses less than $\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]$. This suggests that for $\theta \approx 0$ the convergence rate guarantee is loose.

We now explicitly extend the analysis of Bai and Wu's work of [BW18b, BW18b, BW19a] to derive a convergence rate guarantee for our more general Algorithm 5.

Theorem 4.7.11. *Consider Algorithm 5. Let $p \in \Delta_q$ be a set of fixed* reference probabilities *and $\theta \in [0,1]$. Let*

$$\gamma \overset{def}{=} \frac{1}{\max_{i=1,\ldots,q}\sum_{j=1,\,j\neq i}^q p_j} > 1. \tag{4.49}$$

It follows for $k \geq 1$

$$\begin{aligned}&\mathbb{E}\|x^k - x^*\|_{\mathbf{B}}^2 \qquad (4.50)\\ &\leq \left(1 - (\theta\gamma + (1-\theta))\,\sigma_p^2(\mathbf{B},\mathbf{S})\right)^{k-1}\left(1 - \theta\sigma_\infty^2(\mathbf{B},\mathbf{S}) - (1-\theta)\sigma_p^2(\mathbf{B},\mathbf{S})\right)\|x^0 - x^*\|_{\mathbf{B}}^2,\end{aligned}$$

where the expectation is taken with respect to the probabilities prescribed by Algorithm 5.

Proof. By Lemma 4.7.7, at least one of the sketched losses is guaranteed to be zero for each iterations $k \geq 1$. Making the conservative assumption that this sketched loss corresponds to the smallest probability $\hat{p}^k_{i_k}$, we have, by Equation (4.39), that for an adaptive sampling strategy that assigns $p^k_i = 0$ to sketches $\mathbf{S}_i$ with a sketched loss $f_i(x^k) = 0$ that

$$\frac{\max_{j=1,\dots,q} f_j(x^{k+1})}{\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right]} \geq \gamma. \tag{4.51}$$

Combining this with Equation (4.47),

$$\begin{aligned}
\sum_{i\in\mathcal{W}_k} f_i(x^{k+1})p^{k+1}_i \quad &\geq \quad \left(\theta\frac{\max_{j=1,\dots,q} f_j(x^{k+1})}{\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right]} + (1-\theta)\right)\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right] \\
&\overset{(4.51)}{\geq} \quad (\theta\gamma + (1-\theta))\,\mathbb{E}_{j\sim p}\left[f_j(x^{k+1})\right] \\
&\overset{(4.30)}{\geq} \quad (\theta\gamma + (1-\theta))\,\sigma^2_p(\mathbf{B},\mathbf{S}).
\end{aligned} \tag{4.52}$$

Consequently for $k \geq 1$, by Equation (4.46), we then have

$$\mathbb{E}\|x^{k+1} - x^*\|^2_{\mathbf{B}} \,|\, x^k \leq \|x^k - x^*\|^2_{\mathbf{B}} - (\theta\gamma + (1-\theta))\,\sigma^2_p(\mathbf{B},\mathbf{S})\|x^k - x^*\|^2_{\mathbf{B}}.$$

Taking the expectation and unrolling the recursion gives,

$$\mathbb{E}\|x^{k+1} - x^*\|^2_{\mathbf{B}} \leq \left(1 - (\theta\gamma + (1-\theta))\,\sigma^2_p(\mathbf{B},\mathbf{S})\right)^{k-1}\|x^1 - x^*\|^2_{\mathbf{B}}.$$

Since, at the very first update, we cannot guarantee that there exists $i \in [1,\dots,q]$ such that $f_i(x^0) = 0$, Equation (4.52) is not guaranteed for $k = 0$. So instead we use Equation (4.45) to unroll the last step in this recurrence to arrive Equation (4.50). $\square$

The convergence rate for Algorithm 5 of Theorem 4.7.11 is an improvement over the convergence rate guarantee for a fixed probability distribution since $\gamma > 1$. As was the case for Theorem 4.7.10, the convergence rate is maximized when $\theta = 1$, at which point the resulting method is equivalent to the max-distance sampling strategy of Algorithm 4. Further, when $\theta = 1$, Theorem 4.7.11 guarantees

$$\mathbb{E}\|x^k - x^*\|^2_{\mathbf{B}} \leq \left(1 - \gamma\sigma^2_p(\mathbf{B},\mathbf{S})\right)^{k-1}\left(1 - \sigma^2_\infty(\mathbf{B},\mathbf{S})\right)\|x^0 - x^*\|^2_{\mathbf{B}}.$$

For $\theta = 0$, Theorem 4.7.11 recovers the same convergence guarantee as for sampling according to the non-adaptive probabilities p.

Sampling Strategy	**Convergence Rate Bound**	**Rate Bound Shown In**
Fixed, $p_i^k \equiv p_i$	$1 - \sigma_p^2(\mathbf{B}, \mathbf{S})$	[GR15b], Theorem 4.7.5
Max-distance	$1 - \sigma_\infty^2(\mathbf{B}, \mathbf{S})$	Theorem 4.7.6
$p_i^k \propto f_i(x^k)$	$1 - 2\sigma_u^2(\mathbf{B}, \mathbf{S})$	Theorem 4.7.9
Capped	$1 - (1+\epsilon)\,\sigma_p^2(\mathbf{B}, \mathbf{S})$	Theorem 4.7.11

Table 4.1: Summary of convergence guarantees of Section 4.7, where $\gamma = 1/\max_{i=1,\dots,q}\sum_{j=1,j\neq i}^q p_i$

4.8 Implementation tricks and computational complexity

One can perform adaptive sketching with the same order of cost per iteration as the standard non-adaptive sketch-and-project method when τq, the number of sketches q times the sketch size τ, is not significantly larger than the number of columns n. In particular, adaptive sketching methods can be performed for a per-iteration cost of $O(\tau^2 q + \tau n)$, whereas the standard non-adaptive sketch-and-project method has a per-iteration cost of $O(\tau n)$. Section 4.A discusses the costs of adaptive sketch-and-project methods in more detail. Pseudocode for efficient implementation is provided in Algorithm 6.

The main computational costs of adaptive sketch-and-project (Algorithm 3) at each iteration come from computing the sketched losses $f_i(x^k)$ of Equation (4.8) and updating the iterate from x^k to x^{k+1} via Equation (4.6). The iterate update for x^k and the formula for the sketched loss $f_i(x^k) = \|\mathbf{A}x - b\|_{\mathbf{H}_i}^2$ both require calculating what we call the *sketched residual*,

$$\mathbf{R}_i^k \overset{\text{def}}{=} \mathbf{C}_i^\top \mathbf{S}_i^\top (\mathbf{A}x^k - b), \tag{4.53}$$

where $\mathbf{C}_i$ is any square matrix satisfying $\mathbf{C}_i\mathbf{C}_i^\top = (\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger$. The adaptive methods considered here require the sketched residual $\mathbf{R}_i^k$ for each sketch index $i = 1, 2, \dots, q$ at each iteration. For such adaptive methods, it is possible to update the iterate x^k and compute the sketched losses $f_i(x^k)$ more efficiently if one maintains the set of sketched residuals $\{\mathbf{R}_i^k : i = 1, 2, \dots, q\}$ in memory.

Different sampling strategies require different amounts of computation as well. Among the adaptive sampling strategies considered here, max-distance sampling requires the least amount of computation followed by sampling proportional to the sketched losses. Capped adaptive sampling requires the most computation. The costs for each sampling strategy are discussed in detail in Subsection 4.A.3 and are summarized in Table 4.7.

4.9 Summary of consequences for special cases

We now discuss the consequences of the convergence analyses of Section 4.7 and the computational costs detailed in Section 4.8 for the special sketch-and-project subcases of randomized Kaczmarz and coordinate descent. For $\mathbf{C}_i$ as defined in Equation (4.54), in both the randomized Kaczmarz method and coordinate descent, $\mathbf{C}_i$ is a scalar and thus its value is fixed.

4.9.1 Adaptive Kaczmarz

By choosing the parameter matrix $\mathbf{B} = \mathbf{I}$ and sketching matrices $\mathbf{S}_i = e_i$ for $i = 1, \dots, m$ where $e_i \in \mathbb{R}^n$ is the i^{th} coordinate vector, we arrive at the Kaczmarz method introduced in Subsection 4.1.1. For randomized Kaczmarz, the sketches $\mathbf{S}_i = e_i$ isolate a single row of the matrix $\mathbf{A}$, as $\mathbf{S}_i^\top \mathbf{A} = \mathbf{A}_{i:}$. In this setting, the number of sketches $q = m$ for $\mathbf{A} \in \mathbb{R}^m$, and the sketch size is $\tau = 1$. In order to perform the adaptive update efficiently, the matrices

$$\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i \mathbf{C}_i = \frac{\mathbf{A}_{i:}^\top}{\|\mathbf{A}_{i:}\|} \quad \text{and} \quad \mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_j \mathbf{C}_j = \frac{\langle \mathbf{A}_{i:}, \mathbf{A}_{j:} \rangle}{\|\mathbf{A}_{i:}\| \|\mathbf{A}_{j:}\|} \; \forall \, i, j = 1, 2, \dots m$$

should be precomputed.

In order to succinctly express the convergence rates, we define the diagonal probability matrix $\mathbf{P} = \operatorname{diag}(p_1, \dots, p_m)$ and the normalized matrix $\bar{\mathbf{A}} \overset{\text{def}}{=} \mathbf{D}_{RK}^{-1}\mathbf{A}$, with $\mathbf{D}_{RK} \overset{\text{def}}{=} \operatorname{diag}\left(\|\mathbf{A}_{1:}\|_2, \dots, \|\mathbf{A}_{m:}\|_2\right)$ as in [NSL16]. In the randomized Kaczmarz setting, the projection matrix $\mathbf{Z}_i$ as defined in Equation (4.16) is the orthogonal projection onto the i^{th} row of $\mathbf{A}$ and takes the form

$$\mathbf{Z}_i = \frac{\mathbf{A}_{i:}\mathbf{A}_{i:}^\top}{\|\mathbf{A}_{i:}^2\|}.$$

We then have

$$\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right] = \mathbf{D}_{RK}^{-1}\mathbf{A}\mathbf{P}\mathbf{A}^\top\mathbf{D}_{RK}^{-1} = \bar{\mathbf{A}}^\top\mathbf{P}\bar{\mathbf{A}}.$$

The costs and convergence rates for the adaptive sampling strategies discussed in Section 4.6 applied to the Kaczmarz method are summarized in Table 4.2, where we used the notation $\|x\|_\infty \overset{\text{def}}{=} \max_i |x_i|$ for any vector x.

Sampling Strategy	**Convergence Rate Bound**	**Rate Bound Shown In**	**Flops Per Iteration**
Uniform	$1 - \frac{1}{m}\lambda_{\min}^{+}(\bar{\mathbf{A}}^\top\bar{\mathbf{A}})$	[NSL16], Theorem 4.7.5	$2\min(n,m) + 2n$
$p_i \propto \|\mathbf{A}_{i:}\|_2^2$	$1 - \frac{\lambda_{\min}^{+}(\mathbf{A}^\top\mathbf{A})}{\|\mathbf{A}\|_F^2}$	[SV09], Theorem 4.7.5	$2\min(n,m) + 2n$
Max-distance	$1 - \min\limits_{v\in\operatorname{range}\mathbf{A}^\top} \frac{\|\bar{\mathbf{A}}v\|_\infty}{\|v\|_2}$	[NSL16], Theorem 4.7.6	$3m + 2n$
$p_i^k \propto f_i(x^k)$	$1 - \frac{2}{m}\lambda_{\min}^{+}(\bar{\mathbf{A}}^\top\bar{\mathbf{A}})$	Theorem 4.7.9	$5m + 2n$
Capped	$1 - (\theta\gamma + 1)\,\lambda_{\min}^{+}(\bar{\mathbf{A}}^\top\mathbf{P}\bar{\mathbf{A}})$	[BW18b], Theorem 4.7.11	$9m + 2n$

Table 4.2: Summary of convergence guarantees and costs of various sampling strategies for the randomized Kaczmarz algorithm. Here, $\gamma = 1/\max_{i=1,\ldots,m}\sum_{j=1,j\neq i}^{m} p_i$ as defined in Equation (4.37), $\mathbf{P} = \operatorname{diag}(p_1, \ldots, p_m)$ is a matrix of arbitrary fixed probabilities, and $\bar{\mathbf{A}} := \mathbf{D}_{RK}^{-1}\mathbf{A}$, with $\mathbf{D}_{RK} := \operatorname{diag}\left(\|\mathbf{A}_{1:}\|_2, \ldots, \|\mathbf{A}_{m:}\|_2\right)$. Only leading order flop counts are reported. The number of sketches is q, the sketch size is τ and the number of rows and columns in the matrix $\mathbf{A}$ are m and n respectively.

4.9.2 Adaptive coordinate descent

By choosing the parameter matrix $\mathbf{B} = \mathbf{A}^\top\mathbf{A}$ and sketching matrices $\mathbf{S}_i = \mathbf{A}e_i$ for $i = 1, \ldots, n$ where $e_i \in \mathbb{R}^m$ is the i^{th} coordinate vector, we arrive at the coordinate descent method introduced in Subsection 4.1.2. In this setting, the number of sketches $q = n$, where n is number of columns in $\mathbf{A}$, and the sketch size is $\tau = 1$.

Coordinate descent uses fewer flops per iteration than indicated by the general computation given in Subsection 4.A.1. This computational savings arises from the sparsity of the matrix

$\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}\mathbf{C}_{i_k} = e_i/\|\mathbf{A}_{:i}\|$. As a result, the iterate update of x^k to x^{k+1} using the sketched residuals $\mathbf{R}_{i_k}^k$ requires only $O(1)$ flops instead of $2n$ flops as indicated in the general analysis that is summarized in Table 4.5. The cost of a coordinate descent update is dominated by the $2n$ flops required to calculate $\mathbf{R}_{i_k}^k$ by either the auxiliary update of Line 10 of Algorithm 6 or directly via Equation (4.53).

Similar to the randomized Kaczmarz case, we define the diagonal probability matrix $\mathbf{P} \overset{\text{def}}{=} \operatorname{diag}(p_1, \ldots, p_n)$ and the normalized matrix $\widetilde{\mathbf{A}} \overset{\text{def}}{=} \mathbf{A}\mathbf{D}_{CD}^{-1}$, with $\mathbf{D}_{CD} \overset{\text{def}}{=} \operatorname{diag}\left(\|\mathbf{A}_{:1}\|_2, \ldots, \|\mathbf{A}_{:n}\|_2\right)$. The projection matrix $\mathbf{Z}_i$ as defined in Equation (4.16) is the projection given by

$$\mathbf{Z}_i = (\mathbf{A}^\top\mathbf{A})^{-1/2}\mathbf{A}^\top\mathbf{A}\frac{e_i e_i^\top}{\|\mathbf{A}_{:i}\|^2}\mathbf{A}^\top\mathbf{A}(\mathbf{A}^\top\mathbf{A})^{-1/2} = (\mathbf{A}^\top\mathbf{A})^{1/2}\frac{e_i e_i^\top}{\|\mathbf{A}_{:i}\|^2}(\mathbf{A}^\top\mathbf{A})^{1/2}.$$

We then have

$$\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right] = (\mathbf{A}^\top\mathbf{A})^{1/2}\mathbf{D}_{CD}^{-1}\mathbf{P}\mathbf{D}_{CD}^{-1}(\mathbf{A}^\top\mathbf{A})^{1/2}.$$

Note that $\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]$ is similar to $\mathbf{P}\mathbf{D}_{CD}^{-1}\mathbf{A}^\top\mathbf{A}\mathbf{D}_{CD}^{-1} = \mathbf{P}\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}$ and thus

$$\lambda_{\min}^+(\mathbb{E}_{i\sim p}\left[\mathbf{Z}_i\right]) = \lambda_{\min}^+(\mathbf{P}\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}).$$

The costs and convergence rates for the adaptive sampling strategies discussed in Section 4.6 applied to coordinate descent are summarized in Table 4.3.

4.10 Experiments

We test the performance of various adaptive and non-adaptive sampling strategies in the special sketch-and-project subcases of randomized Kaczmarz and coordinate descent. We report performance via three different metrics: norm-squared error versus iteration, norm-squared error versus approximate flop count, and the worst expected convergence factor.

Results are averaged over 50 trials. For each trial a single matrix $\mathbf{A}$ is used. For the experiments measuring error, a single true solution x^* and vector b are used. The worst expected convergence factor aims to approximate the spectral constants of Definition 4.7.2. Since the max-distance method is deterministic, generating a new exact solution x^* adds more variation between trials, hopefully leading to a more accurate approximation of the

Sampling	**Convergence Rate Bound**	**Rate Bound Shown In**	**Flops Per Iteration**
Uniform	$1 - \frac{1}{n}\lambda_{\min}^{+}(\widetilde{\mathbf{A}}^\top \widetilde{\mathbf{A}})$	Theorem 4.7.5	$2n$
$p_i \propto \|\mathbf{A}_{:i}\|_2^2$	$\left(1 - \frac{\lambda_{\min}^{+}(\mathbf{A}^\top \mathbf{A})}{\|\mathbf{A}\|_F^2}\right)$	[LL10] Theorem 4.7.5	$2n$
Max-distance	$1 - \min\limits_{v \in \operatorname{range} \mathbf{A}^\top} \frac{\|\widetilde{\mathbf{A}}v\|_\infty}{\|v\|_2}.$	Theorem 4.7.6	$3n$
$p_i^k \propto f_i(x^k)$	$1 - \frac{2}{n}\lambda_{\min}^{+}(\widetilde{\mathbf{A}}^\top \widetilde{\mathbf{A}})$	Theorem 4.7.9	$5n$
Capped	$1 - (\theta\gamma + 1)\,\lambda_{\min}^{+}(\mathbf{P}\widetilde{\mathbf{A}}^\top \widetilde{\mathbf{A}})$	Theorem 4.7.11	$9n$

Table 4.3: Summary of convergence guarantees and costs of various sampling strategies for adaptive coordinate descent. Here, $\gamma = 1/\max_{i=1,\ldots,m} \sum_{j=1, j\neq i}^{n} p_i$ as defined in Equation (4.37), $\mathbf{P} = \operatorname{diag}(p_1, \ldots, p_n)$ is a matrix of arbitrary fixed probabilities, and $\widetilde{\mathbf{A}} = \mathbf{A}\mathbf{D}_{CD}^{-1}$, with $\mathbf{D}_{CD} = \operatorname{diag}\left(\|\mathbf{A}_{:1}\|_2, \ldots, \|\mathbf{A}_{:n}\|_2\right)$. Only flop counts of leading order are reported.

spectral constant. The exact solutions x^* are generated by

$$x^* = \frac{\mathbf{A}^\top \omega}{\|\mathbf{A}^\top \omega\|_{\mathbf{B}}},$$

where $\omega \in \mathbb{R}^m$ is a vector of i.i.d. random normal entries. Thus $\|x^*\|_{\mathbf{B}}^2 = 1$ is normalized with respect to the $\mathbf{B}$–norm and lies in the row space of $\mathbf{A}$. The latter condition guarantees that x^* is indeed the unique solution to Equation (4.1). We measure the error in terms of the $\mathbf{B}$-norm. Recall that for randomized Kaczmarz $\mathbf{B} = \mathbf{I}$, while for coordinate descent, $\mathbf{B} = \mathbf{A}^\top \mathbf{A}$. The sketch-and-project methods are implemented using the auxiliary update Line 10 of Algorithm 6. For the max-distance sampling rule, if multiple sketches achieve the maximal sketched-loss value, we select the first such sketch.

We consider synthetic matrices of size 1000×100 and 100×1000 that are generated with i.i.d. standard Gaussian entries. We additionally test the various adaptive sampling strategies on two large-scale matrices arising from real world problems. These matrices are available via the SuiteSparse Matrix Collection [DH11]. The first system (Ash958) is an overdetermined matrix with 958 rows, 292 columns, and 1916 entries [DGL89, DGL92]. The matrix comes from a survey of the United Kingdom and is part of the original Harwell sparse matrix test collection. The second real matrix we consider is the GEMAT1 matrix, which arises from optimal power flow modeling. This matrix is highly underdetermined and consists of 4929 rows, 10,595 columns, and 47,369 entries [DGL89, DGL92].

4.10.1 Error per iteration

We first investigate the convergence of the squared norm of the error, $\|x^k - x^*\|_{\mathbf{B}}^2$ in terms of the number of iterations, see Figure 4.2. The first row of subfigures (Figures 4.2a and 4.2b) shows convergence for randomized Kaczmarz, while the second row of subfigures (Figures 4.2c and 4.2d) gives the convergence of various sampling strategies for coordinate descent. The first column of subfigures (Figures 4.2a and 4.2c) uses an underdetermined system of 100×1000 while the second column of subfigures (Figures 4.2b and 4.2d) considers an overdetermined system of 1000×100. Figures 4.4c and 4.4d demonstrate convergence per iteration for the Ash958 matrix and Figures 4.5a and 4.5c for randomized Kaczmarz and coordinate descent applied to the GEMAT1 matrix.

As expected, we see that the max-distance sampling strategy performs the best per iteration followed by the capped adaptive strategy, then sampling proportional to the sketched residuals and finally followed by the uniform strategy. For randomized Kaczmarz applied to underdetermined systems and coordinate descent applied to overdetermined systems, max-distance and the capped adaptive sampling strategies perform similarly in terms of squared error per iteration. The convergence of randomized Kaczmarz for each sampling strategy applied to overdetermined systems is very similar to that of coordinate descent applied to underdetermined systems. Similarly, the convergence of randomized Kaczmarz for each sampling strategy applied to underdetermined systems is very similar to that of coordinate descent applied to overdetermined systems. For the large and underdetermined GEMAT1 matrix, we find that randomized coordinate descent methods have much larger variance in their performance compared to randomized Kaczmarz methods.

4.10.2 Error versus approximate flops required

If we take into account the number of flops required for each method, the relative performance of the methods changes significantly. In order to approximate the number of flops required for each sampling strategy, we use the leading order flop counts per iteration given in Tables 4.2 and 4.3. We do not consider the precomputational costs, but only the costs incurred at each iteration. The performance in terms of flops of each sampling strategy is reported in Figure 4.3. Performance on the Ash958 matrix is reported in Figures 4.4c and 4.4d. Performance on the GEMAT1 matrix for randomized Kaczmarz and coordinate descent is reported in Figures 4.5b and 4.5d.

