RANDOMIZED NUMERICAL LINEAR ALGEBRA
FOR LARGE-SCALE OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Zachary Frangella

June 2025

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Professor Madeleine Udell) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Professor Aaron Sidford)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Professor Mert Pilanci)

Approved for the Stanford University Committee on Graduate Studies

Abstract

Large-scale optimization problems arising from the big data era pose significant challenges, as traditional algorithms become prohibitively expensive due to their unfavorable scaling with problem size. While first-order methods like gradient descent offer improved scalability, they often struggle with ill-conditioned problems, leading to slow convergence. This thesis addresses this challenge by developing scalable preconditioning techniques that improve the convergence of first-order methods on ill-conditioned problems without sacrificing computational efficiency. We introduce three novel algorithms: Nyström Preconditioned Conjugate Gradients (Nyström PCG) for solving large-scale symmetric positive definite linear systems, NysADMM for composite optimization problems, and SketchySGD, a stochastic second-order method for machine learning tasks. These algorithms leverage randomized numerical linear algebra to efficiently construct preconditioners from low-rank approximations, resulting in methods that are more robust to problem conditioning than existing first-order algorithms. Theoretical analyses and extensive numerical experiments demonstrate the efficacy of these approaches across a range of applications, including ridge regression, kernel ridge regression, and logistic regression, often outperforming state-of-the-art methods. This work bridges the gap between scalability and fast convergence, offering promising directions for tackling large-scale optimization problems in the big data era.

Acknowledgments

I'm very grateful for all the people who supported me throughout my PhD, and have helped make the experience so enjoyable.

Advisor: Madeleine Udell.

I feel extremely fortunate to have been advised by Madeleine during my PhD. Madeleine brings an unmatched enthusiasm to research, which makes her very fun to work with and to discuss interesting ideas with. Working with Madeleine teaches you how to be a world class researcher. She possesses an exceptional gift for clarity when it comes to presenting complex ideas. I am always impressed by how concisely she can present a complex technical idea or represent it in an intuitive picture. I have benefited greatly from this, and have become much better at communicating ideas clearly and simply thanks to her. Madeleine has also been extremely supportive of students development as independent researchers. She allows students to present ideas for projects that interest them, and then helps them craft it into something concrete or suggests interesting ways to improve the idea. This approach helps you grow as researcher, as it allows you to pursue original ideas while still having a good support net if you run into difficulties. Research aside, Madeleine genuinely cares about each of her students as people, and always does her best to help them achieve their goals. Madeleine's mentorship has not only shaped my approach to research but has also equipped me with skills that will be invaluable in my future career. Her emphasis on clear communication and independent thinking has prepared me to tackle complex problems and effectively communicate solutions in both academic and industry settings. She is an excellent collaborator and mentor, and I am very happy to have had her as my advisor.

Orals committee: Dan Iancu, Mert Pilanci, Aaron Sidford, and Joel Tropp.

I appreciate each member serving on the committee and providing their thoughtful feedback.

Dan, it was great being able to TA the CME 307 optimization course for you and Madeleine this past Fall quarter. I am impressed by your deep geometric intuition for many topics in optimization. Like, Madeleine, you have knack for coming up with good figures for representing complicated ideas and making them more digestible for the students. I am very thankful to have had you serve as chair of my committee.

Mert, I appreciate your enthusiasm for research and your creativity at generating new and interesting ideas. It was very fun to have the opportunity to collaborate with you, as I had long enjoyed your work before coming to Stanford.

Aaron, it has been great getting to know you since coming to Stanford, as I greatly admire your research. I always enjoy our conversations and your great paper recommendations, which are always fun reads! I will miss waving to you from the second floor as you walk across the third floor balustrade.

Joel, it has been a great honor to have you as a collaborator. You have incredible intuition for many deep mathematical topics, and like Madeleine are able to communicate them clearly and concisely. Your papers in randomized linear algebra, are what inspired me to work in this area, so I feel very grateful for having been able to collaborate with you.

Collaborators: Tamara Broderick, Michał Dereziński, Theo Diamandis, Mateo Diaz, Ethan Epperly, Shaghayegh Fazliani, Miria Feng, Sachin Garg, Weimu Lei, Lu Lu, Mert Pilanci, Pratik Rathore, Bartolomeo Stellato, Will Stephenson, Jingruo Sun, Joel Tropp, Rob Webber, Jiaming Yang, and Shipu Zhao.

The opportunity to collaborate with so many wonderful colleagues, ranging from Masters and PhD students to senior faculty, has been one of my favorite experiences from my PhD. I've been fortunate to work with some of the best people in applied mathematics, computer science, optimization, and statistics. Without these collaborations, the PhD wouldn't have nearly been as enjoyable, and I would have learned a lot less.

I would like to give a special shout-out to Pratik Rathore and Shipu Zhao, the fellow student collaborators I worked with most during my PhD. Our collaborations have been fruitful, and it has been a pleasure to work with both of you. Aside from work, it has been a lot of fun hanging out with you, you both have excellent senses of humor. I will miss the dinners and fun late night conversations. I'm very fortunate to have you two as friends and collaborators. If the chance comes to collaborate with either of you again in the future, I'd happily take it.

Udell Group labmates: Ya-Chi Chu, Lijun Ding, Shaghayegh Fazliani, Wenzhi Gao, Weimu Lei, Pratik Rathore, Jingruo Sun, Ali Teshnizi, Mike Van Ness, Miaolan Xie, Chengrun Yang, Shipu Zhao, and Yuxuan Zhao.

During my time in Madeleine's lab, I've had the fortune of being surrounded by brilliant peers in a variety of different disciplines. It has been wonderful learning from them in group meetings and offline discussions. I look forward to seeing what direction the lab moves in next!

Faculty mentors at Cornell: David Bindel, Anil Damle, and Alex Townsend.

David, Anil, and Alex form the core of the scientific computing and numerical analysis group at Cornell. I'm fortunate to have been able to learn from all three of them.

I thank David for his inspiring class on matrix computations, which emphasized the interplay of mathematics and intelligent algorithmic thinking, and really shows the elegance of the subject. I'm also thankful for David's service as director of CAM. He did a great job shepherding the students through the tough times of Covid. David also has the most spectacular collection of textbooks in his office, a feat that myself and several other CAMsters hope to match someday.