As discussed in Section 4.8, the adaptive methods are typically more expensive than non-adaptive methods as one must update the sketched residuals $\mathbf{R}_i^k$ for $i = 1, \dots, q$ at each iteration k. Yet even after taking flops into consideration, we find that the max-distance sampling strategy still performs the best overall. For randomized Kaczmarz applied to an overdetermined synthetic matrix, uniform sampling performance is comparable to max-distance (Figure 4.3b). In all other experiments, however, max-distance sampling is the clear winner. Since max-distance sampling performs at least as well per iteration as capped adaptive sampling and sampling with probabilities proportional to the sketched losses, yet the max-distance sampling method is less expensive, it naturally performs the best among the adaptive methods when flop counts are considered.

4.10.3 Spectral constant estimates

Theorems 4.7.5, 4.7.6, and 4.7.8 to 4.7.11 of Section 4.7 provide conservative views of the convergence rates of each method, as the spectral constants of Definition 4.7.2 give the expected convergence corresponding to the worst possible point $x \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}$ as opposed to the iterates x^k. In practice, the convergence at each iteration might perform better than the convergence bounds indicate.

Recall that the convergence rates derived in Section 4.7 are given in terms of spectral constants (Definition 4.7.2) of the form

$$\sigma_p^2(\mathbf{B}, \mathbf{S}) \overset{\text{def}}{=} \min_{x \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top} \frac{\mathbb{E}_{i\sim p}\left[f_i(x)\right]}{\|x - x^\star\|_{\mathbf{B}}^2}.$$

We will refer to the value

$$\frac{\mathbb{E}_{i\sim p^k}\left[f_i(x^k)\right]}{\|x^k - x^\star\|_{\mathbf{B}}^2}$$

as the *expected step size factor* and note that larger values indicate superior performance.

The smallest expected step size factor observed for each method provides an estimate and upper bound on the spectral constants in the derived convergence rates. The minimal expected step size factor for each sampling method applied to random Gaussian matrices of size 1000×100 and 100×1000 are reported in Table 4.4. As expected, we find that these values increase from uniform sampling, sampling proportional to the sketched losses, capped adaptive sampling and finally max-distance selection. In Theorem 4.7.9, we proved a bound on the convergence rate for sampling proportional to the sketched losses that was twice as fast as the convergence guarantee for uniform sampling. We find that the estimated spectral constants in Table 4.4 for the proportional sampling strategy is also at least twice as large as the estimated spectral constant for uniform sampling.

Sampling	**Randomized Kaczmarz**		**Coordinate Descent**	
	1000×100	100×1000	1000×100	100×1000
Uniform	0.00705	0.00667	0.00656	0.00715
$p_i \propto \|\mathbf{A}_{:i}\|_2^2$	0.02019	0.01569	0.01722	0.02014
Capped	0.03885	0.01901	0.01952	0.03878
Max-distance	0.04593	0.01994	0.02171	0.04711

Table 4.4: Minimal expected step size factor for each sampling method applied to matrices containing i.i.d. guassian entries.

4.11 Conclusions

We extend adaptive sampling methods to the general sketch-and-project setting. We present a computationally efficient method for implementing the adaptive sampling strategies using an auxiliary update. For several specific adaptive sampling strategies including max-distance selection, the capped adaptive sampling of [BW18a,BW18b,BW19a], and sampling proportional to the sketched residuals, we derive convergence rates and show that the greedy max-distance sampling rule has the fastest convergence guarantee among the sampling methods considered. This superior performance is seen in practice as well for both the randomized Kaczmarz and coordinate descent subcases.

4.A Implementation tricks and computational complexity

We describe how one can perform adaptive sketching with the same order of cost per iteration as the standard non-adaptive sketch-and-project method when τq, the number of sketches q times the sketch size τ, is not significantly larger than the number of columns n. In particular, we show how adaptive sketching methods can be performed for a per-iteration cost of $O(\tau^2 q+\tau n)$, whereas the standard non-adaptive sketch-and-project method has a per-iteration cost of $O(\tau n)$. The precomputations and efficient update strategies presented here are a generalization of those suggested in [BW18a] for the Kaczmarz setting. Precomputational costs are a one time expense and are independent of the sampling strategy. The precomputational costs depend on the sparsity structure of the sketches and are summarized for randomized Kaczmarz and coordinate descent in Table 4.6. The computational costs given in this section may be over-estimates of the costs required for specific sketch choices such as when the update is sparse, as is the case in coordinate descent. The special cases of adaptive Kaczmarz and adaptive coordinate descent are analyzed in Section 4.9.

Pseudocode for efficient implementation is provided in Algorithm 6. Throughout this section, we will frequently omit $O(1)$ and $O(\log(q))$ flop counts since they are insignificant compared to the number of rows m, the number of columns n, and the number of sketches q.

4.A.1 Per-iteration cost

The main computational costs of adaptive sketch-and-project (Algorithm 3) at each iteration come from computing the sketched losses $f_i(x^k)$ of Equation (4.8) and updating the iterate from x^k to x^{k+1} via Equation (4.6). We now discuss how these steps can be calculated efficiently. A suggested efficient implementation for adaptive sketch-and-project is provided in Algorithm 6. The costs of each step of an iteration of the adaptive sketch-and-project method are summarized in Table 4.5.

Let $\mathbf{C}_i$ be any square matrix satisfying

$$\mathbf{C}_i\mathbf{C}_i^\top = (\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i)^\dagger. \tag{4.54}$$

For example, $\mathbf{C}_i$ could be the Cholesky decomposition of $(\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i)^\dagger$. The sketched loss $f_i(x^k)$ and the iterate update from x^k to x^{k+1} can now be written as

$$f_i(x^k) = \|\mathbf{S}_i^\top(\mathbf{A}x^k - b)\|^2_{\mathbf{C}_i\mathbf{C}_i^\top} = \|\mathbf{C}_i^\top\mathbf{S}_i^\top(\mathbf{A}x^k - b)\|_2^2$$

and

$$x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}\mathbf{C}_{i_k}\mathbf{C}_{i_k}^\top\mathbf{S}_{i_k}^\top(\mathbf{A}x^k - b).$$

Notice that both the iterate update for x^k and the formula for the sketched loss $f_i(x^k)$ share the sketched residual $\mathbf{R}_i^k \overset{\text{def}}{=} \mathbf{C}_i^\top\mathbf{S}_i^\top(\mathbf{A}x^k - b)$ defined in Equation (4.53). In adaptive methods one must compute the sketched residual $\mathbf{R}_i^k$ for $i = 1, 2, \ldots, q$. When sampling from a fixed distribution, however, calculating the sketched losses $f_i(x^k)$ is unnecessary and only the sketched residual $\mathbf{R}_{i_k}^k$ corresponding to the selected index i_k need be computed.

Depending on the sketching matrices $\mathbf{S}_i$ and the matrix $\mathbf{B}$, it is possible to update the iterate x^k and compute the sketched losses $f_i(x^k)$ more efficiently if one maintains the set of sketched residuals $\{\mathbf{R}_i^k : i = 1, 2, \ldots, q\}$ in memory. Using the sketched residuals, the calculations above can be rewritten as

$$f_i(x^k) = \|\mathbf{R}_i^k\|_2^2 \tag{4.55}$$

and

$$x^{k+1} = x^k - \mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_{i_k}\mathbf{C}_{i_k}\mathbf{R}_{i_k}^k. \tag{4.56}$$

The sketched residuals $\{\mathbf{R}_i^k : i = 1, 2, \ldots, q\}$ can either be computed via an auxiliary update applied to the set of previous set of sketched residuals $\{\mathbf{R}_i^{k-1} : i = 1, 2, \ldots, q\}$ or directly using the iterate x^k. Using the auxiliary update,

$$\begin{aligned}\mathbf{R}_i^{k+1} &= \mathbf{C}_i^\top \mathbf{S}_i^\top (\mathbf{A}x^{k+1} - b) \\ &= \mathbf{C}_i^\top \mathbf{S}_i^\top \left(\mathbf{A}(x^k - \mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k}\mathbf{C}_{i_k}\mathbf{R}_{i_k}^k) - b\right) \\ &= \mathbf{R}_i^k - \mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k}\mathbf{C}_{i_k}\mathbf{R}_{i_k}^k \qquad (4.57)\end{aligned}$$

with the initialization

$$\mathbf{R}_i^0 = \mathbf{C}_i^\top \left(\mathbf{S}_i^\top (\mathbf{A}x^0 - b)\right).$$

If the matrix $\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_j \mathbf{C}_j \in \mathbb{R}^{\tau\times\tau}$ is precomputed for each $i, j = 1, 2, \ldots, q$, the sketched residual $\mathbf{R}_i^k$ can be updated to $\mathbf{R}_i^{k+1}$ for $2\tau^2$ flops for each index i via Equation (4.57). Using the precomputed matrices requires storing $\frac{1}{4}\tau(\tau+1)q(q+1)$ floats.

In the non-adaptive case, one only needs to compute the single sketched residual $\mathbf{R}_{i_k}^k$ as opposed to the entire set of sketched residuals, since the sketched losses $f_i(x^k)$ are not needed. If the matrices

$$\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A} \in \mathbb{R}^{\tau\times n} \quad \text{and} \qquad \mathbf{C}_i^\top \mathbf{S}_i^\top b \in \mathbb{R}^{\tau},$$

are precomputed for $i = 1, 2, \ldots, q$, computing each sketched residual $\mathbf{R}_i^k$ directly from the iterate x^k costs $2\tau n$ flops via Equation (4.53). If $q\tau > n$, then it is cheaper to compute the sketched residual $\mathbf{R}_{i_k}^k$ using the auxiliary update Equation (4.57) rather than computing it directly from x^k.

From the sketched residual $\mathbf{R}_i^k$, the sketched losses $f_i(x^k)$ can be computed for $2\tau - 1$ flops for each index i via Equation (4.55). If the matrix $\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i \mathbf{C}_i \in \mathbb{R}^{n\times\tau}$ is precomputed for each $i = 1, 2, \ldots, q$, the iterate x^k can then be updated to x^{k+1} for $2\tau n$ flops via Equation (4.56). These costs are summarized in Table 4.5.

4.A.2 Cost of sampling indices

The cost of computing the sampling probabilities p^k from the sketched losses $f_i(x^k)$ depends on the sampling strategy used. Sampling from a fixed distribution can be achieved with

Algorithm 6 Efficient Adaptive Sampling Sketch-and-Project

```
1: input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\{\mathbf{S}_i \in \mathbb{R}^{m\times\tau} : i = 1, 2, \dots, q\}$, $\mathbf{B} \in \mathbf{R}^{n\times n}$, $x^0 \in \operatorname{range} \mathbf{B}^{-1}\mathbf{A}^\top$,
2: compute $\mathbf{C}_i = \text{Cholesky}\left((\mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i)^\dagger\right)$ for $i = 1, 2, \dots, q$    ▷ The $\mathbf{C}_i$ can be discarded after Line 5.
3: compute $\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i \mathbf{C}_i \in \mathbb{R}^{n\times\tau}$ for $i = 1, 2, \dots, q$
4: compute $\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_j \mathbf{C}_j \in \mathbb{R}^{\tau\times\tau}$ for $i, j = 1, 2, \dots, q$
5: initialize $\mathbf{R}_i^0 = \mathbf{C}_i^\top \left(\mathbf{S}_i^\top(\mathbf{A}x^0 - b)\right) \in \mathbb{R}^\tau$ for $i = 1, 2, \dots, q$
6: for $k = 0, 1, 2, \dots$ do
7:     compute $f_i(x^k) = \|\mathbf{R}_i^k\|_2^2$ for $i = 1, 2, \dots, q$
8:     sample $i_k \sim p_i^k$, where $p^k \in \Delta_q$ is a function of $f(x^k)$
9:     update $x^{k+1} = x^k - (\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k} \mathbf{C}_{i_k})\mathbf{R}_{i_k}^k$
10:    update $\mathbf{R}_i^{k+1} = \mathbf{R}_i^k - (\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_{i_k} \mathbf{C}_{i_k})\mathbf{R}_{i_k}^k$ for $i = 1, 2, \dots, q$
11: end for
12: output: last iterate $x^{k+1}$
```

an $O(1)$ cost using precomputations of $O(q)$ [Wal74]. Adaptive strategies sample from a new, unseen distribution at each iteration, which can be achieved with an average of q flops using, for example, inversion by sequential search [Kem81, Dev86, p. 86]. In practice, the probabilities p_i^k corresponding to each index i are given by a function of the sketched losses $f(x_i^k)$ and normalizing these values is unnecessary. Instead, one can sum the q sketched losses and apply inversion by sequential search with a random value r generated between zero and the sum of these values. This summation requires $q-1$ flops. Thus, the total cost for sampling from an adaptive probability distribution for the methods considered is approximately $2q$ flops on average. The costs for the sampling strategies discussed in Section 4.6 are summarized in Table 4.7. The calculations of these costs are discussed in more detail in Subsection 4.A.3.

4.A.3 Sampling strategy specific costs

We now detail the calculations that lead to the costs associated with each of the specific sampling strategies that are reported in Table 4.7.

Per iteration computation	**Flops**
$f_i(x^k)\ \forall i$ via Equation (4.55)	$(2\tau - 1)q$
x^{k+1} via Equation (4.56)	$2\tau n$
$\mathbf{R}_i^k\ \forall i$ with auxiliary update, Equation (4.57)	$2\tau^2 q$
$\mathbf{R}_{i_k}^k$ via direct computation, Equation (4.53)	$2\tau n$

(a) Baseline flop counts. Flop counts of $O(1)$ have been omitted.

Stored Object	**Storage**
x^k	n
$\mathbf{R}_i^k \quad \forall i$	τq
$\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_i \mathbf{C}_i \quad \forall i$	$\tau q n$
$\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top \mathbf{S}_j \mathbf{C}_j$ $\forall i, j$	$\frac{1}{4}\tau(\tau+1)q(q+1)$
$\mathbf{C}_i^\top \mathbf{S}_i^\top \mathbf{A}$ and $\mathbf{C}_i^\top \mathbf{S}_i^\top b \quad \forall i$	$\tau q(n+1)$

(b) Storage costs.

Table 4.5: Summary of the costs of the of Algorithm 6 excluding costs that are specific to the sampling method. The number of sketches is q, the sketch size is τ and the number of columns in the matrix $\mathbf{A}$ is n.

4.A.3.1 Sampling from a fixed distribution

When sampling the indices i from a fixed distribution, computing the sketched losses $f_i(x^k)$ is unnecessary and only the sketched residual $\mathbf{R}_{i_k}^k$ of the selected index i_k is needed to update the iterate x^k. If $q\tau > n$, where q is the number of sketches, τ is the sketch size and n is the number of columns in the matrix $\mathbf{A}$, it is cheaper to compute the sketched residual $\mathbf{R}_{i_k}^k$ using the auxiliary update Equation (4.57) rather than computing it directly from x^k. Ignoring the $O(1)$ cost of sampling from the fixed distribution, the iterate update takes either $4\tau n$ flops if $q\tau > n$ and one maintains the set of sketched residuals via the auxiliary update Equation (4.57) or $2\tau(n+q)$ flops if the sketched residual $\mathbf{R}_{i_k}^k$ is calculated from the iterate x^k directly via Equation (4.53).

Computation	Randomized Kaczmarz		Coordinate Descent	
$\mathbf{C}_i$ of Equation (4.54)	$\frac{1}{\|\mathbf{A}_{i:}\|}$	$2mn + O(m)$	$\frac{1}{\|\mathbf{A}_{:i}\|}$	$2mn + O(n)$
$\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_i\mathbf{C}_i$	$\frac{\mathbf{A}_{i:}^\top}{\|\mathbf{A}_{i:}\|}$	mn	$\frac{e_i}{\|\mathbf{A}_{:i}\|}$	n
$\mathbf{C}_i^\top\mathbf{S}_i^\top\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^\top\mathbf{S}_j\mathbf{C}_j$	$\frac{\langle\mathbf{A}_{i:},\mathbf{A}_{j:}\rangle}{\|\mathbf{A}_{i:}\|\|\mathbf{A}_{j:}\|}$	m^2n $+O(m^2+mn)$	$\frac{\langle\mathbf{A}_{:i},\mathbf{A}_{:j}\rangle}{\|\mathbf{A}_{:i}\|\|\mathbf{A}_{:j}\|}$	mn^2 $+O(mn+n^2)$

Table 4.6: Precomputational costs for adaptive randomized Kaczmarz and adaptive coordinate descent. The computational costs assume the previous elements have been computed and give the cost of computing the value for all indices.

4.A.3.2 Max-distance selection

Performing max-distance selection requires finding the maximum element of the length q vector of sketched losses given in Equation (4.55). In the average case, this costs $q + O(\log q)$ flops, where q flops are used to check each element and $O(\log q)$ flops arise from updates to the running maximal value. For convenience, we ignore the $O(\log q)$ flops and consider the cost of the selection step using the max-distance rule to be q flops. If the sketches $\mathbf{S}_i$ are vectors, or equivalently we have $\tau = 1$, then the sketched residuals $\mathbf{R}_i^k$ are scalars and finding the maximal sketched loss $f_i(x^k)$ is equivalent to finding the sketched residual $\mathbf{R}_i^k$ of maximal magnitude. We can thus save q flops per iteration by skipping the step of computing the sketched losses and instead taking the sketched residual of maximal magnitude.

4.A.3.3 Sampling proportional to the sketched loss

Sampling indices with probabilities proportional to the sketched losses $f_i(x^k)$ requires approximately $2q$ flops on average using inversion by sequential search.

Sampling Strategy	**Non-Sampling Flops**	**Flops from Sampling**
Fixed, $p_i^k \equiv p_i \ \forall k$	$2\tau \min(n, \tau q) + 2\tau n$	$O(1)$
Max-distance	$(2\tau^2 + 2\tau - 1)q + 2\tau n$	q if $\tau > 1$ $O(\log(q))$ if $\tau = 1$
$p_i^k \propto f_i(x^k)$		$2q$
Capped		$6q$

Table 4.7: Rule-specific per-iteration costs of Algorithm 6. Only leading order flop counts are reported. The non-sampling flops are those that are independent of the specific adaptive sampling method used and are those that correspond to the steps indicated in Table 4.5a. The extra flops for sampling are those that are required to calculate the adaptive sampling probabilities p^k at each iteration. The number of sketches is q, the sketch size is τ and the number of columns in the matrix $\mathbf{A}$ is n.

4.A.3.4 Capped adaptive sampling

Recall that using capped adaptive sampling requires identifying the set

$$\mathcal{W}_k = \left\{ i \mid f_i(x^k) \geq \theta \max_{j=1,\ldots,q} f_j(x^k) + (1-\theta)\mathbb{E}_{j\sim p}\left[f_j(x^k)\right] \right\}.$$

Sampling with the capped adaptive sampling strategy requires identifying the set $\mathcal{W}_k$ and sampling an index from this set. Identifying the set $\mathcal{W}_k$ requires $q + O(\log q)$ flops to identify the maximal sketched loss $f_i(x^k)$, $2q$ flops to compute the weighted average of the sketched losses $\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]$, $O(1)$ flops to calculate the threshold for the set $\mathcal{W}_k$, and q flops to compare each sketched loss against the threshold. Sampling an index from the set $\mathcal{W}_k$ requires on average $2q$ flops by using inversion by sequential search as discussed in Subsection 4.A.2.[4] Thus, the total cost of the sampling step is $6q + O(\log q)$ flops. When a uniform average is used in place of the weighted average, the expected sketched loss $\mathbb{E}_{j\sim p}\left[f_j(x^k)\right]$ can be computed in just q flops as opposed to $2q$. In that case, the total cost of the sampling step is only $5q + O(\log q)$.

[4]The analyses of [BW18a, BW18b] omitted the cost of sampling the index from a new distribution at each iteration, and thus our cost calculations differ by $2q$.

Sampling Strategy	**Flops Per Iteration When $\tau > 1$**	**Flops Per Iteration When $\tau = 1$**
Fixed, $p_i^k \equiv p_i$	$2\tau \min(n, \tau q) + 2\tau n$	$2\min(n, q) + 2n$
Max-distance	$(2\tau^2 + 2\tau)q + 2\tau n$	$3q + 2n$
$p_i^k \propto f_i(x^k)$	$(2\tau^2 + 2\tau + 1)q + 2\tau n$	$5q + 2n$
Capped	$(2\tau^2 + 2\tau + 5)q + 2\tau n$	$9q + 2n$

Table 4.8: Summary of convergence guarantees of Section 4.7, where $\gamma = 1/\max_{i=1,\ldots,m}\sum_{j=1,j\neq i}^m p_i$ as defined in Equation (4.37) and $\epsilon = \theta(\gamma - 1) \leq \theta\frac{1}{m}$. Flop counts of $O(\log(q))$ have been omitted. Flop counts assume all matrices are dense. The number of sketches is q, the sketch size is τ and the number of columns in the matrix $\mathbf{A}$ is n.

4.B Auxiliary lemma

We now invoke a lemma taken from [GR16].

Lemma 4.B.1. *For any matrix $\mathbf{W}$ and symmetric positive semidefinite matrix $\mathbf{G}$ such that*

$$Null(\mathbf{G}) \subset Null\left(\mathbf{W}^\top\right), \tag{4.58}$$

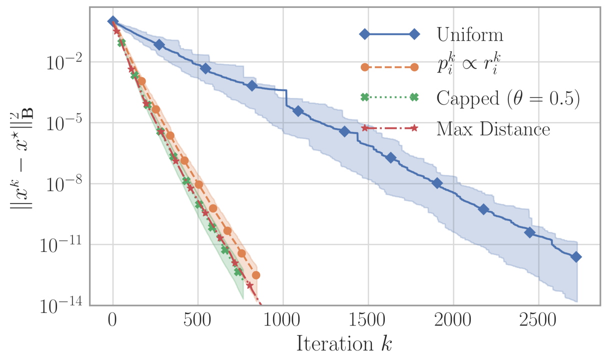
we have that

$$Null(\mathbf{W}) = Null\left(\mathbf{W}^\top \mathbf{G}\mathbf{W}\right). \tag{4.59}$$

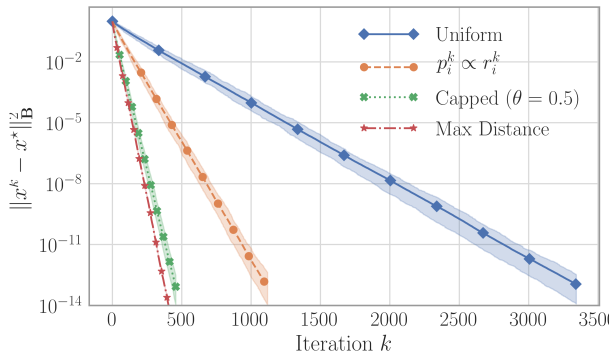
Proof. In order to establish Equation (4.59), it suffices to show the inclusion $\mathrm{Null}(\mathbf{W}) \supseteq \mathrm{Null}\left(\mathbf{W}^\top\mathbf{G}\mathbf{W}\right)$ since the reverse inclusion trivially holds. Letting $s \in \mathrm{Null}\left(\mathbf{W}^\top\mathbf{G}\mathbf{W}\right)$, we see that $\|\mathbf{G}^{1/2}\mathbf{W}s\|^2 = 0$, which implies $\mathbf{G}^{1/2}\mathbf{W}s = 0$. Consequently

$$\mathbf{W}s \in \mathrm{Null}\left(\mathbf{G}^{1/2}\right) = \mathrm{Null}\left(\mathbf{G}\right) \overset{(4.58)}{\subset} \mathrm{Null}\left(\mathbf{W}^\top\right).$$

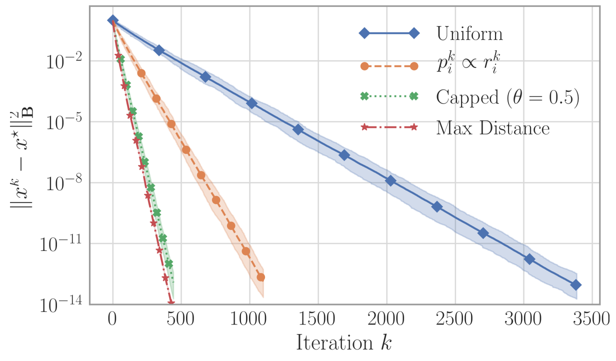
Thus $\mathbf{W}s \in \mathrm{Null}\left(\mathbf{W}^\top\right) \cap \mathrm{range}\,\mathbf{W}$ which are orthogonal complements which shows that $\mathbf{W}s = 0$. □ □



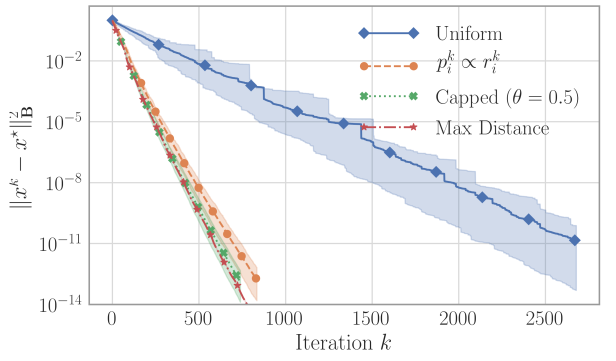
(a) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{100\times1000}$.



(b) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{1000\times100}$.

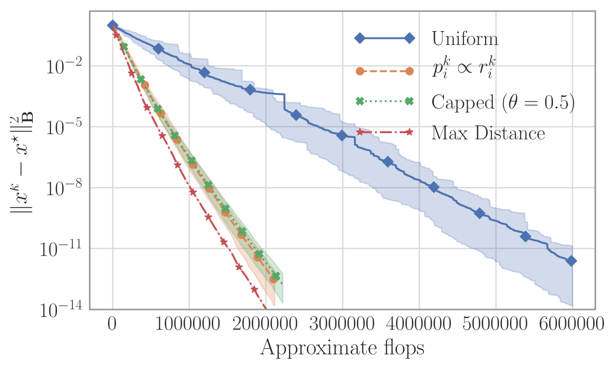


(c) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{100\times1000}$.

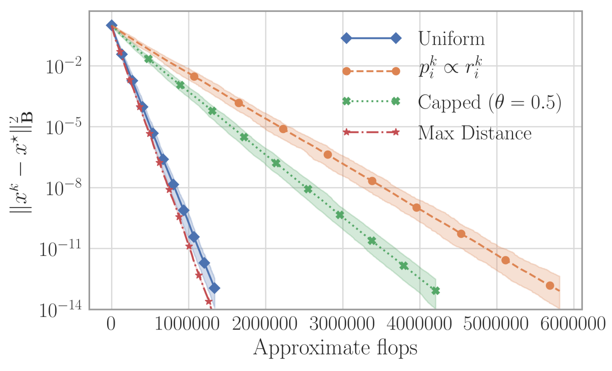


(d) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{1000\times100}$.

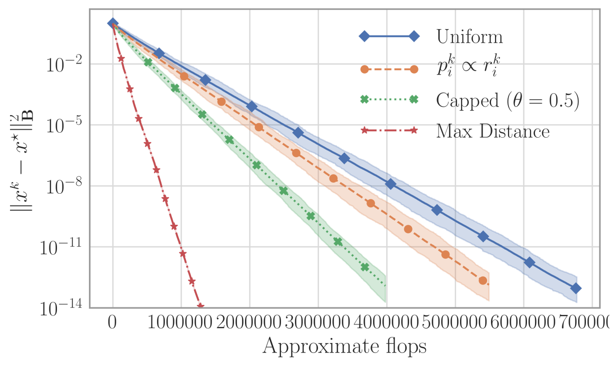
Figure 4.2: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent methods. Squared error norms were averaged over 50 trials. Confidence intervals indicate the middle 95% performance. Subplots on the left show convergence for underdetermined systems, while those on the right show the convergence on an overdetermined systems.



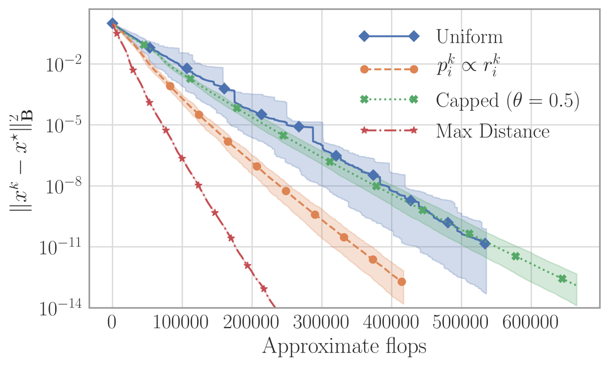
(a) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{100\times 1000}$.



(b) Adaptive randomized Kaczmarz, $\mathbf{A} \in \mathbb{R}^{1000\times 100}$.

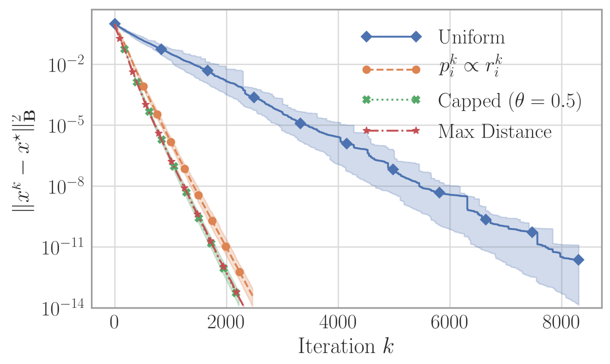


(c) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{100\times 1000}$.

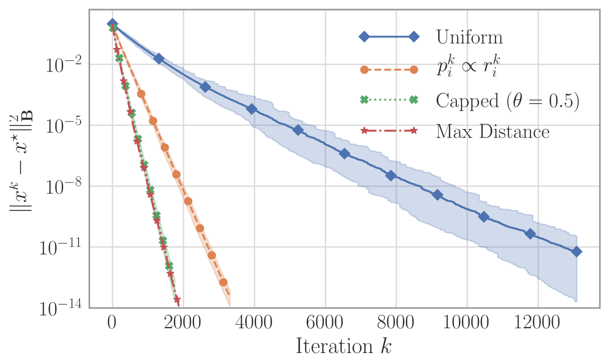


(d) Adaptive coordinate descent, $\mathbf{A} \in \mathbb{R}^{1000\times 100}$.

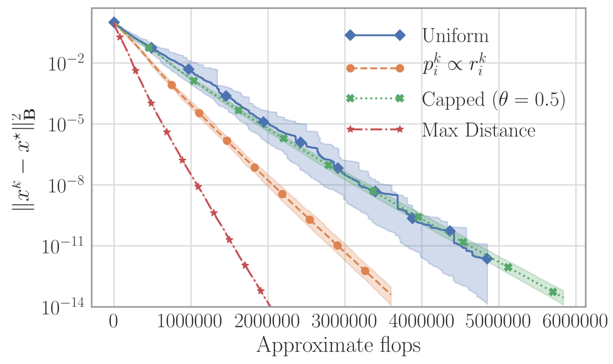
Figure 4.3: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent methods. Squared error norms were averaged over 50 trials and are plotted against the approximate flops aggregated over the computations that occur at each iteration. Confidence intervals indicate the middle 95% performance. Subplots on the left show convergence for underdetermined systems, while those on the right show the convergence on an overdetermined systems.



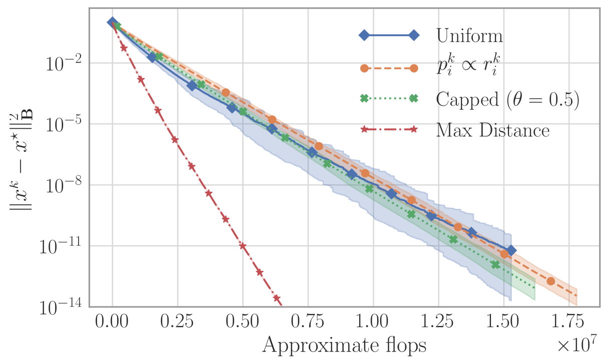
(a) Adaptive coordinate descent.



(b) Adaptive randomized Kaczmarz.

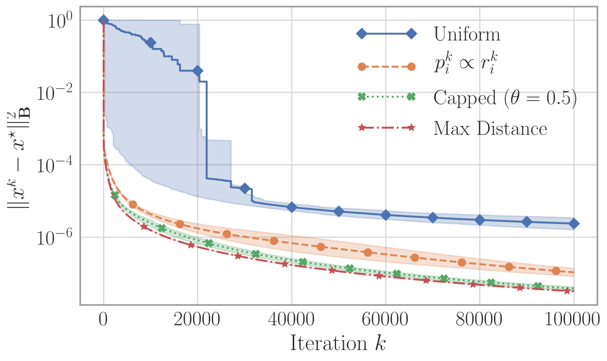


(c) Adaptive coordinate descent.

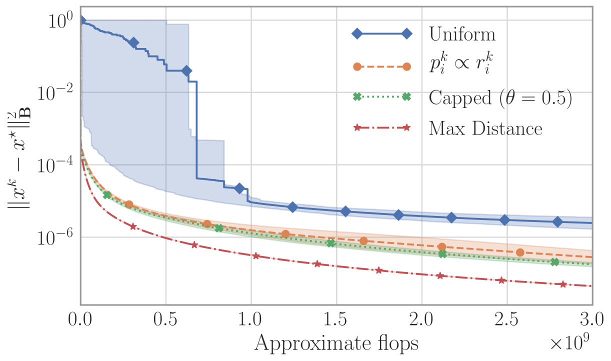


(d) Adaptive randomized kaczmarz.

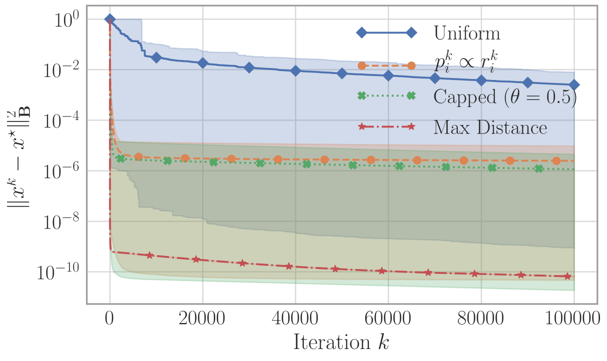
Figure 4.4: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent methods on the Ash958 matrix. Squared error norms were averaged over 50 trials and plotted against both the iteration and the approximate flops required. Confidence intervals indicate the middle 95% performance.



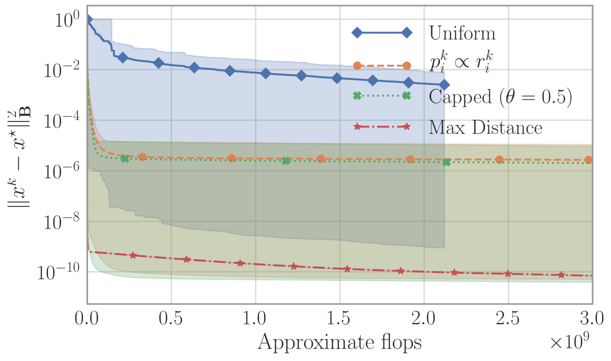
(a) Adaptive randomized Kaczmarz.



(b) Adaptive randomized Kaczmarz.



(c) Adaptive coordinate descent.



(d) Adaptive coordinate descent.

Figure 4.5: A comparison between different selection strategies for randomized Kaczmarz and coordinate descent on the GEMAT1 matrix. Squared error norms were averaged over 50 trials and plotted against both the iteration and the approximate flops required. Confidence intervals indicate the middle 95% performance.

CHAPTER 5

Classification of Binary Data with Hierarchical Class Structure

5.1 Introduction

We consider the problem of classification, where one is given a set of labeled data used for training, and from that data wishes to accurately assign labels to new unlabeled data. In the general problem, the class labels themselves have no relation to one another, however, data can often be organized in a hierarchical way. For example, in image classification problems, the data may contains images of inanimate and living objects. Then, within each of those classes the data may be further identified as images of vehicles and toys, or humans and animals. The data could then be further subdivided into classes of various animal types, and so on. This structure can be visualized as a *tree*, where the children of each node correspond to its sub-classes. Each data point in this case would have a label corresponding to a leaf of the tree, but also possesses the characteristics of all the labels of its ancestors. One option of course would be to simply use generic classification schemes to classify the data using the leaf labels only. *Hierarchical classification*, however, makes use of information and structure between groups in classifying the data [Gor87, SF11]. Extensions of popular classification methods such as the support vector machine (SVM) to the hierarchical setting are not straightforward, and such approaches often decompose the problem into many sub-problems leading to higher computational complexities [COL04, WW98].

Recently, [NSW17] proposed a simple classification scheme that uses only binary representations of data to perform classification; such representations arise naturally or are particularly efficient in many applications, see e.g. [FSL14, JLB11, ASS96, BB11, GNR10].

Here, we show that this method lends itself well to performing hierarchical classification and, in particular, using the hierarchical structure to improve computational efficiency. The classification method uses position of data relative to random hyperplanes to predict in which class a point is most likely to belong. [NSW17] demonstrated that for more complex data, using combinations of hyperplanes enables one to make more accurate predictions. However, the computation required to make a prediction scales exponentially in the number of hyperplane combinations used. Fortunately, the method is highly adjustable and for data that is likely to be more or less difficult to classify, one can adjust the number of these hyperplane combinations. Such a method is likely to be particularly useful for hierarchical data in which certain subclasses of data are more or less difficult to classify than others.

5.2 Underlying Classification Algorithm

In this section, we describe the classification algorithm proposed in [NSW17]. Let A be a random matrix in $\mathbb{R}^{m\times n}$, $X = [x_1\, x_2\, \cdots\, x_p] \in \mathbb{R}^{n\times p}$, where the $x_i \in \mathbb{R}^p$ are the data with labels $b = (b_1\, \cdots\, b_p)$ assigned from a possibility of G classes. Suppose we only have access to binary measurements of the data in the form $Q = \text{sign}(AX)$, where $\text{sign}(M)_{i,j} = \text{sign}(M_{i,j})$. The rows of A will correspond to (random) hyperplanes, and thus $Q_{i,j}$ simply captures on which side of the ith hyperplane the jth data point lies.

Let us build some intuition for the approach. Consider the two-dimensional data X shown in the top plot of Figure 5.1, consisting of three labeled classes (green, blue, red). Consider the four hyperplanes shown in the same plot, and suppose we had access only to the binary data $Q = \text{sign}(AX)$, where A contains the normals to each hyperplane as its rows. For the new test point x (which by visual inspection should be labeled blue) and its binary data $q = \text{sign}(Ax)$, one could simply cycle through the hyperplanes and decide which class x matches most often. For example, for the hyperplane colored purple in the plot, x has the same sign (i.e. lies on the same side) as the blue and green classes. For the black hyperplane, x only matches the blue class, and so on. Then for this example, x will clearly match the blue class most often, and we could assign it that label correctly. However, next consider the more complicated geometry given in the bottom plot, where the data consists of only two classes (red and blue), but they are now no longer linearly separable. This same strategy will no longer be accurate for the test point x. However, now instead of single hyperplanes, consider hyperplane *pairs*, and ask which class label x most often matches (note that in this context, by "matches" we now mean that points lie in the same cone into which the hyperplanes divide the space). For example, for the pair of hyperplanes colored orange and green, x matches both red and blue points, whereas for the pair of hyperplanes colored orange and purple, x only matches the blue class. One could now cycle through all pairs, and again ask which class x matches most often. For complicated data, we could aggregate such information across various *levels*, where at level l we consider l-tuples of hyperplanes in this way.

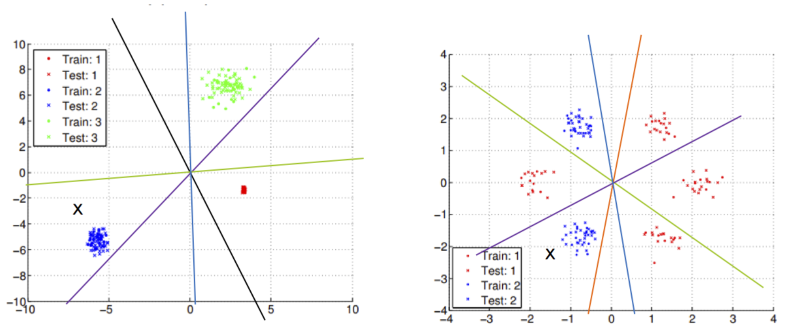


Figure 5.1: Two motivating examples for the classification method.

Let us now describe the method more formally. Consider m l-tuples of hyperplanes of various lengths $l = 1, \cdots, L$. Define the randomly selected index sets $\Lambda_{l,i}$, where $\Lambda_{l,i} \subset [m]$, $|\Lambda_{l,i}| = l$ and each $\Lambda_{l,i}$ is unique. If we then isolate the rows of Q contained in $\Lambda_{l,i}$ to form the $l \times p$ matrix $Q^{\Lambda_{l,i}}$, the columns of this matrix give the sign patterns of the data with respect to the hyperplanes in $\Lambda_{l,i}$. Let $T_{l,i}$ be the number of unique sign patterns, or equivalently columns of $Q^{\Lambda_{l,i}}$. Based on these sign patterns, we then calculate the membership index parameter $r(l,i,t,g)$ for each l-tuple $i = 1, \cdots, m$, level $l = 1, \cdots, L$, unique sign pattern $t = 1, \cdots, T_{l,i}$ and class $g = 1, \cdots, G$. Let $P_{g|t}$ be the number of data points in class g with sign pattern t and define:

$$r(l,i,t,g) := \frac{P_{g|t}}{\sum_{j=1}^{G} P_{j|t}} \frac{\sum_{j=1}^{G} |P_{g|t} - P_{j|t}|}{\sum_{j=1}^{G} P_{j|t}}. \tag{5.1}$$

The first fraction in (5.1) measures the proportion of data with sign pattern t that belong to class g, and the second fraction is a balancing term. This training process is also described in Algorithm 7.

Algorithm 7 Training from [NSW17]

```
**Input:** binary training data $Q$, training labels $b$, number of classes $G$, number of levels $L$.
for $l$ from 1 to $L$, $i$ from 1 to $m$ do
    Randomly select $\Lambda_{l,i} \subset [m], |\Lambda_{l,i}| = l$.
    Determine the $T_{l,i} \in \mathbb{N}$ unique column patterns in $Q^{\Lambda_{l,i}}$.
    for $t$ from 1 to $T_{l,i}$, $g$ from 1 to $G$ do
        Compute $r(l, i, t, g)$ as in Equation 5.1.
    end for
end for
```

Given $x \in \mathbb{R}^n$ and $q = \text{sign}(Ax)$, we predict the class of x as given in Algorithm 8. Intuitively, the membership values $r(l, i, t, g)$ indicate how likely a point with sign pattern t is to lie in class g, given information from the ith l-tuple of hyperplanes. These are aggregated over all measurements m and levels l, giving a likelihood that the point belongs to each class g. The label assigned is simply the class g corresponding to the largest value of $\tilde{r}$.

Algorithm 8 Classification from [NSW17]

```
Input: binary testing data $q$, number of classes $G$, number of levels $L$, learned parameters
$r(l, i, t, g)$, $T_{l,i}$, and $\Lambda_{l,i}$ from Algorithm 7.
for $l$ from 1 to $L$, $i$ from 1 to $m$ do
    Identify the pattern $t^* \in [T_{l,i}]$ to which $q^{\Lambda_{l,i}}$ corresponds.
    for $g$ from 1 to $G$ do
        $\tilde{r}(g) = \tilde{r}(g) + r(l, i, t^*, g)$.
    end for
end for
Set $\tilde{r}(g) = \frac{\tilde{r}(g)}{Lm}$ for $g = 1, \cdots, G$.
$\hat{b}_x = \text{argmax}_{g \in \{1, \cdots, G\}} \tilde{r}(g)$.
```

Flops	Operation
G	Initialize $\tilde{r}$
$m\sum_{l=1}^{L}\lvert T_{l,i}\rvert l$	Identify the sign pattern (worst case)
mGL	Update $\tilde{r}(g)$ for each class and level
$G+1$	Scale
G	Predict $\hat{b}_x = \operatorname{argmax}_{g\in\{1,\cdots,G\}}\tilde{r}(g)$

Table 5.1: Testing flop counts.

5.2.1 Computational Complexity

Given a data point $x \in \mathbb{R}^d$, we require

$$m(\sum_{l=1}^{L}|T_{l,i}|l + GL) + 3G + 1$$

flops to predict its class as described in Algorithm 8. Flop counts for each step in Algorithm 8 are given in Table 5.1. We do not include the cost of calculating $q = \text{sign}(Ax)$, as we assume that the algorithm is provided these binary measurements. Additionally, it may be the case that one does not have access to the underlying vector x and only knows the binary measurements q. As the number of levels increases, the term $m\sum_{l=1}^{L}|T_{l,i}|l$ typically dominates the testing cost. The number of possible sign patterns for a single measurement is 2^l and thus we at least have the bound $|T_{l,i}| \leq 2^l$. The inequality is strict if not all possible sign patterns are realized by points in the training data.