I am grateful to Anil Damle for his excellent course on Datasparse Matrix Computations, the sequel to David's course. It was Anil who first introduced me to randomized numerical linear algebra in this course, and provided the initial foundation for my PhD research. Anil is a true expert in numerical linear algebra, and it was joy to learn from him about topics such as rank-structured matrices, low-rank matrix completion, and compressed sensing.

I've been very fortunate to have known Alex from the beginning of my PhD. He has always been incredibly encouraging and gave me great advice when I was starting out. In particular, I first learned about Madeleine from him. Alex is also a spectacular mathematician and teacher. In my first year I had the great privilege of taking an inspiring class on functional analysis from him. He brought a unique perspective to the course, including non-traditional topics from approximation theory and machine learning. Alex introduced me to kernel learning in this class, a topic I ended up working quite diligently on during my PhD. I am extremely thankful to know Alex, and to have learned so much from him.

Faculty mentors at RPI: Gregor Kovacic, Peter Kramer, Fengyan Li, and Harry McLaughlin.

I thank you all for being such exceptional professors and helping me get to where I am now. I want to give special thanks to Fengyan Li and the late Harry McLaughlin. I thank Fengyan for her excellent mentorship, and for introducing me to research. Harry, I thank you for your incredible generosity, and your support for me as soon as I arrived at RPI. You helped me deal with bureaucratic "red-tape" numerous times. Discussing mathematical and scientific ideas with you was always a joy. I feel very fortunate to have had you as a friend and mentor.

Middle and High school teachers: Seamus Hodgekinson, William Russell, and Robert Weaver

In addition to excellent mentors and teachers at the university level, I've been fortunate to have excellent teachers at the middle and high school level.

I thank Mr. H for making math and science so exciting, and inspiring me to pursue it further. It is no easy task to get eighth graders excited about math and science, but he did, while also making it look effortless.

Mr. Russell for the excellent history and mythology courses and constant encouragement to continue learning outside the classroom. I greatly enjoyed our conversations on everything from military history to finance.

I am grateful to Mr. Weaver for his impeccable taste in literature. The English and Philosophy courses I took with him were great fun, and always encouraged original thinking and creativity, two crucial traits of any good researcher.

Friends: Tricia Agarwal, Catherine Chen, Ya-Chi Chu, Matt Davidow, Phil Doldo, Wenzhi Gao, Mallory Gaspard, Andrew Horning, Katherine Lee, Weimu Lei, Yingxi Li, Jiayi Liu, Yueyang Liu, Si Yi Meng, Shriya Nagpal, Gokul Nair, Parth Nobel, Shawn Ong, Dongping Qi, Gauri Pidatala, Xueye Ping, Pratik Rathore, Lily Reeves, Thomas Reeves, Max Ruth, Tianyi Shi, Jingruo Sun, MingYi Wang, Anders Wikum, Wanqiao Xu, Chengrun Yang, Eva Zhang, Xiyue Zhang, and Shipu Zhao.

I am extremely thankful for the great friends I've had throughout my PhD. They are a major reason for why the PhD has been such a fun experience.

I have had countless good times with the CAM crew at Cornell. From the dinners at Sumo, to the numerous boba hangouts, and the occasional watching of Gokul playing Minecraft in the office. I thank the Reeves for being the best officemates one could ask for. I have had countless great conversations with you both, and lots of laughs, or as Lily would say lots of lols. I will never forget our memorable and completely spontaneous Thanksgiving dinner at Nobu! Shriya, thank you for all the fun conversations, support, and your infallible sense of humor. You always manage to get into the funniest situations. I know I can always count on you for a laugh. Phil and MingYi, I greatly enjoyed our hangouts. Your senses of humor both complement mine. It was great sharing crazy stories about RPI or funny events that arise in TAing. Phil, I hope you achieve your most fervent wish—attending a Lil' Pump concert. Xiyue, thank you for all the fun chats, and your encouragement. It was always enjoyable learning about your research, as it was in an area very different from my own. Mallory, I am incredibly fortunate to have had you as a friend at RPI and Cornell. You bring a positive can do attitude wherever you go, that we can all learn from. I appreciate all your support and encouragement throughout our academic careers. Your continued presence at Cornell made the PhD experience a lot more fun, both for myself, and the rest of the CAMsters

Since moving to Stanford I have met many new wonderful people as well as gotten to know people already in the lab much better! Mike and Tricia, thank you both for the wonderful parties you've hosted at your apartment. I'm glad since moving to Stanford I've gotten to know the both of you better. At Cornell, Covid and being on different ends of the campus had made that difficult. Anders, Eva, and Yingxi, it has been great having three more Cornellians around! Anders, you have been a great roommate these past two years. I will miss hearing your research updates, or you telling me when you made a breakthrough on an interesting problem you've been working on. Eva, I will miss your funny stories, and your reactions to the movies and TV shows Anders and I would show you. Yingxi, I'm so glad you were able to join us at Stanford for your PhD. I have greatly enjoyed the conversations we have had when we have hung out, and always appreciate the positive attitude you bring. Xueye, thank you for all the fun hangouts, conversations, and your boba scouting skills. There is no better way to make a hangout more fun then to pair it with good boba.

Parents: Catherine Frangella and Michael Frangella.

Both of you have constantly supported and encouraged me since I was young. My curiosity and love of reading both stem from the positive learning environment you created for me. I am very fortunate to have such supportive parents. Without your love and support over the years, none of this would have been possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The “Big Data” era poses major challenges for optimization, as it leads to optimization problems of unprecedented size. Even seemingly simple machine learning tasks, such as predicting user click behavior or detecting malicious URLs, can result in optimization problems with decision variables containing millions of dimensions. Traditional optimization algorithms, including Newton’s method for unconstrained problems and interior point methods for constrained problems, typically scale cubically (or worse) with the size of the decision variable. Thus, traditional algorithms become prohibitively expensive on large high-dimensional optimization problems.

In the wake of these scalability challenges, first-order algorithms such as gradient descent have gained popularity due to their favorable per-iteration cost, which scales at most linearly with problem size, and their capacity to leverage the massive parallelism available in modern computing hardware. While first-order methods elegantly address the scalability challenges of big data, they introduce a new problem arising from their convergence properties. Namely, the convergence rate of first-order methods is governed by the problem’s condition number [121], which is determined by the underlying data. For matrices, the condition number is defined as the ratio of its largest to smallest singular value. In the case of smooth strongly convex functions, the condition number is defined as the worst-case condition number of the Hessian matrix across the domain.