5.2.2 Adjustment for Hierarchical Classification

We now describe our proposed adjustment for handling hierarchical classification, where the labels possess some sort of tree structure. The classification scheme described above and in [NSW17] has the property that more levels (higher L) are needed to accurately classify more complicated data. Thus, if we know in advance that certain classes may require fewer levels for classification with sufficient accuracy, we may isolate these classes in an initial classification that uses fewer levels and then further classify these groups of classes using only the required number of levels for sufficient accuracy. This strategy leads to computational savings without sacrificing accuracy when some classes are more easily discerned from the others.

For illustration, consider the simple example where we have three classes, g_1, g_2, g_3. Suppose that L_1 levels are necessary to classify data belonging to g_1, but L_2 levels are required to differentiate between classes g_2 and g_3 where $L_2 > L_1$. We can perform binary classification between g_1 and $\{g_2, g_3\}$ using L_1 levels followed by classification between g_2 and g_3 using L_2 levels. These classifications can be organized as a tree with nodes H_1 and H_2 as shown in Figure 5.2. The sets S_1 and S_2 give the class groupings for the model constructed at nodes H_1 and H_2 respectively.

At test time, points initially classified as g_1 require only

$$m\left(\sum_{l=1}^{L_1} |T_{l,i}|l + GL_1\right) + 3G + 1$$

flops, where $G = 2$. In order to further discern between points predicted to belong to g_2 or g_3, we can use the same measurements (or random hyperplanes) as used in the first classification. Then for the two classifications, points initially classified as belonging to g_2 or g_3 require

$$m\left(\sum_{l=1}^{L_2} |T_{l,i}|l + \sum_{c=1}^{2} G_c L_c\right) + \sum_{c=1}^{2}(3G_c + 1)$$

flops to arrive at a prediction. Here, G_c is the number of groups at classification c and L_c is the number of levels used for classification c. In this particular example, we would have $G_1 = |S_1| = |\{g_1, \{g_2, g_3\}\}| = 2$ and $G_2 = |S_2| = |\{g_2, g_3\}| = 2$.

The overhead cost to carrying out two classifications instead of one is quite limited overall. For classifications in which some classes require fewer levels to predict, this hierarchical structure can lead to significant computational savings, as shown in the experimental results that follow. The magnitude of the computational savings is highly dependent on the distribution of the testing data, however, as we only reduce computational costs for those points predicted to be in one of the classes that is ‘easier’ to discern, i.e. requires fewer levels.

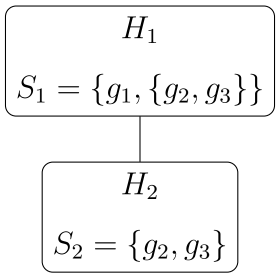


Figure 5.2: Hierarchical classification tree for a simple three-class example in which differentiating g_1 is significantly easier than differentiating g_2 and g_3.

The proposed hierarchical classification algorithm is further described in Algorithms 9 and 10.

Algorithm 9 Proposed adjustment for hierarchical classification (training).

```
**Input:** binary training data $Q$, training labels $b$, set of class groupings $S_c$ for each node $H_c$ in the tree of classifications $H$, number of levels $L = (L_1, \cdots L_C)$ to be used in each classification.
**for** $H_c \in H$, **do**
    Identify rows of $Q$ such that the corresponding component of $b$ is in one of the sets contained in $S_c$. Form a matrix $Q_c$ containing these rows.
    Define $\tilde{b}$ to be the labels indicating to which set of $S_c$ a given row of $Q_c$ corresponds.
    Train a classifier as in Algorithm 7 with training data $Q_c$, labels $\tilde{b}$, number of groups $|S_c|$ and number of levels $L_c$ as input.
**end for**
```

This hierarchical classification strategy naturally generalizes to incorporate more complicated and deeper hierarchical structures in which the classifications can be structured as a tree. See Figure 5.3 for an example. In order to maximize computational gains, however, we would like the tree to be 'imbalanced' in terms of the maximum number of levels required for sufficient classification accuracy along different paths of the tree. Such an imbalance arises naturally in many applications. For example, consider brain imaging and the problem of detecting brain abnormalities including tumors and dementia; tumor detection is a fairly easy learning problem whereas classifying differing types of dementia remains very

Algorithm 10 Proposed adjustment for hierarchical classification (testing).

```
Input: binary testing data $q$, set of class groupings $S_c$, learned parameters $r(l,i,t,g)$, $T_{l,i}$
and $\Lambda_{l,i}$ for the classification associated to each node $H_c$ in the tree of classifications $H$,
number of levels $L = (L_1, \cdots L_C)$ to be used in each classification.
Begin at $H_1$, the root classification.
while $H_c$ is not null, do
    Classify testing data $q$, as in Algorithm 8, with learned parameters $r(l,i,t,g)$, $T_{l,i}$, $\Lambda_{l,i}$
    from $H_c$ into one of the sets contained in $S_c$.
    if If $q$ is predicted to belong to a single class then
        Set $H_c$ to be null.
    else
        Set $H_c$ to be the node corresponding to the predicted set of classes for $q$ within $S_c$.
    end if
end while
```

challenging [DS16, HFK04].

5.2.3 Determining class hierarchies

When the number of classes is large, reorganizing a flat multiclass classification problem into hierarchical (binary) classifications can be used as a general strategy to reduce the computation required for testing [GP08]. Our proposed strategy need not only be applied in settings where the data follows or is presented within the context of a clear hierarchical structure. A variety of previous works have studied ways in which to detect structure among classes and use this information to construct an informed hierarchy of the classes [GP08, GSC02, SFR17, LZO07, ZBD99]. These strategies generally aim to group classes that are deemed 'similar' by some measure, in order to reduce the number of missclassifications that occur high in the tree. For example, some work suggests constructing a hierarchy based on the confusion matrix of the flat multiclass classification problem [GP08, GSC02, SFR17]. Preferentially constructing class hierarchies that are imbalanced in terms of ease of classification along different paths will also

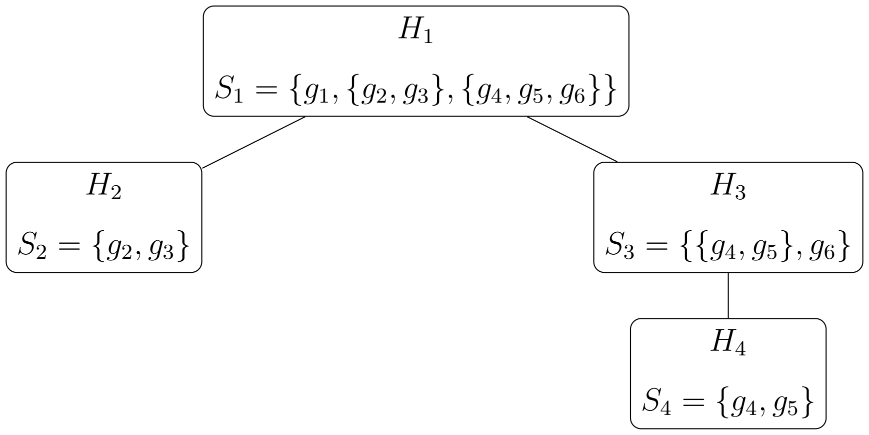

H_1
$S_1 = \{g_1, \{g_2, g_3\}, \{g_4, g_5, g_6\}\}$
H_2
$S_2 = \{g_2, g_3\}$
H_3
$S_3 = \{\{g_4, g_5\}, g_6\}$
H_4
$S_4 = \{g_4, g_5\}$

Figure 5.3: Example hierarchical classification tree. A classifier would be trained at each node, H_c, to classify data among the sets given in S_c.

largely affect the computational savings achieved by our proposed hierarchical classification method. We save details of how one might achieve this for future work.

5.3 Experimental results

In the following experiments, we test the computational gains achieved by the proposed hierarchical classification strategy as described in Algorithms 9 and 10 compared with direct classification into each individual group via 'flat multiclass classification' as described in Algorithms 7 and 8. The 'flat multiclass classification' is a direct application of the method proposed in [NSW17].

5.3.1 Two-dimensional synthetic data

We first test the computational gains achieved by the proposed hierarchical classification strategy on the two-dimensional data shown in Figure 5.5. Each color represents a different class and there are six classes in total. The red and yellow clusters each contain 200 training and testing points, while the remaining four classes, green, black, blue and cyan, contain 100 training and testing points each. The distribution of testing points among the classes will have a significant effect on the computation needed for testing in the hierarchical case. We expect classifying points from the red and yellow classes to be easier and to require fewer levels than correctly classifying points as green, black, blue or cyan.

To take advantage of this structure in the data, we first predict whether a testing point is red or yellow versus green, black, blue or cyan using only one level. If the test point was predicted to be red or yellow, we then discern between these two classes again using only a single level. If the test point was predicted to be green, black, blue or cyan, we then predict among these classes by using varying numbers of levels. Accuracies and testing flops for the hierarchical classification strategy versus flat multiclass classification are shown in Figure 5.5 for varying numbers of measurements m. We see a significant reduction in computational cost using the hierarchical strategy without sacrificing accuracy.

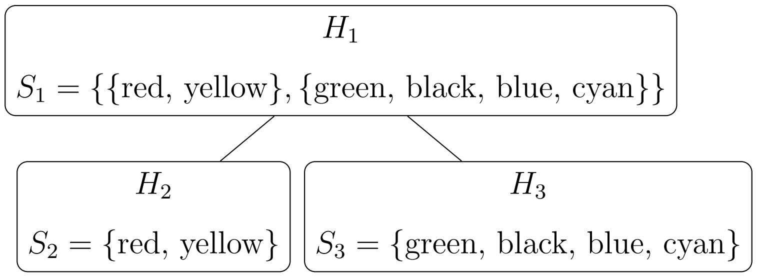

H_1
$S_1 = \{\{\text{red, yellow}\}, \{\text{green, black, blue, cyan}\}\}$
H_2
$S_2 = \{\text{red, yellow}\}$
H_3
$S_3 = \{\text{green, black, blue, cyan}\}$

Figure 5.4: Hierarchical classification tree used to classify two-dimensional synthetic data as shown in Figure 5.5.

5.3.2 Three-dimensional synthetic data

We test the hierarchical classification strategy and flat multiclass classification on three-dimensional synthetic data as given in Figure 5.6. Each color represents a different class. Again, we expect the four Gaussian clusters to require fewer levels for sufficiently accurate classification than the 'arcs'. The training data are distributed so that there is an equal number of training and testing points in the Gaussian clusters and arcs. Specifically we have 100 training and testing points in each arc and 200 training and testing points in each Gaussian cluster.

Using a strategy similar to that used in the two-dimensional experiment, we first build a classifier to predict whether a point belongs to one of the arcs or one of the Gaussian clusters

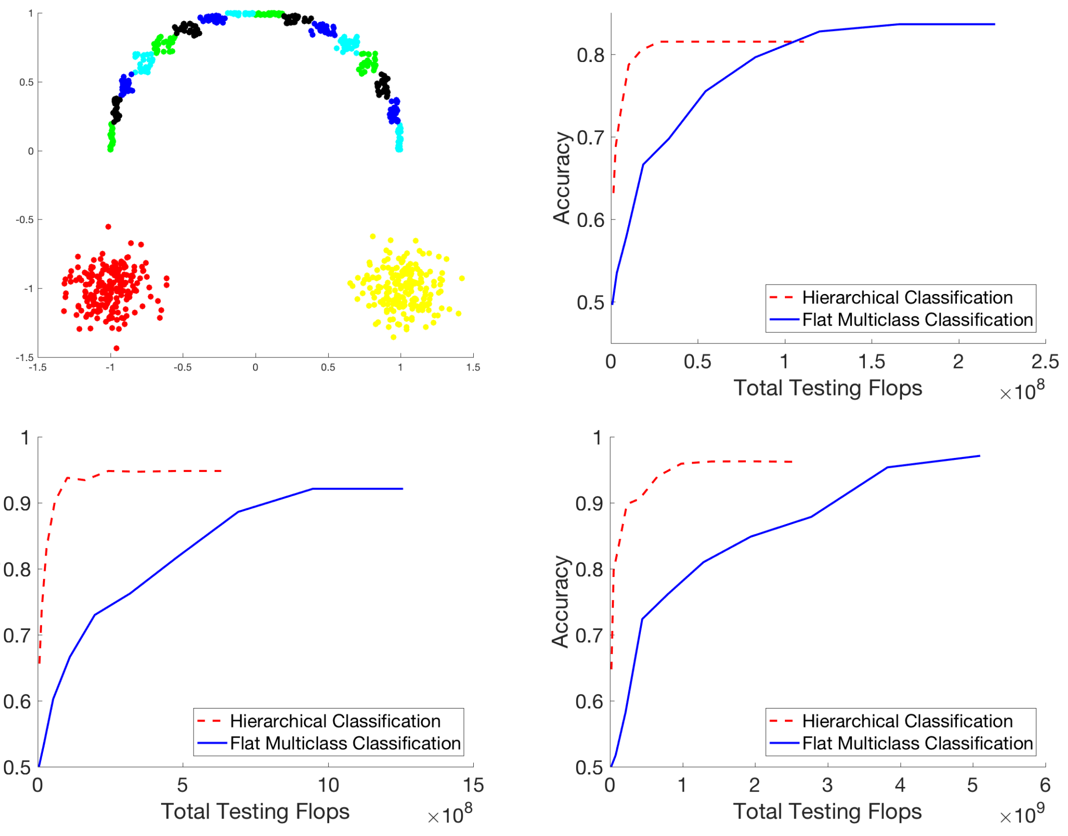


Figure 5.5: For the data distributed as given in the upper-left plot, where each color represents a different class, we classify testing data either by flat multiclass classification or our proposed hierarchical classification strategy where the first classification discerns between red or yellow versus green, black, blue or cyan. Accuracy and testing flops required are given in the subsequent plots using $m = 20, 50$ and 100 respectively. Results are averaged over 10 trials.

using only a single level. If a data point is predicted to be in one of the Gaussian clusters, we then use a single level again to predict to which of the clusters it belongs. If a data point is predicted to be in one of the arcs, we use more levels to perform the subsequent classification to discern between the arcs. We test the accuracy and computation required for using a variety of levels in this second classification. As in the two-dimensional experiment, we again see a reduction in the computational cost of testing without sacrificing accuracy.

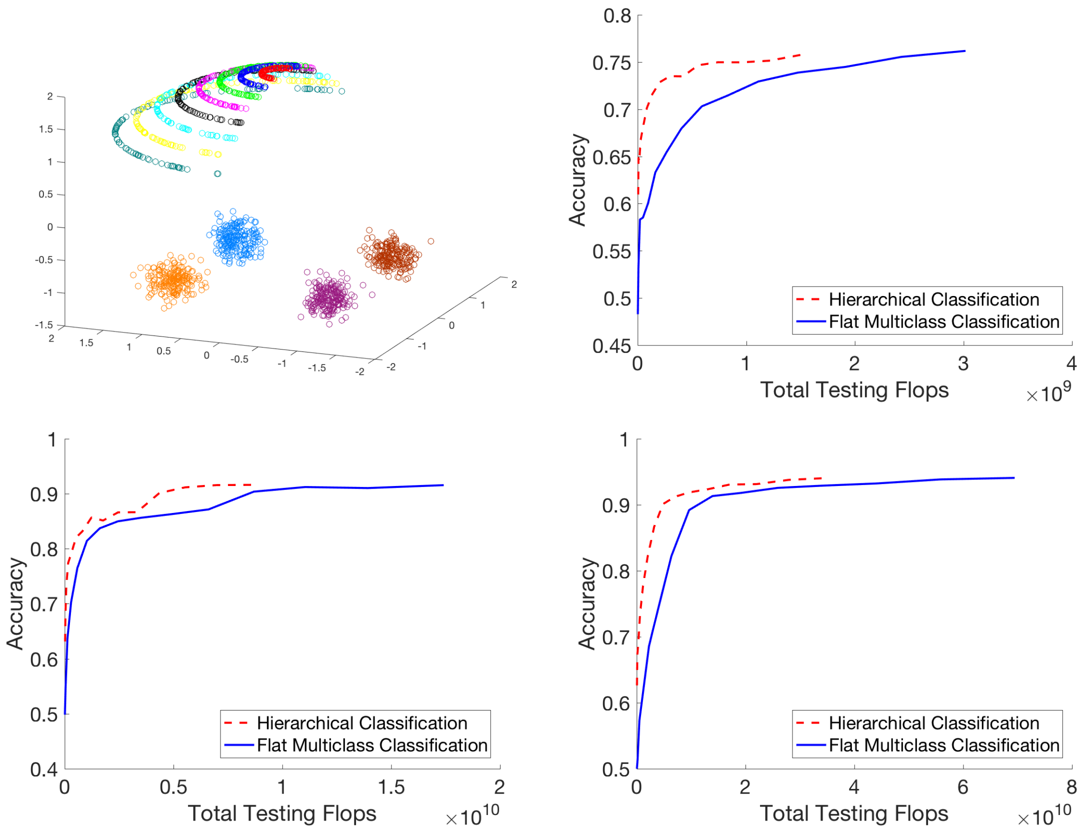


Figure 5.6: For the data distributed as given in the upper-left plot, where each color represents a different class, we classify testing data either by flat multiclass classification or our proposed hierarchical classification strategy where the first classification discerns between red or yellow versus green, black, blue or cyan. Accuracy and testing flops required are given in the subsequent plots using $m = 20, 50$ and 100 respectively. Results are averaged over 10 trials.

5.3.3 MNIST

Although not inherently hierarchical in nature, we demonstrate that our hierarchical strategy can lead to computational savings on the MNIST dataset of handwritten digits [LCB10]. Consider the digits 1-5. Intuitively and in practice, the digit 1 tends to be easier to classify correctly than the other digits. For example, if we apply the multiclass classification from [NSW17] to classify the digits 1-5 using 1000 training points for each class, 10 levels and testing on 200 training points from each class, we find that 98.5% of the 1s are classified correctly, whereas the overall accuracy of classifying the digits 1-5 was 89.2% (the accuracy for classifying digits 2-5 was 86.88%). Thus, it is reasonable to expect that fewer levels are required for sufficiently accurate classification of the 1s than are required to classify the remaining digits.

We induce hierarchical structure by first classifying into 1s versus not 1s, followed by classification into the digits 2,3,4 and 5 for those test points that were predicted to not be 1s in the first classification. When training the first classifier, we downsample the training data for the digits 2-5 so that we have an equal number of training data points for 1s and not 1s. We found that this adjustment improved the accuracy of the first classification. Five levels are used for the first classification into 1s versus not 1s and a varying number of levels (five to 10) are used for the subsequent classification. We again see a reduction in the total testing flops required to achieve a given accuracy. Here, we use an equal number of test points for each digit and thus get computational savings for approximately 1/5 of the test points, specifically for all of the test points that are predicted to be 1s. If we had a much higher proportion of 1s as compared to the other digits, then we would expect the computational savings to be even more significant. Additionally, since this tree is fairly shallow, as expected the improvements are mild, and we would expect more significant improvement for real data that has a larger and more imbalanced tree structure, as in the other experiments.

Conclusion

We have demonstrated that the classification algorithm proposed in [NSW17] can be readily adapted to to classify data in a hierarchical way that improves computational efficiency. We achieve this by using fewer levels to classify data points predicted to be from classes that are more readily identifiable. We could potentially further reduce computational costs for easier to classify data by reducing the number of measurements m in those cases as well. Theoretical guarantees as well as modifications that alleviate error propagation down the tree are important directions for future work.

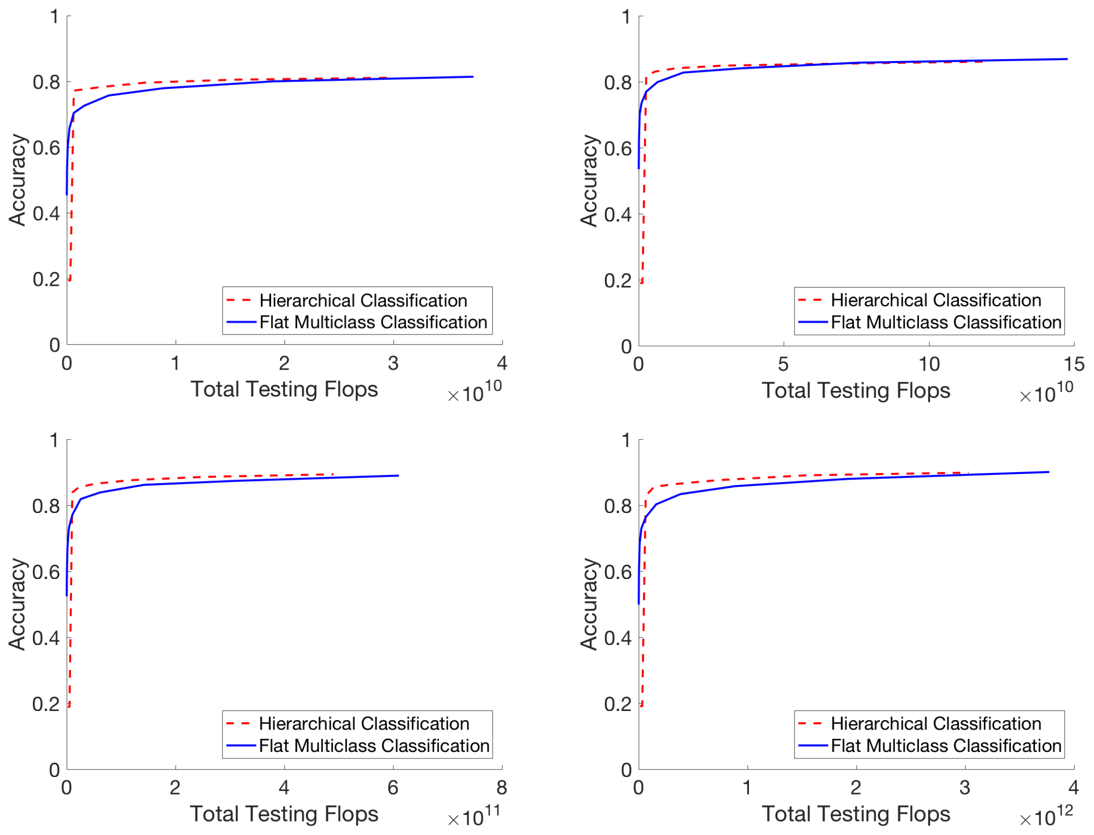


Figure 5.7: Accuracy and testing flops required for flat multiclass classification versus our proposed hierarchical classification strategy in classifying digits 1-5 in the MNIST dataset are given using $m = 50, 100, 200$ and 500 respectively. Results are averaged over 10 trials.