The condition number significantly impacts the convergence speed of first-order methods. When the condition number is modest, first-order methods can achieve low to moderate accuracy solutions in reasonable time, which suffices for many large-scale applications, particularly in machine learning [22]. However, a large condition number dramatically slows convergence, making it challenging to reach even modest accuracy.

The sensitivity of first-order algorithms to ill-conditioning is particularly problematic in the context of big data, as large-scale data matrices and Hessians are often ill-conditioned. In particular, they typically exhibit approximate low-rank structure, where a small proportion of large singular values dominate the rest, leading to large condition numbers. Consequently, first-order methods can struggle to efficiently provide acceptable solutions in many real-world scenarios.

While the preceding discussion presents a pessimistic outlook, suggesting that slow convergence is the inevitable price for scalability, this dissertation demonstrates that this trade-off is not fundamental. We develop scalable algorithms that enjoy better practical performance on ill-conditioned problems by leveraging preconditioning, which transforms the problem to a new geometry where the condition number is closer to unity.

Classic algorithms like Newton's method and BFGS employ preconditioning, enabling them to achieve fast local convergence independent of the condition number. However, the various costs associated with the preconditioner are precisely why these methods fail to scale. The key insight we leverage in this thesis is that the source of ill-conditioning in data matrices and Hessians arising in large-scale optimization primarily comes from dominant outlying eigenvalues. Therefore, to improve conditioning, we only need to reduce these dominant eigenvalues.

In this thesis, we apply randomized numerical linear algebra (RandNLA) to efficiently construct preconditioners from randomized low-rank approximations that provably improve problem conditioning. By integrating this technique with existing ideas from optimization, we obtain new scalable algorithms that are much more robust to problem conditioning than existing first-order algorithms in the literature. The approach developed here, helps bridge the gap between scalability and fast convergence, offering a useful set of tools for practitioners, and provides a promising direction for tackling large-scale optimization problems in the big data era.

1.2 Contributions and organization

The core contributions of this thesis are three new algorithms: (i) Nyström Preconditioned Conjugate Gradients (SIMAX '23) (ii) NysADMM (ICML '22), and (iii) SketchySGD (SIMODS '24), which were developed by the author and collaborators in the papers [57, 58, 179]. Each paper is the subject of its own chapter. We summarize the content and contributions of each chapter below:

Chapter 2. We develop the randomized Nyström preconditioner for solving large-scale symmetric positive definite linear systems via preconditioned conjugate gradients (PCG), leading to the Nyström Preconditioned Conjugate Gradients Algorithm (Nyström PCG) In particular, we introduce the idea of constructing a preconditioner from a randomized Nyström approximation of a matrix, which will prove central to algorithms developed in this thesis. We provide a detailed performance analysis of the preconditioner, and show that when appropriately constructed, PCG with the randomized Nyström preconditioner is guaranteed to converge at a fast linear rate, independent of the condition number. A systematic principled way of setting hyperparameters is presented, and experiments on large-scale ridge and kernel ridge regression tasks shows Nyström PCG exhibits improved performance relative to CG and other randomized preconditioning techniques.

Chapter 3. The Alternating Directions Method of Multipliers (ADMM) algorithm is one of the most effective first-order algorithms for large-scale composite optimization. Unfortunately, at each iteration ADMM involves solving an expensive subproblem, that is generally ill-conditioned. Building off of Chapter 2, we apply function linearization to obtain a subproblem that reduces to solving a large symmetric positive-definite linear system. Nyström PCG is then applied to solve this system efficiently. Numerical experiments show that NysADMM can yield $3 - 58\times$ speedups relative to bespoke benchmark algorithms and solvers like GLMNet [60], LIBSVM [29], SAGA [37], and Accelerated Proximal Gradient with restarts [126].

Chapter 4. We introduce SketchySGD, a stochastic second-order method that uses randomized low-rank approximations of the subsampled Hessian to estimate curvature, along with an automated stepsize. Theoretical analysis shows SketchySGD converges linearly to a small ball around the minimum with a fixed stepsize, and outperforms SGD on ill-conditioned least-squares problems. Empirical results on ridge and logistic regression tasks demonstrate that SketchySGD, with default hyperparameters, matches or exceeds the performance of tuned stochastic gradient methods and preconditioned conjugate gradient. Notably, SketchySGD solves an ill-conditioned logistic regression problem with a 840GB data matrix, where competitors fail. SketchySGD's out-of-the-box performance and robustness to ill-conditioning distinguishes it from other methods that require careful tuning and struggle with ill-conditioned problems.

Chapter 5. In this chapter, we summarize the contributions of the thesis and discuss works that extend the ideas developed here. In particular, we discuss extensions to massive scale kernel ridge regression and Gaussian processes, variance reduced algorithms for finite-sum minimization, and extensions of NysADMM to a general convex composite optimization solver.

1.3 Other publications and projects

In addition to the chapters that are the subject of this thesis, I have produced four other publications during my PhD: [155] (NeurIPS '21), [141] (ICML '24, Oral), [56] (JMLR '24), [52] (NeurIPS '24). I also have six preprints in submission: [42], [41], [59], [140], [158], and [139].

Chapter 2

Randomized Nyström Preconditioning

2.1 Motivation

In their elegant 1997 textbook on numerical linear algebra [160], Trefethen and Bau write,

> "In ending this book with the subject of preconditioners, we find ourselves at the philosophical center of the scientific computing of the future... Nothing will be more central to computational science in the next century than the art of transforming a problem that appears intractable into another whose solution can be approximated rapidly. For Krylov subspace matrix iterations, this is preconditioning... we can only guess where this idea will take us."

The next century has since arrived, and one of the most fruitful developments in matrix computations has been the emergence of new algorithms that use randomness in an essential way. This paper explores a topic at the nexus of preconditioning and randomized numerical linear algebra. We will show how to use a randomized matrix approximation algorithm to construct a preconditioner for an important class of linear systems that arises throughout data analysis and scientific computing.

2.1.1 The preconditioner

Consider the regularized linear system

$$(A + \mu I)x = b \quad \text{where } A \in \mathbb{R}^{n \times n} \text{ is symmetric psd and } \mu \geq 0. \tag{2.1}$$

Here and elsewhere, psd abbreviates the term "positive semidefinite." This type of linear system emerges whenever we solve a regularized least-squares problem. We will design a class of preconditioners for the problem Equation (2.1).

Throughout this paper, we assume that we can access the matrix A through matrix–vector products $x \mapsto Ax$, commonly known as *matvecs*. The algorithms that we develop will economize on the number of matvecs, and they may not be appropriate in settings where matvecs are very expensive or there are cheaper ways to interact with the matrix.

For a rank parameter $s \in \mathbb{N}$, the randomized Nyström approximation of A takes the form

$$\hat{A}_{\text{nys}} = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T \quad \text{where } \Omega \in \mathbb{R}^{n\times s} \text{ is standard normal.} \tag{2.2}$$

This matrix provides the best psd approximation of A whose range coincides with the range of the sketch $A\Omega$. The randomness in the construction ensures that $\hat{A}_{\text{nys}}$ is a good approximation to the original matrix A with high probability [108, Sec. 14].

We can form the Nyström approximation with sketch size s, using s matvecs with A, plus some extra arithmetic. See Algorithm 13 for the implementation details.

Given the eigenvalue decomposition $\hat{A}_{\text{nys}} = U\hat{\Lambda}U^T$ of the randomized Nyström approximation, we construct the Nyström preconditioner:

$$P = \frac{1}{\hat{\lambda}_s + \mu} U(\hat{\Lambda} + \mu I)U^T + (I - UU^T). \tag{2.3}$$

In a slight abuse of terminology, we refer to s as the rank of the Nyström preconditioner. The key point is that we can solve the linear system $Py = c$ very efficiently, and the action of P^{-1} dramatically reduces the condition number of the regularized matrix $A_\mu = A + \mu I$.

We propose to use Equation (2.3) in conjunction with the preconditioned conjugate gradient (PCG) algorithm. Each iteration of PCG involves a single matvec with A, and a single linear solve with P. When the preconditioned matrix $P^{-1}A_\mu$ has a modest condition number, the algorithm converges to a solution of Equation (2.1) very quickly. See Algorithm 3 for pseudocode for Nyström PCG.

The idea of using the randomized Nyström approximation to construct the preconditioner in Equation (2.3) was suggested by P.-G. Martinsson in the survey [108, Sec. 17], but it has not been implemented or analyzed. An earlier (folklore) preconditioner with similar motivation uses a partial eigendecomposition to form a preconditioner of the form Equation (2.3); for instance, in [65], this idea is called a "deflating preconditioner". However, as computing an exact partial eigendecomposition is prohibitively expensive for large problems, these deflating preconditioners are rarely used. Randomized numerical linear algebra, such as the randomized Nyström approximation used here, provides the key ingredient to make such a preconditioner practical.

2.1.2 Guarantees

This paper contains the first comprehensive study of the preconditioner Equation (2.3), including theoretical analysis and testing on prototypical problems from data analysis and machine learning. One of the main contributions is a rigorous method for choosing the rank s to guarantee good performance, along with an adaptive rank selection procedure that performs well in practice.

A key quantity in our analysis is the *effective dimension* of the regularized matrix $A + \mu I$. That is,

$$d_{\text{eff}}^{\mu}(A) = \text{tr}\left(A(A+\mu I)^{\dagger}\right) = \sum_{j=1}^{n} \frac{\lambda_j(A)}{\lambda_j(A)+\mu}, \tag{2.4}$$

where $(A + \mu I)^{\dagger}$ is the Moore-Penrose pseudoinverse. Our definition differs slightly from the literature [3, 13] which uses $(A + \mu I)^{-1}$, the definition we use allows for the effective dimension to be defined even when $\mu = 0$, in which case it equals the rank of A. The effective dimension measures the degrees of freedom of the problem after regularization. It may be viewed as a (smoothed) count of the eigenvalues larger than μ. Many real-world matrices exhibit strong spectral decay, so for $\mu > 0$ the effective dimension is typically much smaller than the nominal dimension n. As we will discuss, the effective dimension also plays a role in a number of machine learning papers [3, 9, 13, 33, 95] that consider randomized algorithms for solving regularized linear systems.

Remark 2.1.1. Often when the underlying matrix A is clear from context, we shall omit the dependence upon A in the effective dimension, and simply write d_{eff}^{μ}.

Our theory tells us the randomized Nyström preconditioner P is successful when its rank s is proportional to the effective dimension.

Theorem 2.1.2 (Randomized Nyström Preconditioner)**.** *Let $A \in \mathbb{S}_n^{+}(\mathbb{R})$ be a psd matrix, and write $A_\mu = A + \mu I$ where the regularization parameter $\mu > 0$. Define the effective dimension $d_{\text{eff}}^{\mu}(A)$ as in Equation* (2.4)*. Construct the randomized preconditioner P from Equations* (2.2) *and* (2.3) *with rank parameter $s = 2\lceil 1.5\, d_{\text{eff}}^{\mu}(A)\rceil + 1$. Then the condition number of the preconditioned system satisfies*

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_\mu P^{-1/2})\big] < 28. \tag{2.5}$$

Theorem 2.1.2 is a restatement of Theorem 2.5.1.

Simple probability bounds follow from Equation (2.5) via Markov's inequality. For example,

$$\mathbb{P}\big\{\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \le 56\big\} > 1/2.$$

The main consequence of Theorem 2.1.2 is a convergence theorem for PCG with the randomized Nyström preconditioner.

Corollary 2.1.3 (Nyström PCG: Convergence)**.** *Construct the preconditioner P as in Theorem 2.1.2,* *and condition on the event* $\{\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq 56\}$*. Solve the regularized linear system Equation* (2.1) *using Nyström PCG, starting with an initial iterate* $x_0 = 0$*. After* t *iterations, the relative error* δ_t *satisfies*

$$\delta_t := \frac{\|w_t - w_\star\|_{A_\mu}}{\|w_\star\|_{A_\mu}} < 2 \cdot (0.77)^t \quad \textit{where } A_\mu w_\star = b.$$

The error norm is defined as $\|u\|_{A_\mu}^2 = u^T A_\mu u$*. In particular,* $t \geq \lceil 3.9 \log(2/\epsilon) \rceil$ *iterations suffice to achieve relative error* ϵ*.*

Although Theorem 2.1.2 gives an interpretable bound for the rank s of the preconditioner, we cannot instantiate it without knowledge of the effective dimension. To address this shortcoming, we have designed adaptive methods for selecting the rank in practice (Section 2.5.4).