CHAPTER 6

An Iterative Method for Classification of Binary Data

6.1 Introduction

We consider the problem of performing classification when only binary measurements of data are available. This situation may arise due to the need for extreme compression of data or in the interest of hardware efficiency [FSL14, JLB13, LWY11, ASS96]. Despite this extremely coarse quantization of the data, one can still perform learning tasks, such as classification, with high accuracy. The authors of [NSW17] recently proposed a classification method for binary data, which they show to be reasonably accurate and sufficiently simple to allow for theoretical analysis in certain settings. Additionally, the predicted class can be approximately understood as the class whose binarized training data most closely and frequently matches that of the test point. As this approach will be the foundation of the work presented here, we discuss it in detail in the next section.

Interpretability of algorithms and the ability to explain predictions is of increasing importance as machine learning algorithms are applied to an expanding range of problems in areas such as medicine, criminal justice, and finance [BS16, BBH17, PMD16]. Decisions made based on algorithmic predictions can have profound repercussions for both participating individuals as well as society at large. A major drawback to complex models such as deep neural networks [LBB98, HZR16, CW08, LBH15] is that it is extremely difficult to explain how or why such algorithms arrive at a specific prediction, see e.g. [ZWZ17, ZZ18, VML17] and references therein. Studying and advancing models for which model output can be understood will help to both improve methods that are more readily interpretable and develop tools for understanding more complex models. The aim of this paper is to continue developing a framework with these two simultaneous goals in mind.

6.1.1 Contribution

We propose an extension of the simple classification method for binary data proposed in [NSW17], which we will henceforth refer to as SCB. Specifically, we propose an iterative method that uses output from SCB as input to a subsequent application of SCB. We refer to this iterative method as ISCB. We find that the iterative extension, ISCB, often leads to improved performance over SCB. Additionally, we demonstrate that SCB can be used for dimension reduction or as a data preprocessing step to improve the performance of other algorithms, such as support vector machines (SVM). While one can draw similarities between methods such as decision tree ensembles [MKS94, MKS93, BHB07] and approximate nearest neighbors [IM98, GIM99, ZLM14] with SCB, we focus here on the extension of SCB to an iterative framework.

The iterative ISCB approach is reminiscent of the compositional nature of neural networks. Due to the simplicity of the SCB framework, we can provide theoretical guarantees for the accuracy of the iterative extension in simple two-dimensional settings. We believe that studying this kind of iterative classification framework is interesting and practical in its own right, and will also serve as a step toward gaining a more thorough understanding of more complex deep learning strategies.

6.1.2 Organization

The paper is organized as follows. Section 6.2 introduces the problem statement and classification strategies of interest. Subsection 6.2.1 describes the SCB framework introduced in [NSW17] and Subsection 6.2.2 our proposed iterative extension. In Section 6.3, we demonstrate the performance of the proposed approach on real and synthetic datasets. Section 6.4 discusses variations and practical considerations. We provide theoretical guarantees for the proposed iterative method in several simplified settings and provide intuition as to why the iterative method generally outperforms the original approach in Section 6.5. Finally, Section 6.6 demonstrates how SCB can be adapted to serve as a data preprocessing and dimension reduction strategy for other methods applied to binary data.

6.2 Classification using binary data

We first introduce the problem and notation that will be used throughout. Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be a random measurement matrix (e.g. typically it will contain i.i.d. standard normal entries). Let $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_p] \in \mathbb{R}^{n\times p}$ be the matrix of p data vectors $\mathbf{x}_i \in \mathbb{R}^n$ with labels $\mathbf{b} = (b_1, \cdots b_p)$. Let G be the number of groups or classes to which the data points belong, so that we may assume $b_i \in \{1, 2, \dots, G\}$. Suppose we have the binary measurements of the data

$$\mathbf{Q} = \text{sign}(\mathbf{AX}),$$

where $\text{sign}(\mathbf{M})_{i,j} = \text{sign}(M_{i,j})$ and for a real number c the sign function simply assigns $\text{sign}(c) = 1$ if $c \geq 0$ and -1 otherwise. For a matrix $\mathbf{M}$, let $\mathbf{M}_{(j)}$ denote the j^{th} column of $\mathbf{M}$.

The rows of the matrix $\mathbf{A}$ can be viewed as the normal vectors to randomly oriented hyperplanes, in which case the $(i, j)^{\text{th}}$ entry of $\mathbf{Q}$ denotes on which side of the i^{th} hyperplane the j^{th} data point $\mathbf{x}_j$ lies. In practice, the binary data $\mathbf{Q}$ may be obtained during processing or be provided as direct input from some other source. In the latter case, we may not have access to the data matrix $\mathbf{X}$ or the measurement matrix $\mathbf{A}$, but only the resulting binary data $\mathbf{Q}$. We refer to the binary information indicating the position of a data point relative to a set of hyperplanes as a *sign pattern*. In particular, for a column $\mathbf{Q}_{(j)}$ and any subset of its entries, the resulting vector indicates the sign pattern of the j^{th} data point relative to that subset of hyperplanes.

We aim to classify a data point $\mathbf{x}$ based only on the binary information contained in $\mathbf{Q}$. As a simple motivating example, consider the left plot of Figure 6.1. The training data points each belong to one of three classes, red, blue, or green. Consider the test point indicated by the black $\mathbf{x}$. Cycling through the hyperplanes, the green hyperplane indicates that the test point likely belongs to the blue or red class (since it lies on the same side as these clusters), the purple hyperplane indicates that the test point likely belongs to the blue or green class,

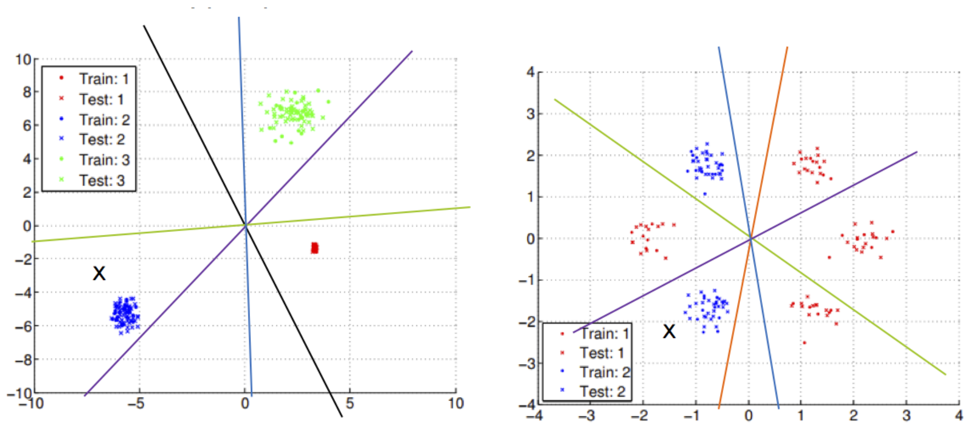


Figure 6.1: A motivating example for using positions relative to hyperplanes for classification.

the blue hyperplane indicates that the test point likely belongs to the blue class, and the black hyperplane indicates that the test point likely belongs to the blue class. In aggregate, the test point matches the relative positions of the blue class to the hyperplanes most often. This prediction matches what we might predict visually.

For the data in the right plot of Figure 6.1, there are both red and blue points on the same side as $\mathbf{x}$ for each hyperplane. However, if we consider sign patterns with respect to *pairs* of hyperplanes instead of only single hyperplanes, we can isolate data within cones or wedges as opposed to simply half-spaces. Comparing the sign patterns of the training data with respect to pairs of hyperplanes with that of the test point, we find that the test point $\mathbf{x}$ matches the sign patterns of the blue class most often. Thus, it may not be enough to consider hyperplanes individually, but in tuples. SCB uses this intuition as motivation.

6.2.1 Simple classification for binary data (SCB)

In SCB, sign patterns of the data with respect to tuples of hyperplanes of various lengths are recorded and aggregated to arrive at a prediction. The length of the sign patterns, or the number of hyperplanes considered, is referred to as the *level*. For each level $\ell = 1, \cdots, L$, we choose m random combinations of ℓ hyperplanes. Each of the hyperplane-tuples provides a *m*easurement of the data points. Fixing the number of hyperplane combinations considered, as opposed to considering all possible combinations, prevents the number of measurements from growing exponentially with the level.

Let t be a sign pattern for the i^{th} measurement at the ℓ^{th} level and $P_{g|t}$ be the number of training points in class g with sign pattern t. This sign pattern information is then aggregated for the training data points in the membership function $\mathbf{r}(\ell, i, t, g)$, with

$$\mathbf{r}(\ell, i, t, g) := \frac{P_{g|t}}{\sum_{j=1}^{G} P_{j|t}} \frac{\sum_{j=1}^{G} |P_{g|t} - P_{j|t}|}{\sum_{j=1}^{G} P_{j|t}}, \tag{6.1}$$

where $\ell = 1, \cdots, L$, $i = 1, \cdots m$, and $g = 1, \cdots, G$. The first term in this formula gives the fraction of points with sign pattern t that belong to class g, while the second acts as a balancing term to account for differences in the relative sizes of different classes. Each value in this membership function gives an indication of how likely a data point is to belong to class g based on the fact that it has sign pattern t for the i^{th} measurement at the ℓ^{th} level. Larger $\mathbf{r}(\ell, i, t, g)$ values indicate that a data point is more likely to belong to the g^{th} class. Training is detailed in Algorithm 11, which simply computes all of these quantities.

Given a test point $\mathbf{x}$, with binary data $\mathbf{q} = \text{sign}(\mathbf{A}\mathbf{x})$, for each level ℓ, measurement i and associated sign pattern t^* we find the corresponding $\mathbf{r}(\ell, i, t^*, g)$ value and keep a running sum for each group g, stored in the vector $\widetilde{\mathbf{r}}$ (note that the vector $\widetilde{\mathbf{r}}$ depends on the data point $\mathbf{x}$, but we notationally ignore this dependence for tidiness, and will write $\widetilde{\mathbf{r}}(g)$ for a class g or data point $\mathbf{y}$ when clarification is needed). If t^* does not match any of the sign patterns observed in the training data, then no update to $\widetilde{\mathbf{r}}$ is made. The testing procedure is detailed in Algorithm 12. In [NSW17], the authors showed that this classification method works well on both artificial and real datasets (e.g. MNIST [LCB10], YaleB [CHH07b, CHH07a, CHH06, HYH05]).

6.2.2 Iterative classification for binary data (ISCB)

We now introduce a novel iterative extension to SCB, which we refer to as ISCB. First, we motivate the extension through an example. Consider Figure 6.2, which plots the values of $\widetilde{\mathbf{r}}$ from Algorithm 12 for the task classifying the digits 0-4 of the MNIST dataset (where we will use class labels $0, 1, \ldots, 4$). Note that test images of the digit 0 typically have lower $\widetilde{\mathbf{r}}(1)$ values than do other digits. Similarly, test images of the digit 1 typically have lower $\widetilde{\mathbf{r}}(0)$

Algorithm 11 SCB Training from [NSW17]

```
**Input:** binary training data $\mathbf{Q}$, training labels $\mathbf{b}$, number of classes $G$, number of levels $L$.
for $\ell$ from 1 to $L$, $i$ from 1 to $m$ do
    Randomly select $\ell$ hyperplanes.
    for all observed sign patterns $t$ and classes $g$ from 1 to $G$ do
        Compute $\mathbf{r}(\ell, i, t, g)$ as in Equation (6.1).
    end for
end for
```

Algorithm 12 SCB Classification from [NSW17]

```
Input: binary testing data $\mathbf{q}$, number of classes $G$, number of levels $L$, learned parameters $\mathbf{r}(\ell, i, t, g)$, and hyperplane tuples from Algorithm 11.
for $\ell$ from 1 to $L$, $i$ from 1 to $m$ do
    Identify the sign pattern $t^*$ to which $\mathbf{q}$ corresponds for the $i^{\text{th}}$ $\ell$-tuple of hyperplanes.
    for $g$ from 1 to $G$ do
        $\widetilde{\mathbf{r}}(g) = \widetilde{\mathbf{r}}(g) + \mathbf{r}(\ell, i, t^*, g)$.
    end for
end for
Set $\widetilde{\mathbf{r}}(g) = \frac{\widetilde{\mathbf{r}}(g)}{Lm}$ for $g = 1, \cdots, G$.
Classify $\hat{b} = \text{argmax}_{g \in \{1, \cdots, G\}} \widetilde{\mathbf{r}}(g)$.
```

than do test images of the digits 1-4. Indeed, it is not only likely that

$$\hat{b}_{\mathbf{x}} = \underset{g \in \{1, \cdots, G\}}{\text{argmax}} \, \widetilde{\mathbf{r}}(g)$$

corresponds to the true digit label, but in addition the $\widetilde{\mathbf{r}}$ vectors for testing images from different digits contain different *patterns*. Thus, we expect that using a method more advanced than simply predicting the class corresponding to the maximum of the the $\widetilde{\mathbf{r}}$ vector may improve classification accuracy, specifically a strategy that makes use of the distribution of the values contained in $\widetilde{\mathbf{r}}$.

One could make predictions based on the $\widetilde{\mathbf{r}}$ vectors in a variety of ways. We mention a few such options here. Drawing intuition from simple neural network architectures such as multi-layer perceptron [Cyb89] and boosting algorithms such as AdaBoost [FS97, FSA99], we first consider using iterative applications of SCB, where $\widetilde{\mathbf{r}}$ values of the training data from previous iterations are used as input training data for the following iteration. In particular, the method is reminiscent of the structure of a single neuron in a neural network in which information only propagates forward as opposed to throughout the whole network. In contrast to deep neural networks, the output at each iteration, $\widetilde{\mathbf{r}}$, can be interpreted as a vector indicating to which class a data point $\mathbf{x}$ is likely to belong. This iterative strategy also relates to boosting in that subsequent iterations train on the shortcomings of previous iterations. Specifically, if points from a given class are misclassified, but produce similarly structured $\widetilde{\mathbf{r}}$ vectors this pattern may be corrected in the next application of the algorithm.

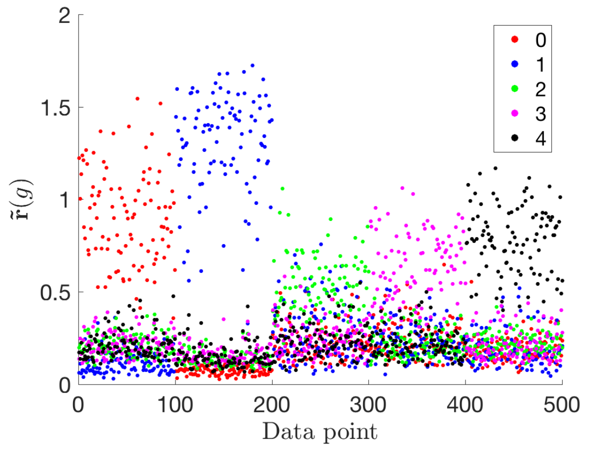


Figure 6.2: The $\widetilde{\mathbf{r}}(g)$ values from SCB trained to classify digits 0-4 from the MNIST dataset are plotted. Five digits are considered to ease visualization. One-hundred test points from each digit are used with points 1-100 corresponding to 0s, 101-200 corresponding to 1s, etc. $\widetilde{\mathbf{r}}(0)$ values are plotted in red, $\widetilde{\mathbf{r}}(1)$ in blue, $\widetilde{\mathbf{r}}(2)$ in green, $\widetilde{\mathbf{r}}(3)$ in magenta and $\widetilde{\mathbf{r}}(4)$ in black.

The training and testing phases of the proposed ISCB, are detailed in Algorithm 13 and Algorithm 14. To ease notation, let $\mathbf{r}_k$, $\widetilde{\mathbf{r}}_k$, and $\mathbf{A}^{(k)}$ be $\mathbf{r}$, $\widetilde{\mathbf{r}}$, and $\mathbf{A}$ from the k^{th} application of SCB (Algorithm 11 and Algorithm 12). During training, the first iteration in ISCB is executed as in Algorithm 11. We collect the data $\mathbf{X} = [\widetilde{\mathbf{r}}_1(\mathbf{x}_1) \cdots \widetilde{\mathbf{r}}_1(\mathbf{x}_p)] \in \mathbb{R}^{G \times p}$, which will be used as training data for the next iteration, where $\mathbf{x}_i$ are training data points. In contrast to SCB, the iterative algorithm calculates $\widetilde{\mathbf{r}}$ values for both the training and test data. Note that the dimension of the data points is fixed at G after the first application of SCB. For high dimensional data, we will typically have $G \ll n$. This reduction in dimension reduces the computational cost of some of the required computations, such as $\mathbf{Q} = \text{sign}(\mathbf{AX})$. One could also make use of the same measurement matrix $\mathbf{A}$ for all iterations after the first. Since the dimension is much smaller after the first iteration of SCB, one may also need fewer levels for accurate classification. We leave an exhaustive study of the many possible variations for future work, and focus here on establishing the mathematical framework of this iterative approach.

After each application of Algorithm 11, we collect sign information of our data with respect to a new set of random hyperplanes. Although the dimension of the data for the subsequent applications lies in $\mathbb{R}^G$ and thus we expect the size of this data to be manageable, there are several motivations for taking binary measurements of the data at each application. First, we can still take advantage of methods for efficient storage of and computation with binary data. Second, the binary measurements roughly preserve angular information about the data. For the $\widetilde{\mathbf{r}}$ values, we are generally interested in the relative sizes of the components, since these represent the likelihood that a point belongs to a given class. The overall magnitude of the $\widetilde{\mathbf{r}}$ values is of less importance and, thus, binary measurements retain the significant information pertaining to the data. Third, considering binary measurements of the data at each application maintains consistency between the applications, making the method more amenable to theoretical analysis, interpretability, and is more in line with sophisticated deep neural net architectures.

Since the components of $\widetilde{\mathbf{r}}$ are always non-negative, we restrict the random hyperplanes to intersect this space after the first application. For example, we can ensure the hyperplanes intersect this region by requiring that the normal vectors have at least one positive and one negative coordinate. We do not recenter the data after each application, as the structure of $\widetilde{\mathbf{r}}$ can lead to poor performance with recentering. For an example, consider Figure 6.3, in which the $\widetilde{\mathbf{r}}$ values follow a roughly linear trend for later applications of the method. Finally, after the last application of SCB, for the iterative algorithm, we make the prediction

$$\hat{b} = \underset{g \in \{1, \cdots, G\}}{\operatorname{argmax}} \, \widetilde{\mathbf{r}}_K(g).$$

Algorithm 13 ISCB Training.

```
Input: binary training data $\mathbf{Q} \in \mathbb{R}^{m \times p}$, training labels $\mathbf{b}$, number of classes $G$, number of levels $L$, number of applications $K$.
for $k$ from 1 to $K$, do
    Train learned parameters $\mathbf{r}_k(\ell, i, t, g)$ as in Algorithm 11, with input: $\mathbf{Q}$, $\mathbf{b}$, $G$ and $L$.
    Set $\mathbf{X} = 0 \in \mathbb{R}^{G \times p}$.
    for $j$ from 1 to $p$ do
        Apply Algorithm 12 to $\mathbf{Q}_{(j)}$ using learned parameters $\mathbf{r}_k(\ell, i, t, g)$ to calculate $\widetilde{\mathbf{r}}_k$.
        Set $\mathbf{X}_{(j)} = \widetilde{\mathbf{r}}_k$.
    end for
    Form the random measurement matrix $\mathbf{A}^{(k)} \in \mathbb{R}^{m \times G}$.
    Set $\mathbf{Q} = \operatorname{sign}(\mathbf{A}^{(k)}\mathbf{X})$.
end for
Output: $\mathbf{r}_k(\ell, i, t, g)$, and $\mathbf{A}^{(k)}$ for $k$ from 1 to $K$.
```

6.3 Experimental results

We test ISCB on synthetic and image datasets. The synthetic datasets demonstrate why the iterative method is effective for certain simple settings and how the data transforms between iterations. ISCB is also tested on the MNIST dataset of hand-written digits [LCB10], the YaleB dataset for facial recognition [CHH07b, CHH07a, CHH06, HYH05] and the Norb dataset for classification of images of various toys [LHB04].

6.3.1 Two-dimensional synthetic data

We further motivate ISCB through examples with two-dimensional synthetic data. For two-dimensional data with two classes, the dimension of the input data for all applications of

Algorithm 14 ISCB Testing.

```
Input: binary test data $\mathbf{q} \in \mathbb{R}^m$, number of classes $G$, levels $L$, and iterations $K$, learned parameters $\mathbf{r}_k(\ell, i, t, g)$, hyperplane tuples, and $\mathbf{A}^{(k)}$ from Algorithm 13.
for $k$ from 1 to $K$ do
  Set $\widetilde{\mathbf{r}}_k = 0$.
  for $\ell$ from 1 to $L$, $i$ from 1 to $m$, do
    Identify the pattern $t^*$ to which $\mathbf{q}$ corresponds for the $i^{\text{th}}$ $\ell$-tuple of hyperplanes.
    for $g$ from 1 to $G$ do
      Update $\widetilde{\mathbf{r}}_k(g) = \widetilde{\mathbf{r}}_k(g) + \mathbf{r}_k(\ell, i, t^*, g)$.
    end for
  end for
  Set $\mathbf{q} = \text{sign}(\mathbf{A}^{(k)}\widetilde{\mathbf{r}}_k)$.
end for
Classify $b = \text{argmax}_{g\in\{1,\cdots,G\}} \widetilde{\mathbf{r}}_K(g)$.
```

SCB is two-dimensional and so we can easily visualize the effect of the iterations on both the training and testing data. We find that the more easily discerned data points are pushed out toward the boundary of the positive quadrant, while data points that are closer to the class boundary linger closer to the line $y = x$ in subsequent iterations and thus has a higher likelihood of being predicted as the other class at the next iteration.