Finally, as part of our investigation, we will also develop a detailed understanding of Nyström sketch-and-solve, a popular algorithm in the machine learning literature [3,13]. Our analysis highlights the deficiencies of Nyström sketch-and-solve relative to Nyström PCG.

2.1.3 Example: Ridge regression

As a concrete example, we consider the l_2-regularized least-squares problem, also known as ridge regression. This problem takes the form

$$\text{minimize}_{x \in \mathbb{R}^d} \quad \frac{1}{2n}\|Xw - b\|^2 + \frac{\mu}{2}\|w\|^2, \tag{2.6}$$

where $X \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ and $\mu > 0$. By calculus, the solution to Equation (2.6) also satisfies the regularized system of linear equations

$$\left(X^T X + n\mu I\right) w = X^T b. \tag{2.7}$$

A direct method to solve Equation (2.7) requires $\mathcal{O}(nd^2)$ flops, which is prohibitive when n and d are both large. Instead, when n and d are large, iterative algorithms, such as the conjugate gradient method (CG), become the tools of choice. Unfortunately, the ridge regression linear system Equation (2.7) is often very ill-conditioned, and CG converges very slowly.

Nyström PCG can dramatically accelerate the solution of Equation (2.7). As an example, consider the shuttle-rf dataset (Section 2.6.2). The matrix X has dimension $43,300 \times 10,000$, while the preconditioner is based on a Nyström approximation with rank $s = 800$. Figure 2.1 shows the progress of the residual as a function of the iteration count. Nyström PCG converges to machine precision in 13 iterations, while CG stalls.

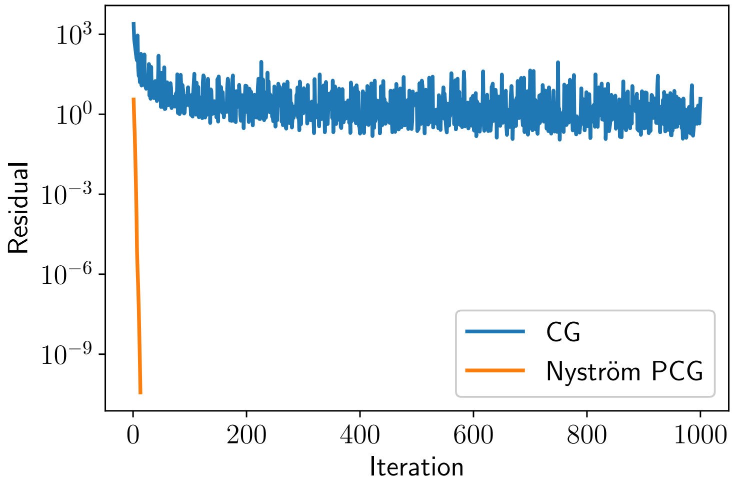


Figure 2.1: **Ridge regression: CG versus Nyström PCG.** For the shuttle-rf data set, Nyström PCG converges to machine precision in 13 iterations while CG stalls. See Sections 2.1.3 and 2.6.2.

2.1.4 Comparison to prior randomized preconditioners

Prior proposals for randomized preconditioners [10, 113, 143] accelerate the solution of highly overdetermined or underdetermined least-squares problems using the sketch-and-precondition paradigm [108, Sec. 10]. For $n \geq d$, these methods require $\Omega(d^3)$ computation to factor the preconditioner. In contrast, the randomized Nyström preconditioner applies to any symmetric positive-definite linear system and can be significantly faster for regularized problems. See Section 2.5.2 more details.

2.1.5 Roadmap

Section 2.2 contains an overview of the Nyström approximation and its key properties. Section 2.3 studies the role of the Nyström approximation in estimating the inverse of the regularized matrix. We analyze the Nyström sketch-and-solve method in Section 2.4, and we give a rigorous performance bound for this algorithm. Section 2.5 presents a full treatment of Nyström PCG, including theoretical results and guidance on numerical implementation. Computational experiments in Section 2.6 demonstrate the power of Nyström PCG for three different applications involving real data sets.

2.1.6 Notation

We write $\mathbb{S}_n(\mathbb{R})$ for the linear space of $n \times n$ real symmetric matrices, while $\mathbb{S}_n^+(\mathbb{R})$ denotes the convex cone of real psd matrices. The symbol $\preceq$ denotes the Loewner order on $\mathbb{S}_n(\mathbb{R})$. That is, $A \preceq B$ if and only if the eigenvalues of $B - A$ are all nonnegative. The function $\operatorname{tr}[\cdot]$ returns the trace of a square matrix. The map $\lambda_j(A)$ returns the jth largest eigenvalue of A; we may omit the matrix if it is clear. As usual, κ_2 denotes the ℓ^2 condition number. We write $\|M\|$ for the spectral norm of a matrix M. For a psd matrix A, we write $\|u\|_A^2 = u^T Au$ for the A-norm. Given $A \in \mathbb{S}_n(\mathbb{R})$ and $1 \leq s \leq n$, the symbol $\lfloor A \rfloor_s$ refers to any best rank-s approximation to A relative to the spectral norm. For $A \in \mathbb{S}_n^+(\mathbb{R})$ and $\mu \geq 0$, the regularized matrix is abbreviated $A_\mu = A + \mu I$. For $A \in \mathbb{S}_n^+(\mathbb{R})$ and $\mu \geq 0$ the effective dimension of A_μ is defined as $d_{\text{eff}}^{\mu}(A) = \text{tr}(A(A+\mu I)^\dagger)$. For $A \in \mathbb{S}_n^+(\mathbb{R})$, the p-stable rank of A is defined as $\text{sr}_p(A) = \lambda_p^{-1} \sum_{j>p}^n \lambda_j$. For $A \in \mathbb{S}_n^+(\mathbb{R})$, we denote the time taken to compute a matvec with A by T_{mv}.

2.2 The Nyström approximation

Let us begin with a review of the Nyström approximation and the randomized Nyström approximation.

2.2.1 Definition and basic properties

The Nyström approximation is a natural way to construct a low-rank psd approximation of a psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$. Let $Z \in \mathbb{R}^{n \times s}$ be an arbitrary test matrix. The *Nyström approximation* of A with respect to the range of X is defined by

$$A\langle Z \rangle = (AZ)(Z^T AZ)^\dagger (AZ)^T \in \mathbb{S}_n^+(\mathbb{R}). \tag{2.8}$$

The Nyström approximation is the best psd approximation of A whose range coincides with the range of AX. It has a deep relationship with the Schur complement and with Cholesky factorization [108, Sec. 14].

The Nyström approximation enjoys several elementary properties that we record in the following lemma.

Lemma 2.2.1. *Let $A\langle Z \rangle \in \mathbb{S}_n^+(\mathbb{R})$ be a Nyström approximation of the psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$. Then*

1. *The approximation $A\langle Z \rangle$ is psd and has rank at most s.*
2. *The approximation $A\langle Z \rangle$ depends only on* range(Z)*, that is*

$$\text{range}(A\langle Z \rangle) \subset \text{range}(Z).$$

3. *In the Loewner order, $A\langle Z \rangle \preceq A$.*
4. *In particular, the eigenvalues satisfy $\lambda_j(A\langle Z \rangle) \leq \lambda_j(A)$ for each $1 \leq j \leq n$.*

The proof of Lemma 2.2.1, Item 3 is not completely obvious. It is a consequence of the fact that we may express $A\langle Z \rangle = A^{1/2} \Pi A^{1/2}$, where Π is an orthogonal projector.

Algorithm 1 RandNysAppx [101, 119, 163]

```
Input: Positive-semidefinite matrix $A \in \mathbb{S}_n^+(\mathbb{R})$, rank $s$
$\Omega = \mathrm{randn}(n, s)$ {Gaussian test matrix}
$\Omega = \mathrm{qr}(\Omega, 0)$ {Thin QR decomposition}
$Y = A\Omega$ {$s$ matvecs with $A$}
$\nu = \mathrm{eps}(\mathrm{norm}(Y, \text{'fro'}))$ {Compute shift}
$Y_\nu = Y + \nu\Omega$ {Shift for stability}
$C = \mathrm{chol}(\Omega^T Y_\nu)$
$B = Y_\nu / C$
$[U, \Sigma, \sim] = \mathrm{svd}(B, 0)$ {Thin SVD}
$\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$ {Remove shift, compute eigs}
Output: Nyström approximation in factored form $\hat{A}_{\text{nys}} = U\hat{\Lambda}U^T$
```

2.2.2 Randomized Nyström approximation

How should we choose the test matrix Z so that the Nyström approximation $A\langle Z\rangle$ provides a good low-rank model for A? Surprisingly, we can obtain a good approximation simply by drawing the test matrix at random. See [163] for theoretical justification of this claim.

Let us outline the construction of the randomized Nyström approximation. Draw a standard normal test matrix $\Omega \in \mathbb{R}^{n\times s}$ where s is the sketch size, and compute the sketch $Y = A\Omega$. By Lemma 2.2.1, the sketch size s is equal to the rank of $\hat{A}_{\text{nys}}$ with probability 1, hence we use these terms interchangeably. The Nyström approximation Equation (2.8) is constructed directly from the test matrix Ω and the sketch Y:

$$\hat{A}_{\text{nys}} = A\langle\Omega\rangle = Y(\Omega^T Y)^\dagger Y^T. \tag{2.9}$$

The formula Equation (2.9) is not numerically sound. We refer the reader to Algorithm 13 for a stable and efficient implementation of the randomized Nyström approximation [101, 119, 163]. Conveniently, Algorithm 13 returns the truncated eigendecomposition $\hat{A}_{\text{nys}} = U\hat{\Lambda}U^T$, where $U \in \mathbb{R}^{n\times s}$ is an orthonormal matrix whose columns are eigenvectors and $\hat{\Lambda} \in \mathbb{R}^{s\times s}$ is a diagonal matrix listing the eigenvalues, which we often abbreviate as $\hat{\lambda}_1, \ldots, \hat{\lambda}_s$.

The randomized Nyström approximation described in this section has a key difference from the Nyström approximations that have traditionally been used in the machine learning literature [3, 13, 66, 169]. In machine learning settings, the Nyström approximation is usually constructed from a sketch Y that samples random columns from the matrix (i.e., the random test matrix Ω has 1-sparse columns). In contrast, Algorithm 13 computes a sketch Y via random projection (i.e., the test matrix Ω is standard normal). In most applications, we have strong reasons (Section 2.2.2) for preferring random projections to column sampling.

Cost of randomized Nyström approximation

Throughout the paper, we write T_{mv} for the time required to compute a matrix–vector product (matvec) with A. Forming the sketch $Y = A\Omega$ with sketch size s requires s matvecs, which costs $T_{\mathrm{mv}}s$. The other steps in the algorithm have arithmetic cost $\mathcal{O}(ns^2)$. Hence, the total computational cost of Algorithm 13 is $\mathcal{O}(T_{\mathrm{mv}}s + s^2n)$ operations. The storage cost is $\mathcal{O}(sn)$ floating-point numbers.

For Algorithm 13, the worst-case performance occurs when A is dense and unstructured. In this case, forming Y costs $\mathcal{O}(n^2s)$ operations. However, if we have access to the columns of A then we may reduce the cost of forming Y to $\mathcal{O}(n^2 \log s)$ by using a structured test matrix Ω, such as a scrambled subsampled randomized Fourier transform (SSRFT) map or a sparse map [108, 163].

A priori guarantees for the randomized Nyström approximation

In this section, we present an a priori error bound for the randomized Nyström approximation. The result improves over previous analyses [66, 67, 163] by sharpening the error terms. This refinement is critical for the analysis of the preconditioner.

Proposition 2.2.2 (Randomized Nyström approximation: Error)**.** *Consider a psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. Choose a sketch size $s \geq 4$, and draw a standard normal test matrix $\Omega \in \mathbb{R}^{n\times s}$. Then the rank-s Nyström approximation $\hat{A}_{\mathrm{nys}}$ computed by Algorithm 13 satisfies*

$$\mathbb{E}\|A - \hat{A}_{\mathrm{nys}}\| \leq \min_{2\leq p\leq s-2} \left[\left(1 + \frac{2(s-p)}{p-1}\right) \lambda_{s-p+1} + \frac{2\mathrm{e}^2 s}{p^2-1} \left(\sum_{j>s-p} \lambda_j \right) \right]. \tag{2.10}$$

The proof of Proposition 2.2.2 may be found in Section 2.8.2.