Consider the data given in the upper left plot of Figure 6.3. There are two times as many points from the red class considered, both in the training and testing set. Half of the red points in testing and training lie on either side of the blue data points. Thus, applying SCB using a single level leads to all of the blue test points being misclassified as red. The abhorrent misclassification of the blue points is caused by the fact that we are using only a single level ($L = 1$) and for any hyperplane at least as many red points as blue lie on either side of it. The $\widetilde{\mathbf{r}}_1$ values, plotted in the upper right plot of Figure 6.3, have a much nicer distribution in terms of ease of classification; in fact, they are nearly linearly separable. The separation in the $\widetilde{\mathbf{r}}_1$ values between the blue and red points occurs since $\widetilde{\mathbf{r}}_1$(red) is generally

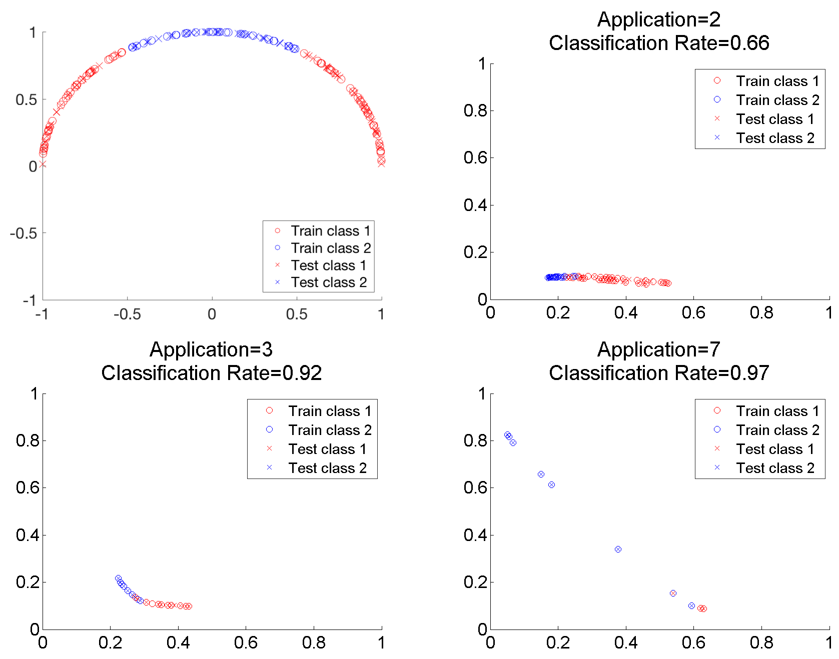


Figure 6.3: The $\widetilde{\mathbf{r}}_k$ vectors are plotted for ISCB with varying numbers of iterations k. The original training and testing data are shown in the upper left plot. Circles indicate training data and crosses indicate testing data. One level is used for each application of SCB and the subsequent plots give the $\widetilde{\mathbf{r}}_k$ vectors for $k = 1, 3$, and 7 respectively. The classification rate after a single application of SCB is 66%. After three iterations the classification rate is 92% and after seven application the classification rate is 97%.

larger for red points than for the misclassified blue points. That is, the points that truly belong to the red class are more "confidently" classified as red than are the blue points. If we consider the $\widetilde{\mathbf{r}}_1$ values as data, applying SCB now classifies the data with much higher accuracy (92% as compared to 66% for the original training data), while still only using a single level. By the seventh iterative application of SCB, the accuracy increases to 97%. If we perform the same experiment, but include a higher density of blue points so that the total number of red and blue points are the same, we achieve higher accuracy at the first application of SCB, but again see improved accuracy for later iterations.

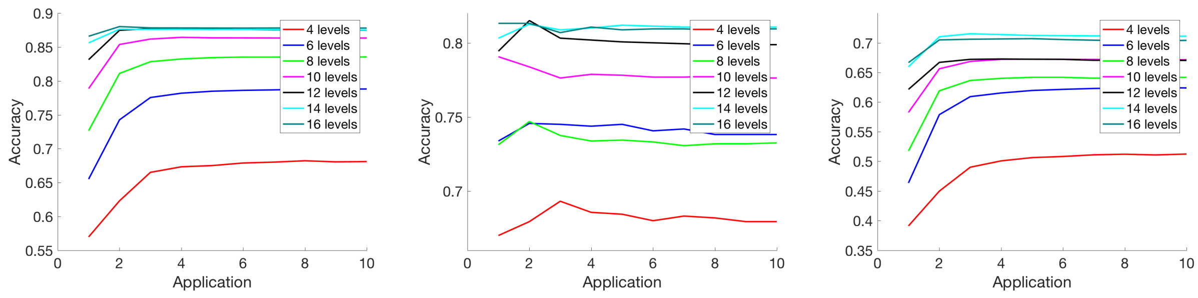


Figure 6.4: Accuracies for classifying MNIST data among 10 classes (left plot), YaleB data among eight classes (middle plot) and Norb data among five classes (right plot) are given in terms of the number of applications of SCB used. For the MNIST dataset, the model is trained with $p = 1000$ images of each class and tested on 100 images from each class. The model for the YaleB dataset is trained using $p = 40$ training images from each class and applied to 20 test images from each class. For the Norb dataset, the model is trained on $p = 1000$ training images and is applied to 200 test images for each class. In each model, $m = 500$ measurements are used. Results are averaged over 10 trials.

6.3.2 Image datasets

We test ISCB on the MNIST dataset of hand-written digits [LCB10], the YaleB dataset for facial recognition [CHH07b, CHH07a, CHH06, HYH05], and the Norb dataset for classification of images of various toys [LHB04]. Results are shown in Figure 6.4. We generally find both that increasing the number of levels used in each SCB application of ISCB and increasing the number of applications leads to improved performance. The classification accuracies typically level off after only a few applications of SCB, with the largest improvement typically occuring between the first and second application. These trends are less clear in the YaleB dataset, but this may be in part due to the limited amount of training data available for this dataset.

6.4 Alternative iterative method

We motivated ISCB by noting that the $\widetilde{\mathbf{r}}$ vectors from Algorithm 12 for test data from the same classes share similar structures. We additionally find that the contributions to the $\widetilde{\mathbf{r}}$ values coming from different levels admit different patterns as well. We could thus choose to use

$$\hat{\mathbf{r}}_k(\ell, g) = \sum_{i=1}^{m} \sum_{t^*} \mathbf{r}_k(\ell, i, t^*, g)$$

as data for the k^{th} application of SCB instead of $\widetilde{\mathbf{r}}_k(g)$ as is done in Algorithm 13. Here, t^* ranges over all observed sign patterns for the i^{th} ℓ-tuple of hyperplanes. We refer to this method as *ISCB with* $\hat{\mathbf{r}}$. Note that we have the following relation between $\widetilde{\mathbf{r}}_k$ and $\hat{\mathbf{r}}_k$,

$$\widetilde{\mathbf{r}}_k(g) = \sum_{\ell=1}^{L} \hat{\mathbf{r}}_k(\ell, g).$$

After the first application of SCB, the dimension of the data for ISCB with $\hat{\mathbf{r}}$ is now $\mathbb{R}^{LG}$.

In certain settings, ISCB with $\hat{\mathbf{r}}$ performs better than ISCB of Subsection 6.2.2. Typically, using $\hat{\mathbf{r}}_k(\ell, g)$ as opposed to $\widetilde{\mathbf{r}}_k(g)$ as input to the subsequent applications of Algorithm 11 performs better when the number of levels L used is small. Unfortunately, for higher numbers of levels L we see drastic declines in performance for later applications when using $\hat{\mathbf{r}}_k(\ell, g)$, as this method is more prone to overfit. These trends are illustrated in Figure 6.5 for the MNIST dataset. In the left plot of Figure 6.5, ISCB with $\hat{\mathbf{r}}$ leads to improved performance over ISCB with $\widetilde{\mathbf{r}}$. As the number of levels L used increases from four to 10, however, this difference diminishes. For greater than 14 levels, using ISCB with $\hat{\mathbf{r}}_k(\ell, g)$ leads to decreasing performance in the number of applications of SCB (seen in the right plot of Figure 6.5). The same decrease in performance does not occur when using the $\widetilde{\mathbf{r}}_k(g)$ values as data for the next iteration.

ISCB is relatively prone to overfitting, since the model output at iteration k is used as input for the $(k+1)^{\text{st}}$ iteration. Thus, any overfitting that occurs at earlier iterations gets propagated to later iterations. In particular, if we overfit to the training data at iteration k, then the training and testing data at the next iteration are no longer sampled from similar distributions. ISCB with $\hat{\mathbf{r}}$ has a much greater propensity to overfit as compared to ISCB

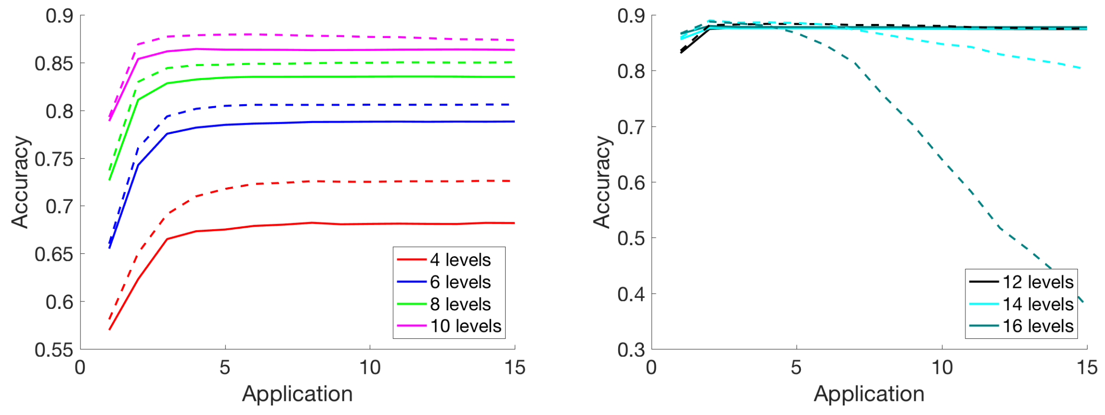


Figure 6.5: The performance of ISCB using $\widetilde{\mathbf{r}}$, presented in Subsection 6.2.2, (solid) is compared with that of the alternative version of ISCB using $\hat{\mathbf{r}}$, presented in Section 6.4, (dashed) on the MNIST dataset. $p = 1000$ training and 100 testing images are used for each digit. Each method uses $m = 500$ binary measurements of the data at each application of Algorithm 11. The number of levels L used with each method is indicated in the legend.

with $\widetilde{\mathbf{r}}$, especially when the number of levels is large. This effect makes intuitive sense since for longer ℓ-tuples of hyperplanes the testing data is less likely to match the sign patterns of training data and so the $\hat{\mathbf{r}}_k(\ell, g)$ values for training and testing data may diverge for longer ℓ-tuples and later iterations k. These observations suggest that choosing an appropriate number of levels is especially critical for ISCB as compared to SCB. Fortunately, if a model is trained using too many levels, one could simply use the model output from the first application to arrive at more accurate predictions. In particular, there is no need to re-train the model.

6.5 Theoretical analysis

We next offer some theoretical analyses pertaining to why we expect performance to improve through multiple applications of SCB for several simple scenarios. At a high level, the iterative framework has the opportunity to train on its own output and correct misclassifications that occur in previous iterations. Qualitatively, as the number of iterations increases, we find that the data points that are more easily identifiable as belonging to a single class are pushed toward extreme points of the range of outputs, while data points that are more difficult to classify fall in the interior of the range and have the chance to be classified correctly at the next iteration.

6.5.1 Binary classification of point masses of equal mass

As a first simple but illustrative example, consider a classification task between two classes. Assume that the training and testing data for each class is concentrated at a single point, i.e. a point mass, and that each class has the same number of training points or equivalently that each point mass has the same density. Let j be the number of hyperplanes that separate the two point masses at the first application of the classification method (j will clearly depend on the angle between the point masses and can be easily bounded probabilistically). Consider the simplified setting in which we use a single level; note that in this case, since $L = 1$, we have $\widetilde{\mathbf{r}}_k = \hat{\mathbf{r}}_k$ and so ISCB and ISCB with $\hat{\mathbf{r}}$ are equivalent. With this setup, for testing data in class 1,

$$\widetilde{\mathbf{r}}_1 = \left(\sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 1), \sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 2) \right) = (j, 0).$$

For testing data in class 2,

$$\widetilde{\mathbf{r}}_1 = \left(\sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 1), \sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 2) \right) = (0, j).$$

Thus, if at least one hyperplane separates the two point masses initially, then at the next iteration, the angle between data points of class 1 and 2 is $\pi/2$ (the best possible). Since the data are two-dimensional and we restrict the hyperplanes to intersect the positive quadrant after the first SCB application, then if the model classifies the point masses correctly at the first iteration, it will correctly classify at all subsequent iterations as well.

6.5.2 Binary classification of point masses

We next consider the slightly more involved setting in which the data from each class is again concentrated at a single point, however, the number of points in the two classes differ. We again consider only a single level L and let j be the number of hyperplanes that separate the two point masses in the first application of SCB. In expectation, $\frac{j}{m}$ gives an indication of the angle separating the two point masses, where m is the number of rows in the measurement matrix. Let A_1 be the number of points in class 1 and A_2 be the number of points in class 2. For testing data in class 1,

$$\widetilde{\mathbf{r}}_1(1) = \sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 1) = j + (m-j)\frac{A_1|A_1 - A_2|}{(A_1+A_2)^2} \text{ and}$$
$$\widetilde{\mathbf{r}}_2(2) = \sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 2) = (m-j)\frac{A_2|A_1 - A_2|}{(A_1+A_2)^2}.$$

For testing data in class 2,

$$\widetilde{\mathbf{r}}_1(1) = \sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 1) = (m-j)\frac{A_1|A_1 - A_2|}{(A_1+A_2)^2} \text{ and}$$
$$\widetilde{\mathbf{r}}_2(2) = \sum_{i=1}^{m} \mathbf{r}(\ell, i, t^*, 2) = j + (m-j)\frac{A_2|A_1 - A_2|}{(A_1+A_2)^2}.$$

Note that the data for the second application of the method are again two-dimensional. Let $\widetilde{\mathbf{g}}_1$ be the $\widetilde{\mathbf{r}}_1$ vector for a data point in class 1 and $\widetilde{\mathbf{g}}_2$ be the $\widetilde{\mathbf{r}}_1$ vector for a data point in class 2. The following formula gives the angle θ between the two point masses at the second application,

$$\theta = \cos^{-1}\left(\frac{\langle \widetilde{\mathbf{g}}_1, \widetilde{\mathbf{g}}_2 \rangle}{||\widetilde{\mathbf{g}}_1||_2 \cdot ||\widetilde{\mathbf{g}}_2||_2}\right). \tag{6.2}$$

Figure 6.6 shows the angle that separates the point masses of the training data at the second application in terms of $\frac{j}{m}$ for various ratios $c = \frac{A_1}{A_2}$. We find that if A_1 and A_2 are similar in size, then the expected angle separating the two point masses increases for the second application, making the point masses "easier" to separate in later applications.

6.5.3 Probabilistic bounds for an angular model

We next consider an analogue to Theorem 1 of [NSW17], although the setting is modified slightly. Consider two-dimensional data with two classes. Suppose that the data from each class is distributed within the disjoint wedges, G_1 and G_2, with angles A_1 and A_2 respectively. This setup is illustrated in Figure 6.7. Consider the data points $\mathbf{x}_1$ and $\mathbf{x}_2$, which lie on the

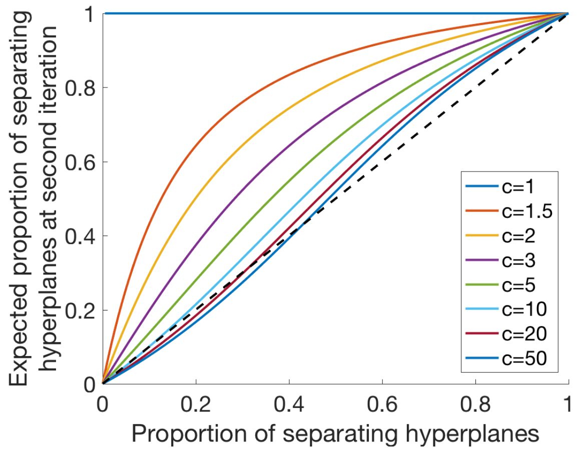


Figure 6.6: This plot shows the expected proportion of hyperplanes that separate data at the second iteration of ISCB given the fraction of separating hyperplanes at the first application of SCB. The relative sizes of the two classes, given by A_1 and A_2, are varied as well, as is indicated by the parameter $c = \frac{A_1}{A_2}$ given in the legend.

inside edge of each wedge. Let A_{12} be the angle between these two points. We aim to find a lower bound on the angle between the $\widetilde{\mathbf{r}}_1$ vectors for $\mathbf{x}_1$ and $\mathbf{x}_2$ after a single application of SCB with a single level L. Again, since we only use a single level, $\hat{\mathbf{r}}_1 = \widetilde{\mathbf{r}}_1$ for all points $\mathbf{x}$.

Assume that the data is distributed with uniformly random angles within G_1 and G_2. Let k_1 and k_2 be the number of hyperplanes that intersect G_1 and G_2 respectively and let j be

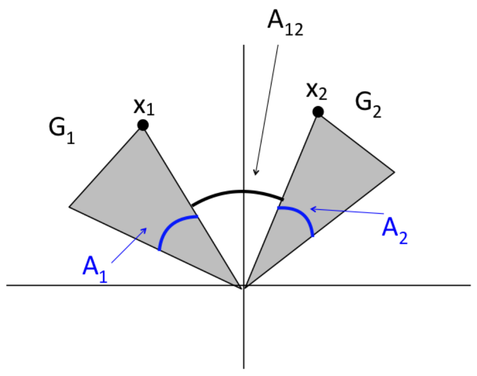


Figure 6.7: Illustration of data setup for Subsection 6.5.3.

Hyperplane case	Number in event	Class	Value of $\mathbf{r}(1, i, t, g)$
Separates $\mathbf{x}_1$ and $\mathbf{x}_2$	j	1	1
		2	0
Does not separate $\mathbf{x}_1$ and $\mathbf{x}_2$	$m - j - k_1 - k_2$	1	$\frac{A_1\|A_1-A_2\|}{(A_1+A_2)^2}$
or intersect G_1 or G_2		2	$\frac{A_2\|A_1-A_2\|}{(A_1+A_2)^2}$
Intersects G_2	k_2	1	$\frac{A_1\|A_1-A_2u'\|}{(A_1+A_2u')^2}$
		2	$\frac{A_2u'\|A_1-A_2u'\|}{(A_1+A_2u')^2}$
Intersects G_1	k_1	1	$\frac{A_1u\|A_1u-A_2\|}{(A_1u+A_2)^2}$
		2	$\frac{A_2\|A_1u-A_2\|}{(A_1u+A_2)^2}$

Table 6.1: Contributions to the membership index parameter $\mathbf{r}$ for the point $\mathbf{x}_1$ and for hyperplanes of various types. The variables u and u' are i.i.d. random variables uniformly distributed between zero and one, indicating the angle at which random hyperplanes intersect the wedges G_1 and G_2. $A_1, A_2, G_1, G_2, \mathbf{x}_1$ and $\mathbf{x}_2$ are as shown in Figure 6.7.

the number of hyperplanes that separate G_1 and G_2. Note that

$$\mathbb{E}k_1 = \frac{A_1}{\pi}, \quad \mathbb{E}j = \frac{A_{12}}{\pi}, \quad \text{and} \quad \mathbb{E}k_2 = \frac{A_2}{\pi}.$$

Assume that the hyperplanes are also distributed with uniformly random angles within these wedges. We can then replace $P_{g|t}$ with angular measures, specifically, $A_i u_h$, where $u_h \in [0,1]$ and depends on the angle at which the hyperplane h intersects G_i. Since the hyperplanes are uniformly distributed at random within each region, the u_h are uniform random variables between zero and one.

The contribution to the membership index parameter $\mathbf{r}$ for SCB with a single level L and for each possible type of hyperplane in this setup for the point $\mathbf{x}_1$ are summarized in Table 6.1. To simplify calculations, assume that $A_1 = A_2$. With this assumption, the membership index parameters no longer depend on A_1 or A_2. Summing over all hyperplanes, for $\mathbf{x}_1$ we have

$$\widetilde{\mathbf{r}}_1(1) = \sum_{i=1}^{m} \mathbf{r}(1, i, t_i^*, 1) = j + \sum_{h=1}^{k_1} \frac{u_h(1-u_h)}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{1-u'_h}{(1+u'_h)^2}$$

and

$$\widetilde{\mathbf{r}}_1(2) = \sum_{i=1}^{m} \mathbf{r}(1, i, t_i^*, 2) = \sum_{h=1}^{k_1} \frac{1-u_h}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{u'_h(1+u'_h)}{(1+u'_h)^2}.$$

The calculation for $\mathbf{x}_2$ is similar. Let $\widetilde{\mathbf{g}}_1$ and $\widetilde{\mathbf{g}}_2$ be the $\widetilde{\mathbf{r}}_1$ vectors corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively. Then at the next application, we have

$$\widetilde{\mathbf{g}}_1 = \left(j + \sum_{h=1}^{k_1} \frac{u_h(1-u_h)}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{1-u'_h}{(1+u'_h)^2}, \sum_{h=1}^{k_1} \frac{1-u_h}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{u'_h(1-u'_h)}{(1+u'_h)^2} \right) \tag{6.3}$$

and

$$\widetilde{\mathbf{g}}_2 = \left(\sum_{h=1}^{k_1} \frac{u_h(1-u_h)}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{1-u'_h}{(1+u'_h)^2}, j + \sum_{h=1}^{k_1} \frac{1-u_h}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{u'_h(1-u'_h)}{(1+u'_h)^2} \right). \tag{6.4}$$

The angle between these two vectors is again given by Equation (6.2). The resulting angles from simulations for various $k_1 = k_2$, and j are given in the left plot of Figure 6.8. We make the simplification $k_1 = k_2$ to ease visualization. Unsurprisingly, as j increases so does the separation between $\widetilde{\mathbf{g}}_1$ and $\widetilde{\mathbf{g}}_2$ at the second iteration. As k_1 and k_2 increase, for fixed j, the separation between the two points at the next application decreases.