Proposition 2.2.2 shows that, in expectation, the randomized Nyström approximation $\hat{A}_{\mathrm{nys}}$ provides a good rank-s approximation to A. The first term in the bound is comparable with the spectral-norm error λ_{s-p+1} in the optimal rank-$(s-p)$ approximation, $\lfloor A \rfloor_{s-p}$. The second term in the bound is comparable with the trace-norm error $\sum_{j>s-p} \lambda_j$ in the optimal rank-$(s-p)$ approximation.

Proposition 2.2.2 is better understood via the following simplification.

Corollary 2.2.3 (Randomized Nyström approximation)**.** *Instate the assumptions of Proposition 2.2.2. For $p \geq 2$ and $s = 2p - 1$, we have the bound*

$$\mathbb{E}\|A - \hat{A}_{\mathrm{nys}}\| \leq \left(3 + \frac{4\mathrm{e}^2}{p}\mathrm{sr}_p(A)\right) \lambda_p.$$

The p-stable rank, $\mathrm{sr}_p(A) = \lambda_p^{-1} \sum_{j=p}^n \lambda_j$, reflects decay in the tail eigenvalues.

Corollary 2.2.3 shows that the Nyström approximation error is on the order of λ_p when the rank parameter $s = 2p - 1$. The constant depends on the p-stable rank $\mathrm{sr}_p(A)$, which is small when the tail eigenvalues decay quickly starting at λ_p. This bound is critical for establishing our main results (Theorems 2.4.2 and 2.5.1).

Random projection versus column sampling

Most papers in the machine learning literature [3, 13] construct Nyström approximations by sampling columns at random from an adaptive distribution. In contrast, for most applications, we advocate using an oblivious random projection of the matrix to construct a Nyström approximation.

Random projection has several advantages over column sampling. First, column sampling offers no computational advantage when we only have black-box matvec access to the matrix, while random projections are natural in this setting and possess stronger performance guarantees. Second, it can be very expensive to obtain adaptive distributions for column sampling. Indeed, computing approximate ridge leverage scores costs just as much as solving the ridge regression problem directly using random projections [44, Theorem 2]. Third, even with a good sampling distribution, column sampling produces higher variance results than random projection, so it is far less reliable.

On the other hand, we have found that there are a few applications where it is more effective to compute a randomized Nyström preconditioner using column sampling in lieu of random projections. In particular, this seems to be the case for kernel ridge regression (Section 2.6.5). Indeed, the entries of the kernel matrix are given by an explicit formula, so we can extract full columns with ease. Sampling s columns may cost only $\mathcal{O}(sn)$ operations, whereas a single matvec generally costs $\mathcal{O}(n^2)$. Furthermore, kernel matrices usually exhibit fast spectral decay, which limits the performance loss that results from using column sampling in lieu of random projection.

2.3 Approximating the regularized inverse

Let us return to the regularized linear system Equation (2.1). The solution to the problem has the form $w_\star = (A + \mu I)^{-1} b$. Given a good approximation $\hat{A}$ to the matrix A, it is natural to ask whether $\hat{w} = (\hat{A} + \mu I)^{-1} b$ is a good approximation to the desired solution $w_\star$.

There are many reasons why we might prefer to use $\hat{A}$ in place of A. In particular, we may be able to solve linear systems in the matrix $\hat{A} + \mu I$ more efficiently. On the other hand, the utility of this approach depends on how well the inverse $(\hat{A} + \mu I)^{-1}$ approximates the desired inverse $(A + \mu I)^{-1}$. The next result addresses this question for a wide class of approximations that includes the Nyström approximation.

Proposition 2.3.1 (Regularized inverses)**.** *Consider psd matrices A, $\hat{A} \in \mathbb{S}_n^+(\mathbb{R})$, and assume that the difference $E = A - \hat{A}$ is psd. Fix $\mu > 0$. Then*

$$\left\| (\hat{A} + \mu I)^{-1} - (A + \mu I)^{-1} \right\| \leq \frac{1}{\mu} \frac{\|E\|}{\|E\| + \mu}. \tag{2.11}$$

Algorithm 2 Nyström sketch-and-solve

```
Input: Psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$, right-hand side $b$, regularization $\mu$, rank $s$
$[U, \hat{\Lambda}] = \mathrm{RandNysAppx}(A, s)$
Use Equation (2.12) to compute $\hat{w} = (\hat{A}_{\mathrm{nys}} + \mu I)^{-1} b$
Output: Approximate solution $\hat{w}$ to Equation (2.1)
```

Furthermore, the bound (2.11) *is attained when* $\hat{A} = \lfloor A \rfloor_s$ *for* $1 \leq s \leq n$.

The proof of Proposition 2.3.1 may be found in Section 2.8.4. It is based on [18, Lemma X.1.4].

Proposition 2.3.1 has an appealing interpretation. When $\|A - \hat{A}\|$ is small in comparison to the regularization parameter μ, then the approximate inverse $(\hat{A} + \mu I)^{-1}$ can serve in place of the inverse $(A + \mu I)^{-1}$.

2.4 Nyström sketch-and-solve

The simplest mechanism for using the Nyström approximation is an algorithm called Nyström sketch-and-solve. This section introduces the method, its implementation, and its history. We also provide a general theoretical analysis that sheds light on its performance. In spite of its popularity, the Nyström sketch-and-solve method is rarely worth serious consideration.

2.4.1 Overview

Given a rank-s Nyström approximation $\hat{A}_{\mathrm{nys}}$ of the psd matrix A, it is tempting to replace the regularized linear system $(A + \mu I)w = b$ with the proxy $(\hat{A}_{\mathrm{nys}} + \mu I)w = b$. Indeed, we can solve the proxy linear system in $O(sn)$ time using the Sherman–Morrison–Woodbury formula [68, Eqn. (2.1.4)]:

Lemma 2.4.1 (Approximate regularized inversion)**.** *Consider any rank-s matrix* $\hat{A}$ *with eigenvalue decomposition* $\hat{A} = U \hat{\Lambda} U^T$*. Then*

$$(\hat{A} + \mu I)^{-1} = U(\hat{\Lambda} + \mu I)^{-1} U^T + \frac{1}{\mu}(I - UU^T). \tag{2.12}$$

We refer to the approach in this paragraph as the Nyström sketch-and-solve algorithm because it is modeled on the sketch-and-solve paradigm that originated in [148].