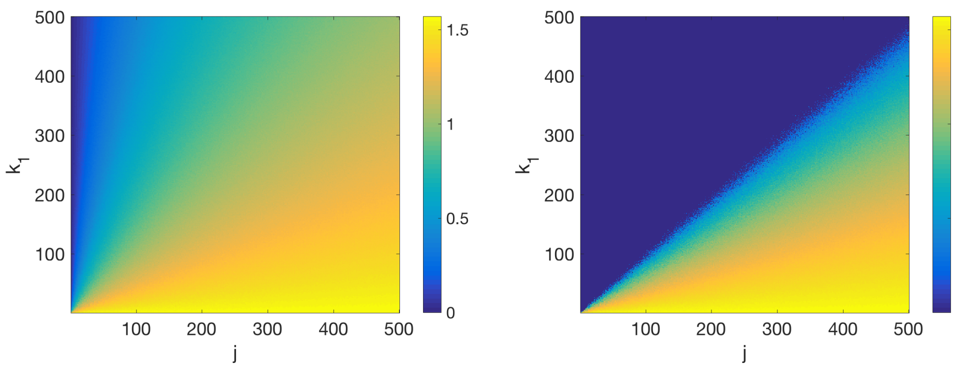


Figure 6.8: For various values of $k_1 = k_2$ (the number of hyperplanes intersecting the wedges G_1 and G_2 respectively) and j (the number of hyperplanes separating the wedges G_1 and G_2), the left plot indicates the true angle (in radians) between $\widetilde{\mathbf{g}}_1$ and $\widetilde{\mathbf{g}}_2$ as given in Equations (6.3) and (6.4). The right plot indicates the angle using the upper bound for $\cos(\theta)$ given in Equation (6.5).

Ideally, we would like to find a lower bound on the angle θ between $\widetilde{\mathbf{g}}_1$ and $\widetilde{\mathbf{g}}_2$ that depends on k_1, k_2, and j. Unfortunately, the explicit form of the resulting angle is relatively complicated. We can simplify the denominator of Equation (6.2) by using the bounds $||\widetilde{\mathbf{g}}_i||_2 \geq j$. We expect this bound to be quite loose, if not trivial, when j is small, but to provide a reasonable bound for larger j. With this simplification,

$$\begin{aligned}\cos(\theta) \leq & \frac{\left(j + \sum_{h=1}^{k_1} \frac{u_h(1-u_h)}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{1-u'_h}{(1+u'_h)^2}\right)\left(\sum_{h=1}^{k_1} \frac{u_h(1-u_h)}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{1-u'_h}{(1+u'_h)^2}\right)}{j^2} \\ & + \frac{\left(\sum_{h=1}^{k_1} \frac{1-u_h}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{u'_h(1-u'_h)}{(1+u'_h)^2}\right)\left(j + \sum_{h=1}^{k_1} \frac{1-u_h}{(u_h+1)^2} + \sum_{h=1}^{k_2} \frac{u'_h(1-u'_h)}{(1+u'_h)^2}\right)}{j^2}.\end{aligned} \tag{6.5}$$

For this simplified bound, taking an expectation is a straightforward calculation. See Section 6.A for details. We eventually arrive at the bound

$$\begin{aligned}\mathbb{E}(\cos(\theta)) \leq & \frac{(k_1+k_2)(2\log 2 - 1)}{j} + \frac{(k_1^2+k_2^2)(10(\log 2)^2 - 14\log 2 + 5)}{j^2} \\ & + \frac{4k_1k_2(1-\log 2)(3\log 2 - 2) + (k_1+k_2)(-2/3 + 8\log 2 - 10(\log 2)^2)}{j^2}\end{aligned} \tag{6.6}$$

Using Markov's inequality, for $a \in (0, \pi/2)$,

$$\mathbb{P}(\theta \leq a) = \mathbb{P}\left[\cos(\theta) \geq \cos(a)\right] \leq \frac{\mathbb{E}(\cos(\theta))}{\cos(a)}. \tag{6.7}$$

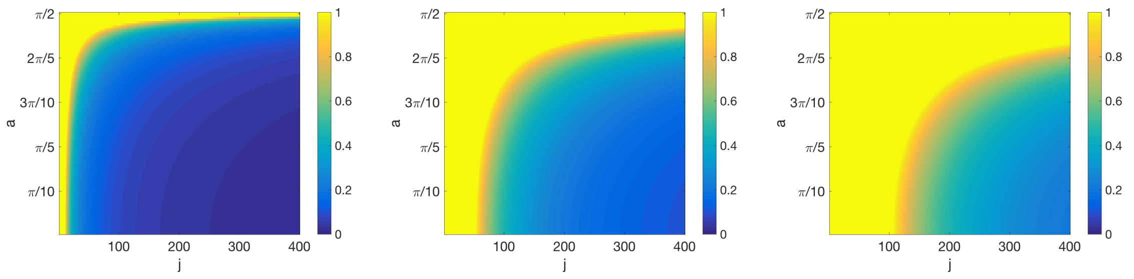


Figure 6.9: For various values of $k_1 = k_2$ (the number of hyperplanes intersecting the wedges G_1 and G_2 respectively), j (the number of hyperplanes separating the wedges G_1 and G_2), and angles a, we plot the bound for $\mathbb{P}(\theta \leq a)$ given by Theorem 6.5.1. From left to right, the plots use $k_1 = 10, 50$, and 100 respectively.

Although this bound is relatively loose, for sufficiently small a and large j, the probability that $\theta \leq a$ is small. We summarize this result in Theorem 6.5.1. More visually appealing, Figure 6.9 gives the probabilities that result from combining Equation (6.6) and Equation (6.7) for a variety of hyperplane combinations and angles a.

Theorem 6.5.1. *Suppose data is distributed as in Figure 6.7, where points from classes 1 and 2 are uniformly distributed within the wedges G_1 and G_2 respectively. Suppose that the angles A_1 and A_2 are equal. Let k_1 and k_2 be the number of hyperplanes that intersect G_1 and G_2 respectively. Let j be the number of hyperplanes that separate G_1 and G_2. Consider the points $\mathbf{x}_1$ in class 1 and $\mathbf{x}_2$ in class 2 as shown in Figure 6.7. The angle θ between the $\widetilde{\mathbf{r}}$ vectors for $\mathbf{x}_1$ and $\mathbf{x}_2$ after a single iteration of SCB with one level L satisfies the following inequality,*

$$\mathbb{P}(\theta \leq a) \leq \frac{C_1 j(k_1+k_2) + C_2(k_1^2+k_2^2) + C_3 k_1 k_2 + C_4(k_1+k_2)}{j^2\cos(a)},$$

where

$$C_1 = 2(\log 2) - 1, \qquad C_2 = 10(\log 2)^2 - 14\log 2 + 5,$$

$$C_3 = 4(1-\log 2)(3\log 2 - 2), \qquad C_4 = -10(\log 2)^2 + 8\log 2 - 2/3.$$

6.6 ISCB for data preprocessing and dimension reduction

We remark here briefly about another potential strategy using the output of the SCB approach. Although this is not the focus of the current work, it may lead to fruitful future directions. The idea is to use the output from SCB and then apply other established classification methods such as SVM [CV95] to the $\widetilde{\mathbf{r}}$ vectors. Considering SVM specifically, we find that this strategy can perform better than SVM applied directly to the data.

First, consider a simple example with the synthetic data shown in the upper left plot of Figure 6.10. Applying SVM with a linear kernel [FHT01, CV95] unsurprisingly performs poorly, achieving an accuracy of 65%. An RBF kernel [Buh03, FHT01] SVM performs much better, achieving an accuracy of 90%. Applying SVM instead to the $\widetilde{\mathbf{r}}_1$ values of the training data produced via SCB with a single level L and $m = 100$ measurements leads to 80% accuracy using a linear kernel and 97% accuracy using an RBF kernel. Thus, applying SVM to the $\widetilde{\mathbf{r}}_1$ values as opposed to the original data leads to an improvement in accuracy of 15% for SVM with a linear and 7% for SVM with an RBF kernel.

For the same initial data, if we increase the number of levels L used in SCB to four and the number of measurements to $m = 200$, the accuracies of SVM trained on the resulting $\widetilde{\mathbf{r}}_1$ values are 97% with a linear kernel and 94% with an RBF kernel (Figure 6.11). The respective accuracies are improved by 21% and 4% respectively as compared to SVM applied to the original data. This increase in the number of levels L and measurements m also leads to improved performance for both SCB and ISCB with two applications. Note that if SCB is able to perfectly classify the training data points, then SVM with a linear kernel trained on the $\widetilde{\mathbf{r}}_1$ values of the training points will also perfectly classify the training data points, as the $\widetilde{\mathbf{r}}_1$ values of the training points will be linearly separable.

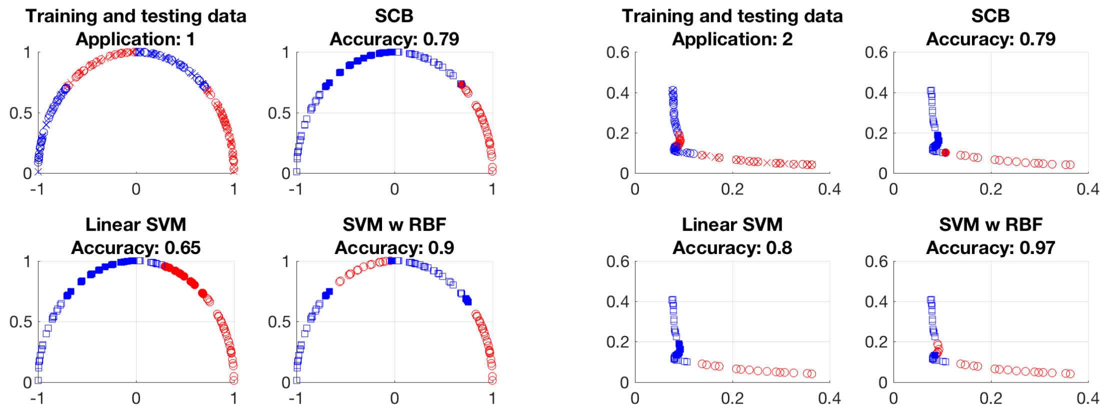


Figure 6.10: The four plots on the left display accuracies and predictions made via various methods for the data given in the upper left-most plot. In the plots of the training and testing data, circles indicate training data and crosses indicate test data. Filled markers indicate that a given method misclassified that particular data point. The methods considered are SCB and SVM with both a linear and RBF kernel. The right set of four plots display accuracies and predictions made via the same set of methods applied to the $\widetilde{\mathbf{r}}$ values from a single application of SCB with a single level $(L = 1)$ and $m = 100$ measurements.

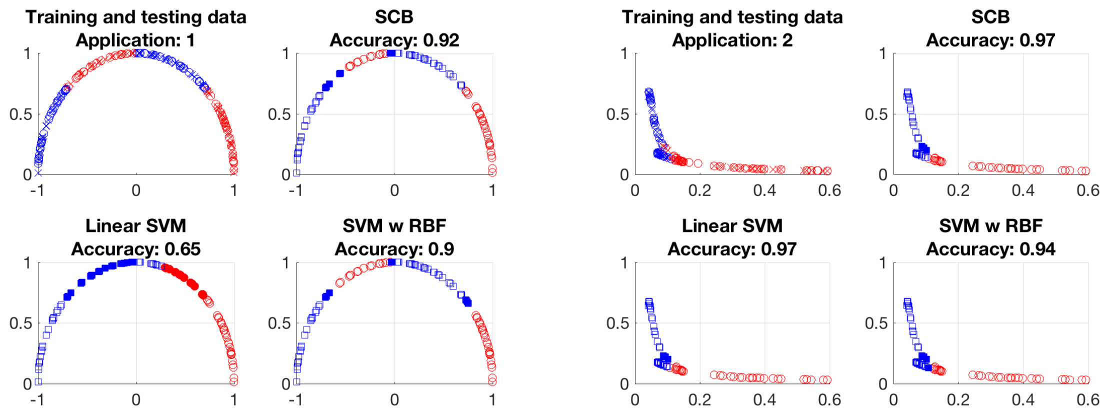


Figure 6.11: The four plots on the left display accuracies and predictions made via various methods for the data given in the upper left-most plot. In the plots of the training and testing data, circles indicate training data and crosses indicate test data. Filled markers indicate that a given method misclassified that particular data point. The methods considered are SCB and SVM with both a linear and RBF kernel. The right set of four plots display accuracies and predictions made via the same set of methods applied to the $\widetilde{\mathbf{r}}$ values from SCB at the first application. $L = 4$ levels and $m = 200$ measurements are used for each application of SCB.

6.7 Conclusion

We have illustrated that iterative applications of SCB of [NSW17] lead to improved classification accuracies as compared to a single application in a variety of settings. Numerical experiments on the MNIST, YaleB, and Norb datasets support this claim. Experiments and theoretical analyses on synthetic data in simple settings demonstrate the effects of multiple iterations on the data and predictions. These examples also highlight simple situations in which the ISCB framework excels. We also demonstrate that an application of SCB can be used as a dimension reduction or data preprocessing technique to improve the performance of other classification methods such as SVM.

Acknowledgments

The authors would like to thank Tina Woolf for help with the initial code used for the SCB method. The authors are grateful to and were partially supported by NSF CAREER DMS #1348721 and NSF BIGDATA DMS #1740325.

6.A Detailed calculations for Subsection 6.5.3

In this section, we provide details for calculating Equation (6.6). Let

$$K_{11} = \sum_{h=1}^{k_1} \frac{u_h(1-u_h)}{(u_h+1)^2},\ K_{12} = \sum_{h=1}^{k_1} \frac{1-u_h}{(u_h+1)^2},\ K_{21} = \sum_{h=1}^{k_2} \frac{1-u'_h}{(1+u'_h)^2},\ K_{22} = \sum_{h=1}^{k_2} \frac{u'_h(1-u'_h)}{(1+u'_h)^2},$$

where u_h and u'_h are i.i.d. uniformly random variables between zero and one. We can then rewrite Equation (6.5) as

$$\begin{aligned}\cos(\theta) &\leq \frac{(j+K_{11}+K_{21})(K_{11}+K_{21})+(j+K_{12}+K_{22})(K_{12}+K_{22})}{j^2}\\ &= \frac{j(K_{11}+K_{21}+K_{12}+K_{22})+K_{11}^2+2K_{11}K_{21}+K_{21}^2+K_{12}^2+2K_{12}K_{22}+K_{22}^2}{j^2}. \quad (6.8)\end{aligned}$$

We then require the expectation of each term in the numerator. Since u_h and u'_h are i.i.d., $\mathbb{E}K_{11}K_{21} = \mathbb{E}K_{11}\mathbb{E}K_{21}$. Straightforward integral calculations lead to the following expected values:

$$\mathbb{E}\left(\frac{u_h(1-u_h)}{(u_h+1)^2}\right) = 3\log 2 - 2, \qquad \mathbb{E}\left(\frac{1-u_h}{(u_h+1)^2}\right) = 1 - \log 2,$$
$$\mathbb{E}\left(\frac{u_h^2(1-u_h)^2}{(u_h+1)^4}\right) = 25/6 - 6\log 2, \qquad \mathbb{E}\left(\frac{(1-u_h)^2}{(u_h+1)^4}\right) = 1/6.$$

We then have the following expectations:

$$\begin{aligned}
\mathbb{E}K_{11} &= k_1(3\log 2 - 2)\\
\mathbb{E}K_{12} &= k_1(1-\log 2)\\
\mathbb{E}K_{11}^2 &= k_1(k_1-1)(3\log 2 - 2)^2 + k_1(25/6 - 6\log 2)\\
\mathbb{E}K_{12}^2 &= k_1(k_1-1)(1-\log 2)^2 + k_1(1/6).
\end{aligned}$$

$\mathbb{E}K_{22}, \mathbb{E}K_{21}, \mathbb{E}K_{22}^2$, and $\mathbb{E}K_{21}^2$ take the same forms with k_2 replacing k_1.

Taking the expectation of Equation (6.8),

$$\begin{aligned}
\mathbb{E}(\cos(\theta)) \le\ & \frac{(k_1+k_2)(2\log 2 - 1)}{j}\\
&+ \frac{(k_1^2 - k_1 + k_2^2 - k_2)(3\log 2 - 2)^2 + (k_1^2 - k_1 + k_2^2 - k_2)(1-\log 2)^2}{j^2}\\
&+ \frac{4k_1k_2(1-\log 2)(3\log 2 - 2) + (k_1+k_2)(1/6 + 25/6 - 6\log 2)}{j^2}\\
\le\ & \frac{(k_1+k_2)(2\log 2 - 1)}{j}\\
&+ \frac{(k_1^2 - k_1 + k_2^2 - k_2)(10(\log 2)^2 - 14\log 2 + 5)}{j^2}\\
&+ \frac{4k_1k_2(1-\log 2)(3\log 2 - 2) + (k_1+k_2)(13/3 - 6\log 2)}{j^2}\\
\le\ & \frac{(k_1+k_2)(2\log 2 - 1)}{j} + \frac{(k_1^2 + k_2^2)(10(\log 2)^2 - 14\log 2 + 5)}{j^2}\\
&+ \frac{4k_1k_2(1-\log 2)(3\log 2 - 2) + (k_1+k_2)(-2/3 + 8\log 2 - 10(\log 2)^2)}{j^2},
\end{aligned}$$

providing the desired bound.

CHAPTER 7

Summary

With the current magnitude and prevalence of data collection and analysis, it is important to have methods that are well-understood and can scale to handle large amounts of data. Additionally, a wide range of tasks and underlying subroutines must be considered. Related topics discussed here include inferring missing data entries, optimization, and classification for compressed data.

For data in which entries with value zero are more likely to be missing, we propose a regularized matrix completion method to infer the missing entries. Experiments on synthetic and real data show that the regularized variant improves recovery in such settings.

We analyze a gradient descent method for learning the maximally separating hyperplane for support vector machines. While other strategies exhibit faster convergence guarantees, gradient descent is a simple strategy and its variants are extremely popular in practice. We provide an analysis of the convergence of gradient descent when applied to the non-smooth hinge loss so that it can be readily compared to other methods of learning maximally separating hyperplanes. We analyze the convergence rates of adaptive sampling strategies for sketch-and-project methods and show that greedy adaptive sampling strategies can lead to improved performance.

Finally, we consider the classification of binary data and extend a recently developed method to perform hierarchical classification and propose an iterative extension to improve accuracy. We show that the method can be adapted to hierarchical classification in a way that reduces computational cost by taking advantage of classes that are easier to identify. We also show that the proposed iterative extension leads to improved accuracy and provide theoretical analysis for simple synthetic data.

Developing methods that can be well-understood mathematically and developing mathematics to understand the methods that are used in practice are key to having useful and trustworthy tools that can be used to understand complex and large data sources.

REFERENCES

[AG18] Brahim Khalil Abid and Robert M. Gower. "Greedy stochastic algorithms for entropy-regularized optimal transport problems." In *Proceedings of the 21th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 2018.

[ALS15] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. "Variance reduction in SGD by distributed importance sampling." *arXiv preprint arXiv:1511.06481*, 2015.

[ASS96] Pervez M Aziz, Henrik V Sorensen, and J Vn der Spiegel. "An overview of sigma-delta converters." *IEEE signal processing magazine*, **13**(1):61–84, 1996.

[BB11] L. Bottou and O. Bousquet. "The Tradeoffs of Large-Scale Learning." *Optimization for Machine Learning*, p. 351, 2011.

[BBH17] Solon Barocas, Elizabeth Bradley, Vasant Honavar, and Foster Provost. "Big Data, Data Science, and Civil Rights." *arXiv preprint arXiv:1706.03102*, 2017.

[BCN18] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning." *SIAM Review*, **60**(2):223–311, 2018.

[BGM18] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. "SGD Learns Over-parameterized Networks that Provably Generalize on Linearly Separable Data." In *International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[BHB07] Robert E Banfield, Lawrence O Hall, Kevin W Bowyer, and W Philip Kegelmeyer. "A comparison of decision tree ensemble creation techniques." *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):173–180, 2007.

[BK07] Robert M Bell and Yehuda Koren. "Lessons from the Netflix prize challenge." *Acm Sigkdd Explorations Newsletter*, **9**(2):75–79, 2007.

[BL07] James Bennett and Stan Lanning. "The Netflix prize." In *Proceedings of KDD cup and workshop*, volume 2007, p. 35. New York, NY, USA, 2007.

[BLW06] Pratik Biswas, Tzu-Chen Lian, Ta-Chung Wang, and Yinyu Ye. "Semidefinite programming based algorithms for sensor network localization." *ACM Trans. Sensor Networks (TOSN)*, **2**(2):188–220, 2006.

[BN15] Jonathan Briskman and Deanna Needell. "Block Kaczmarz method with inequalities." *Journal of Mathematical Imaging and Vision*, **52**(3):385–396, 2015.

[BS99] Peter Bartlett and John Shawe-Taylor. "Generalization performance of support vector machines and other pattern classifiers." *Advances in Kernel methodssupport vector learning*, pp. 43–54, 1999.

[BS16] Solon Barocas and Andrew D Selbst. "Big data's disparate impact." *Cal. L. Rev.*, **104**:671, 2016.

[Bub14] Sébastien Bubeck. "Convex optimization: Algorithms and complexity." *arXiv e-prints*, p. arXiv:1405.4980, May 2014.

[Bub15] Sébastien Bubeck et al. "Convex optimization: Algorithms and complexity." *Foundations and Trends in Machine Learning*, **8**(3-4):231–357, 2015.

[Buh03] Martin D Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.

[BW18a] Zhong-Zhi Bai and Wen-Ting Wu. "On Greedy Randomized Kaczmarz Method for Solving Large Sparse Linear Systems." *SIAM Journal on Scientific Computing*, **40**(1):A592–A606, 2018.

[BW18b] Zhong-Zhi Bai and Wen-Ting Wu. "On relaxed greedy randomized Kaczmarz methods for solving large sparse linear systems." *Applied Mathematics Letters*, **83**:21 – 26, 2018.

[BW19a] Zhong-Zhi Bai and Wen-Ting Wu. "On greedy randomized coordinate descent methods for solving large linear least-squares problems." *Numerical Linear Algebra with Applications*, **26**(4):e2237, 2019. e2237 nla.2237.

[BW19b] Zhong-Zhi Bai and Wen-Ting Wu. "On partially randomized extended Kaczmarz method for solving large sparse overdetermined inconsistent linear systems." *Linear Algebra and its Applications*, **578**:225 – 250, 2019.

[CCS17] P. Chaudhari, Anna Choromanska, S. Soatto, Yann LeCun, C. Baldassi, C. Borgs, J. Chayes, Levent Sagun, and R. Zecchina. "Entropy-SGD: Biasing gradient descent into wide valleys." In *International Conference on Learning Representations*, 2017.

[Cha07] Olivier Chapelle. "Training a support vector machine in the primal." *Neural Computation*, **19**:1155–1178, 2007.

[CHH06] Deng Cai, Xiaofei He, Jiawei Han, and Hong-Jiang Zhang. "Orthogonal Laplacianfaces for Face Recognition." *IEEE Transactions on Image Processing*, **15**(11):3608–3614, 2006.

[CHH07a] Deng Cai, Xiaofei He, and Jiawei Han. "Spectral Regression for Efficient Regularized Subspace Learning." In *Proc. Int. Conf. Computer Vision (ICCV'07)*, 2007.

[CHH07b] Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas Huang. "Learning a Spatially Smooth Subspace for Face Recognition." In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*, 2007.

[CLM11] E. J. Candés, X. Li, Y. Ma, and J. Wright. "Robust Principal Component Analysis?" *J. of the ACM*, **58**(1):1–37, 2011.

[COL04] Sungmoon Cheong, Sang Hoon Oh, and Soo-Young Lee. “Support vector machines with binary tree architecture for multi-class classification.” *Neural Information Processing-Letters and Reviews*, **2**(3):47–51, 2004.

[CP10] E. J. Candès and Y. Plan. “Matrix completion with noise.” *Proceedings of the IEEE*, **9**(6):925–936, 2010.

[CPS18] Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron C. Courville, and Yoshua Bengio. “On the learning dynamics of deep neural networks.” *CoRR*, **abs/1809.06848**, 2018.

[CQR15] Dominik Csiba, Zheng Qu, and Peter Richtárik. “Stochastic Dual Coordinate Ascent with Adaptive Probabilities.” *International Conferences on Machine Learning*, 2015.

[CR09] E. J. Candès and B. Recht. “Exact matrix completion via convex optimization.” *Found. Comput. Math.*, **9**(6):717–772, 2009.

[CR18] Dominik Csiba and Peter Richtárik. “Importance Sampling for Minibatches.” *The Journal of Machine Learning Research*, **19**(1):962–982, 2018.

[CT10] Emmanuel J Candès and Terence Tao. “The power of convex relaxation: Near-optimal matrix completion.” *IEEE Trans. Inform. Theory*, **56**(5):2053–2080, 2010.

[Cut13] Marco Cuturi. “Sinkhorn Distances: Lightspeed Computation of Optimal Transport.” In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pp. 2292–2300. Curran Associates, Inc., 2013.

[CV95] Corinna Cortes and Vladimir Vapnik. “Support-vector networks.” *Machine learning*, **20**(3):273–297, 1995.

[CW08] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning.” In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. ACM, 2008.

[Cyb89] George Cybenko. “Approximation by superpositions of a sigmoidal function.” *Mathematics of control, signals and systems*, **2**(4):303–314, 1989.

[Dev86] Luc Devroye. *Non-uniform random variate generation.* Springer-Verlag, New York, 1986.

[DGL89] Iain S Duff, Roger G Grimes, and John G Lewis. “Sparse matrix test problems.” *ACM Transactions on Mathematical Software (TOMS)*, **15**(1):1–14, 1989.

[DGL92] Iain S Duff, Roger G Grimes, and John G Lewis. “Users’ guide for the Harwell-Boeing sparse matrix collection (Release I).” 1992.

[DH11] Timothy A. Davis and Yifan Hu. "The University of Florida Sparse Matrix Collection." *ACM Transactions on Mathematical Software*, **38**(1):1–25, 2011.

[DS16] Dominique Duncan and Thomas Strohmer. "Classification of alzheimer s disease using unsupervised diffusion component analysis." *Mathematical Biosciences and Engineering*, **13**:1119–1130, 2016.

[Dum15] Bogdan Dumitrescu. "On the relation between the randomized extended Kaczmarz algorithm and coordinate descent." *BIT Numerical Mathematics*, **55**(4):1005–1015, Dec 2015.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[FS97] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of computer and system sciences*, **55**(1):119–139, 1997.

[FSA99] Yoav Freund, Robert Schapire, and Naoki Abe. "A short introduction to boosting." *Journal-Japanese Society For Artificial Intelligence*, **14**(771-780):1612, 1999.

[FSL14] Jun Fang, Yanning Shen, Hongbin Li, and Zhi Ren. "Sparse signal recovery from one-bit quantized data: An iterative reweighted algorithm." *Signal Processing*, **102**:201–206, 2014.

[GBH70] R. Gordon, R. Bender, and G. T. Herman. "Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography." *Journal of Theoretical Biology*, **29**:471–481, 1970.

[GIM99] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. "Similarity search in high dimensions via hashing." In *Vldb*, volume 99, pp. 518–529, 1999.

[GLS18] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. "Characterizing implicit bias in terms of optimization geometry." In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1832–1841, Stockholmsmssan, Stockholm Sweden, 2018. PMLR.

[GMM19] R. Gower, D. Molitor, J. Moorman, and D. Needell. "Adaptive Sketch-and-Project." 2019. Submitted.

[GNR10] A. Gupta, R. Nowak, and B. Recht. "Sample complexity for 1-bit compressed sensing and sparse classification." In *International Symposium on Information Theory (ISIT)*. IEEE, 2010.

[GO12] Michael Griebel and Peter Oswald. "Greedy and randomized versions of the multiplicative Schwarz method." *Linear Algebra and its Applications*, **437**(7):1596–1610, 2012.

[Gor87] Allan D Gordon. "A review of hierarchical classification." *Journal of the Royal Statistical Society. Series A (General)*, pp. 119–137, 1987.

[GP08] Gregory Griffin and Pietro Perona. "Learning and using taxonomies for fast visual categorization." In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

[GR15a] Robert M. Gower and Peter Richtárik. "Stochastic Dual Ascent for Solving Linear Systems." *arXiv:1512.06890*, 2015.

[GR15b] Robert Mansel Gower and Peter Richtárik. "Randomized Iterative Methods for Linear Systems." *SIAM Journal on Matrix Analysis and Applications*, **36**(4):1660–1690, 2015.

[GR16] R.M. Gower and P. Richtárik. "Linearly Convergent Randomized Iterative Methods for Computing the Pseudoinverse." *arXiv preprint arXiv:1612.06255*, 2016.

[GR17] Robert M Gower and Peter Richtárik. "Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms." *SIAM Journal on Matrix Analysis and Applications*, **38**(4):1380–1409, 2017.

[Gro11] D. Gross. "Recovering low-rank matrices from few coefficients in any basis." *IEEE Trans. Inform. Theory*, **57**(3):1548–1566, 2011.

[GSC02] Shantanu Godbole, Sunita Sarawagi, and Soumen Chakrabarti. "Scaling multi-class support vector machines using inter-class confusion." In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 513–518. ACM, 2002.

[HFK04] Roger Higdon, Norman L Foster, Robert A Koeppe, Charles S DeCarli, William J Jagust, Christopher M Clark, Nancy R Barbas, Steven E Arnold, R Scott Turner, Judith L Heidebrink, et al. "A comparison of classification methods for differentiating fronto-temporal dementia from Alzheimer's disease using FDG-PET imaging." *Statistics in medicine*, **23**(2):315–326, 2004.

[HHS17] Elad Hoffer, Itay Hubara, and Daniel Soudry. "Train longer, generalize better: closing the generalization gap in large batch training of neural networks." In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

[HMN19] J. Haddock, D. Molitor, D. Needell, S. Sambandam, J. Song, and S. Sun. "On inferences from Completed Data." *Proc. Sampling Theory and Applications*, 2019.

[HN19] Jamie Haddock and Deanna Needell. "On Motzkin's method for inconsistent linear systems." *BIT Numerical Mathematics*, **59**(2):387–401, Jun 2019.

[HRS16] Moritz Hardt, Benjamin Recht, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent." In *International Conference on Machine Learning*, ICML'16, pp. 1225–1234. JMLR.org, 2016.

[HRT04] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. "The entire regularization path for the support vector machine." *Journal of Machine Learning Research*, **5**(Oct):1391–1415, 2004.

[HS52] M. R. Hestenes and E. Stiefel. "Methods of Conjugate Gradients for Solving Linear Systems." *Journal of Research of the National Bureau of Standards*, **49**(6):409–436, 1952.

[HYH05] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. "Face Recognition Using Laplacianfaces." *IEEE Trans. Pattern Anal. Mach. Intelligence*, **27**(3):328–340, 2005.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[IM98] Piotr Indyk and Rajeev Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 604–613, New York, NY, USA, 1998. ACM.

[JLB11] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk. "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors." *IEEE T. Inform. Theory*, **59**(4):2082–2102, 2011.

[JLB13] Laurent Jacques, Jason Laska, Petros Boufounos, and Richard Baraniuk. "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors." **59**(4):2082–2102, 2013.

[JZ13] Rie Johnson and Tong Zhang. "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction." In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

[Kac37] M S Kaczmarz. "Angenäherte Auflösung von Systemen linearer Gleichungen." *Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques*, **35**:355–357, 1937.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer*, **42**(8), 2009.

[Kem81] AW Kemp. "Efficient generation of logarithmically distributed pseudo-random variables." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **30**(3):249–253, 1981.

[KF18] Angelos Katharopoulos and Franois Fleuret. "Not All Samples Are Created Equal: Deep Learning with Importance Sampling." In *International Conference on Machine Learning*, 2018.

[LBB98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature*, **521**(7553):436, 2015.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database." *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, **2**, 2010.

[LH15] Ilya Loshchilov and Frank Hutter. "Online batch selection for faster training of neural networks." *arXiv preprint arXiv:1511.06343*, 2015.

[LHB04] Yann LeCun, Fu Jie Huang, and Leon Bottou. "Learning methods for generic object recognition with invariance to pose and lighting." In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pp. II–104. IEEE, 2004.

[LHN17] J. A. De Loera, J. Haddock, and D. Needell. "A Sampling Kaczmarz-Motzkin Algorithm for Linear Feasibility." *SIAM Journal on Scientific Computing*, **39**(5), 2017.

[LL10] Dennis Leventhal and Adrian S Lewis. "Randomized methods for linear constraints: convergence rates and conditioning." *Mathematics of Operations Research*, **35**(3):641–654, 2010.

[LRZ12] Xiao Liang, Xiang Ren, Zhengdong Zhang, and Yi Ma. "Repairing sparse low-rank texture." *Computer Vision–ECCV 2012*, pp. 482–495, 2012.

[LS18] Yuanzhi Li and Yoram Singer. "The Well Tempered Lasso." *arXiv preprint arXiv:1806.03190*, 2018.

[LSB12] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. "A simpler approach to obtaining an O(1/t) convergence rate for the projected stochastic subgradient method." *arXiv preprint arXiv:1212.2002*, 2012.

[LT92] Zhi-Quan Luo and Paul Tseng. "On the convergence of the coordinate descent method for convex differentiable minimization." *Journal of Optimization Theory and Applications*, **72**(1):7–35, 1992.

[LV09] Zhang Liu and Lieven Vandenberghe. "Interior-point method for nuclear norm approximation with application to system identification." *SIAM J. Matrix Analysis and Appl.*, **31**(3):1235–1256, 2009.

[LWY11] Jason N Laska, Zaiwen Wen, Wotao Yin, and Richard G Baraniuk. "Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements." **59**(11):5289–5301, 2011.

[LX15] Zhaosong Lu and Lin Xiao. “On the complexity analysis of randomized block-coordinate descent methods.” *Mathematical Programming*, **152**(1-2):615–642, 2015.

[LZO07] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. “Hierarchical document classification using automatically generated hierarchy.” *Journal of Intelligent Information Systems*, **29**(2):211–230, 2007.

[MKS93] Sreerama K Murthy, Simon Kasif, Steven Salzberg, and Richard Beigel. “OC1: A randomized algorithm for building oblique decision trees.” In *Proceedings of AAAI*, volume 93, pp. 322–327. Citeseer, 1993.

[MKS94] Sreerama K Murthy, Simon Kasif, and Steven Salzberg. “A system for induction of oblique decision trees.” *Journal of artificial intelligence research*, **2**:1–32, 1994.

[MN18a] D. Molitor and D. Needell. “Hierarchical Classification using Binary Data.” *AAAI Magazine special Issue on Deep Models, Machine Learning and Artificial Intelligence Applications in National and International Security*, 2018.

[MN18b] D. Molitor and D. Needell. “Matrix Completion for Structured Observations.” 2018.

[MN20] D. Molitor and D. Needell. “An iterative method for classification of binary data.” *Information and Inference*, 2020. To appear.

[MNR15a] Anna Ma, Deanna Needell, and Aaditya Ramdas. “Convergence properties of the randomized extended Gauss–Seidel and Kaczmarz methods.” *SIAM Journal on Matrix Analysis and Applications*, **36**(4):1590–1604, 2015.

[MNR15b] Anna Ma, Deanna Needell, and Aaditya Ramdas. “Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods.” *SIAM Journal on Matrix Analysis and Applications*, **36**(4):1590–1604, 2015.

[MNW19] D. Molitor, D. Needell, and R. Ward. “Bias of Homotopic Gradient Descent for the Hinge Loss.” *Applied Mathematics and Optimization*, 2019. To appear.

[MS54] Theodore Samuel Motzkin and Isaac Jacob Schoenberg. “The relaxation method for linear inequalities.” *Canadian Journal of Mathematics*, **6**:393–404, 1954.

[Nat01] F. Natterer. *The Mathematics of Computerized Tomography*, volume 32 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Reprint of the 1986 original.

[Nec19] Ion Necoara. “Faster randomized block Kaczmarz algorithms.” *arXiv e-prints*, p. arXiv:1902.09946, Feb 2019.

[Nes12] Yu Nesterov. “Efficiency of coordinate descent methods on huge-scale optimization problems.” *SIAM Journal on Optimization*, **22**(2):341–362, 2012.

[NLG19] Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Savarese, Nathan Srebro, and Daniel Soudry. "Convergence of gradient descent on separable data." *International Conference on Artificial Intelligence and Statistics*, 2019.

[NNG17] Ion Necoara, Yurii Nesterov, and François Glineur. "Random block coordinate descent methods for linearly constrained optimization over networks." *Journal of Optimization Theory and Applications*, **173**(1):227–254, 2017.

[NS17] Yurii Nesterov and Sebastian U Stich. "Efficiency of the accelerated coordinate descent method on structured optimization problems." *SIAM Journal on Optimization*, **27**(1):110–123, 2017.

[NSL15] Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. "Coordinate descent converges faster with the Gauss-Southwell rule than random selection." In *International Conference on Machine Learning*, pp. 1632–1641, 2015.

[NSL16] J. Nutini, B. Sepehry, I. Laradji, M. Schmidt, H. Koepke, and A. Virani. "Convergence Rates for Greedy Kaczmarz Algorithms, and Faster Randomized Kaczmarz Rules Using the Orthogonality Graph." *Conference on Uncertainty in Artificial Intelligence*, 2016.

[NSS19] Mor Shpigel Nacson, Nathan Srebro, and Daniel Soudry. "Stochastic Gradient Descent on Separable Data: Exact Convergence with a Fixed Learning Rate." In *Proceedings of Machine Learning Research*, volume 89, pp. 3051–3059. PMLR, 2019.

[NSW15] Deanna Needell, Nathan Srebro, and Rachel Ward. "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm." *Mathematical Programming*, **155**(1):549–573, 2015.

[NSW17] D. Needell, R. Saab, and T. Woolf. "Simple Classification using Binary Data." 2017. Submitted.

[NT14] Deanna Needell and Joel A Tropp. "Paved with good intentions: analysis of a randomized block Kaczmarz method." *Linear Algebra and its Applications*, **441**:199–221, 2014.

[NTS14] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. "In search of the real inductive bias: On the role of implicit regularization in deep learning." *arXiv preprint arXiv:1412.6614*, 2014.

[NZZ15] Deanna Needell, Ran Zhao, and Anastasios Zouzias. "Randomized block Kaczmarz method with projection for solving least squares." *Linear Algebra and its Applications*, **484**:322–343, 2015.

[OAL16] Anton Osokin, Jean-Baptiste Alayrac, Isabella Lukasewitz, Puneet Dokania, and Simon Lacoste-Julien. "Minding the Gaps for Block Frank-Wolfe Optimization of Structured SVMs." In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 593–602, 2016.

[PCJ17] Dmytro Perekrestenko, Volkan Cevher, and Martin Jaggi. "Faster Coordinate Descent via Adaptive Importance Sampling." *arXiv preprint arXiv:1703.02518*, 2017.

[PJM19] Vivak Patel, Mohammad Jahangoshahi, and Daniel Adrian Maldonado. "An Implicit Representation and Iterative Solution of Randomly Sketched Linear Systems.", 2019.

[PKL17] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. "Theory of deep learning III: Explaining the non-overfitting puzzle." *arXiv preprint arXiv:1801.00173*, 2017.

[PLM18] Tomaso Poggio, Qianli Liao, Brando Miranda, Andrzej Banburski, Xavier Boix, and Jack Hidary. "Theory IIIb: Generalization in deep networks." *arXiv preprint arXiv:1806.11379*, 2018.

[PMD16] Executive Office of the President, Cecilia Munoz, Domestic Policy Council Director, Megan (US Chief Technology Officer Smith (Office of Science, Technology Policy)), DJ (Deputy Chief Technology Officer for Data Policy, Chief Data Scientist Patil (Office of Science, and Technology Policy)). *Big data: A report on algorithmic systems, opportunity, and civil rights.* Executive Office of the President, 2016.

[Pop99] Constantin Popa. "Characterization of the solutions set of inconsistent least-squares problems by an extended Kaczmarz algorithm." *Korean Journal of Computational and Applied Mathematics*, **6**(1):51–64, 1999.

[PP16] Stefania Petra and Constantin Popa. "Single projection Kaczmarz extended algorithms." *Numerical Algorithms*, **73**(3):791–806, 2016.

[Rec11] Benjamin Recht. "A simpler approach to matrix completion." *J. Machine Learning Research*, **12**(Dec):3413–3430, 2011.

[RFP10] B. Recht, M. Fazel, and P. A. Parrilo. "Guaranteed Minimum Rank Solutions to Linear Matrix Equations via Nuclear Norm Minimization." *SIAM Review*, **52**(3):471–501, 2010.

[RT13] Peter Richtárik and Martin Takáč. "Distributed Coordinate Descent Method for Learning with Big Data." *Journal of Machine Learning Research*, 2013.

[RT14] Peter Richtárik and Martin Takáč. "Iteration Complexity of Randomized Block-Coordinate Descent Methods for Minimizing a Composite Function." *Mathematical Programming*, **144**(1):1–38, 2014.

[RT17] Peter Richtárik and Martin Takáč. "Stochastic Reformulations of Linear Systems: Algorithms and Convergence Theory." *arXiv:1706.01108*, 2017.

[RZH04] Saharon Rosset, Ji Zhu, and Trevor J Hastie. "Margin maximizing loss functions." In *Advances in Neural Information Processing Systems*, pp. 1237–1244, 2004.

[Sch86] Ralph Schmidt. "Multiple emitter location and signal parameter estimation." *IEEE Trans. Antennas and Propagation*, **34**(3):276–280, 1986.

[SF11] Carlos N Silla and Alex A Freitas. "A survey of hierarchical classification across different application domains." *Data Mining and Knowledge Discovery*, **22**(1-2):31–72, 2011.

[SFR17] Daniel Silva-Palacios, Cèsar Ferri, and María José Ramírez-Quintana. "Improving Performance of Multiclass Classification by Inducing Class Hierarchies." *Procedia Computer Science*, **108**:1692–1701, 2017.

[SG02] Joseph L Schafer and John W Graham. "Missing data: our view of the state of the art." *Psychological methods*, **7**(2):147, 2002.

[SHN18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. "The implicit bias of gradient descent on separable data." *The Journal of Machine Learning Research*, **19**(1):2822–2878, 2018.

[Sin08] Amit Singer. "A remark on global positioning from local distances." *Proc. National Academy of Sciences*, **105**(28):9507–9511, 2008.

[SV09] Thomas Strohmer and Roman Vershynin. "A Randomized Kaczmarz Algorithm with Exponential Convergence." *Journal of Fourier Analysis and Applications*, **15**(2):262–278, 2009.

[SZ13] Ohad Shamir and Tong Zhang. "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes." In *International Conference on Machine Learning*, pp. 71–79, 2013.

[Tse90] Paul Tseng. "Dual ascent methods for problems with strictly convex costs and linear constraints: A unified approach." *SIAM Journal on Control and Optimization*, **28**(1):214–242, 1990.

[Vap82] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin, Heidelberg, 1982.

[Vap99] Vladimir Naumovich Vapnik. "An overview of statistical learning theory." *IEEE Transactions on Neural Networks*, **10**(5):988–999, 1999.

[Vap13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[VC74] Vladimir N Vapnik and Alexey J Chervonenkis. "Theory of pattern recognition." 1974.

[VML17] Carles Ventura, David Masip, and Agata Lapedriza. “Interpreting cnn models for apparent personality trait regression.” In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pp. 1705–1713. IEEE, 2017.

[Wal74] A. J. Walker. “New fast method for generating discrete random numbers with arbitrary frequency distributions.” *Electronics Letters*, **10**(8):127–128, April 1974.

[WGC19] G. Wang, G. B. Giannakis, and J. Chen. “Learning ReLU Networks on Linearly Separable Data: Algorithm, Optimality, and Generalization.” *IEEE Transactions on Signal Processing*, **67**(9):2357–2370, 2019.

[WW98] Jason Weston and Chris Watkins. “Multi-class support vector machines.” Technical report, Citeseer, 1998.

[ZBD99] Blaž Zupan, Marko Bohanec, Janez Demšar, and Ivan Bratko. “Learning by discovering concept hierarchies.” *Artificial Intelligence*, **109**(1-2):211–242, 1999.

[ZBH17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding deep learning requires rethinking generalization.” In *International Conference on Machine Learning*, 2017.

[ZF13] Anastasios Zouzias and Nikolaos M Freris. “Randomized extended Kaczmarz for solving least squares.” *SIAM Journal on Matrix Analysis and Applications*, **34**(2):773–793, 2013.

[Zha04] Tong Zhang. “Solving large scale linear prediction problems using stochastic gradient descent algorithms.” In *International Conference on Machine learning*, p. 116. ACM, 2004.

[ZLM14] Kang Zhao, Hongtao Lu, and Jincheng Mei. “Locality Preserving Hashing.” In *AAAI*, pp. 2874–2881, 2014.

[ZWZ17] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. “Interpretable convolutional neural networks.” *arXiv preprint arXiv:1710.00935*, **2**(3):5, 2017.

[ZZ15] Peilin Zhao and Tong Zhang. “Stochastic optimization with importance sampling for regularized loss minimization.” In *International Conference on Machine Learning*, pp. 1–9, 2015.

[ZZ18] Quan-shi Zhang and Song-Chun Zhu. “Visual interpretability for deep learning: a survey.” *Frontiers of Information Technology & Electronic Engineering*, **19**(1):27–39, 2018.