See Algorithm 2 for a summary of the Nyström sketch-and-solve method. The algorithm produces an approximate solution $\hat{w}$ to the regularized linear system Equation (2.1) in time $\mathcal{O}(T_{\mathrm{mv}} s + s^2 n)$. The arithmetic cost is much faster than a direct method, which costs $\mathcal{O}(n^3)$. It can also be faster than running CG for a long time at a cost of $\mathcal{O}(T_{\mathrm{mv}})$ per iteration. The method looks attractive if we only consider the runtime, and yet...

Nyström sketch-and-solve only has one parameter, the rank s of the Nyström approximation, which controls the quality of the approximate solution $\hat{w}$. When $s \ll n$, the method has an appealing computational profile. As s increases, the approximation quality increases but the computational burden becomes heavy. We will show that, alas, an accurate solution to the linear system actually requires $s \approx n$, at which point the computational benefits of Nyström sketch-and-solve evaporate completely.

In summary, Nyström sketch-and-solve is almost never the right algorithm to use. We will see that Nyström PCG generally produces much more accurate solutions with a similar computational cost.

2.4.2 Guarantees and deficiencies

Using Proposition 2.3.1 together with the a priori guarantee in Proposition 2.2.2, we quickly obtain a performance guarantee for Algorithm 2.

Theorem 2.4.2. *Fix $p \geq 2$, and set $s = 2p-1$. For a psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$, construct a randomized Nyström approximation $\hat{A}_{\text{nys}}$ using Algorithm 13. Then the approximation error for the inverse satisfies*

$$\mathbb{E}\big\|(A+\mu I)^{-1} - (\hat{A}_{\text{nys}}+\mu I)^{-1}\big\| \leq \left(3 + \frac{4\mathrm{e}^2}{p}\operatorname{sr}_p(A)\right)\frac{\lambda_p}{\mu \cdot (\lambda_p + \mu)}. \tag{2.13}$$

Define $w_\star = (A+\mu I)^{-1}b$, and select $s = 2\lceil 1.5\, d_{\text{eff}}^{\epsilon\mu}(A)\rceil + 1$. Then the approximate solution $\hat{w}$ computed by Algorithm 2 satisfies

$$\mathbb{E}\left[\frac{\|\hat{w} - w_\star\|_2}{\|w_\star\|_2}\right] \leq 26\epsilon. \tag{2.14}$$

The proof of Theorem 2.4.2 may be found in Section 2.8.5.

Theorem 2.4.2 tells us how accurately we can hope to solve linear systems using Nyström sketch-and-solve (Algorithm 2). A sketch size $s = \mathcal{O}(d_{\text{eff}}^{\epsilon\mu}(A))$ is needed to guarantee relative error ϵ. When $\epsilon\mu$ is small, we anticipate that $d_{\text{eff}}^{\epsilon\mu}(A) \approx n$. In this setting, Nyström sketch-and-solve has no computational value: it is as expensive as a direct method. As a concrete example, let $\mu = 10^{-4}$ and suppose we want six digits of accuracy, i.e., $\epsilon = 10^{-6}$. Then we must hope to find a sketch size s so that $\lambda_s \sim 10^{-10}$ to achieve the required accuracy; and $s \ll n$ so the method offers a computational advantage. It is rare to find a matrix whose spectrum decays rapidly enough to satisfy both these constraints! Our analysis is sharp in its essential respects, so the pessimistic assessment is irremediable.

2.4.3 History

Nyström sketch-and-solve has a long history in the machine learning literature. It was introduced in [169] to speed up kernel-based learning, and it plays a role in many subsequent papers on kernel methods. In this context, the Nyström approximation is typically obtained using column sampling [3,13,169], which has its limitations (Section 2.2.2). More recently, Nyström sketch-and-solve has been applied to speed up approximate cross-validation [157].

The analysis of Nyström sketch-and-solve presented above differs from previous analysis. Prior works [3, 13] focus on the kernel setting, and they use properties of column sampling schemes to derive learning guarantees. In contrast, we bound the relative error for a Nyström approximation based on a random projection. Our overall approach extends to column sampling if we replace Proposition 2.2.2 by an appropriate analog, such as Gittens's results [66].

2.5 Nyström Preconditioned Conjugate Gradients

We now present our main algorithm, Nyström PCG. This algorithm produces high accuracy solutions to a regularized linear system by using the Nyström approximation $\hat{A}_{\mathrm{nys}}$ as a preconditioner. We provide a rigorous estimate for the condition number of the preconditioned system, and we prove that Nyström PCG leads to fast convergence for regularized linear systems. In contrast, we have shown that Nyström sketch-and-solve cannot be expected to yield accurate solutions.

2.5.1 The preconditioner

In this section, we introduce the optimal low-rank preconditioner, and we argue that the randomized Nyström preconditioner provides an approximation that is easy to compute.

Motivation

As a warmup, suppose we knew the eigenvalue decomposition of the best rank-s approximation of the matrix: $\lfloor A \rfloor_s = V_s \Lambda_s V_s^T$. How should we use this information to construct a good preconditioner for the regularized linear system Equation (2.1)?

Consider the family of symmetric psd matrices that act as the identity on the orthogonal complement of range(V_s). Within this class, we claim that the following matrix is the *optimal preconditioner*:

$$P_\star = \frac{1}{\lambda_{s+1} + \mu} V_s(\Lambda_s + \mu I)V_s^T + (I - V_s V_s^T). \tag{2.15}$$

The optimal preconditioner $P_\star$ requires $\mathcal{O}(ns)$ storage, and we can solve linear systems in $P_\star$ in $\mathcal{O}(ns)$ time. Whereas the regularized matrix A_μ has condition number $\kappa_2(A_\mu) = (\lambda_1 + \mu)/(\lambda_n + \mu)$, the preconditioner yields

$$\kappa_2(P_\star^{-1/2} A_\mu P_\star^{-1/2}) = \frac{\lambda_{s+1} + \mu}{\lambda_n + \mu}. \tag{2.16}$$

This is the minimum possible condition number attainable by a preconditioner from the class that we have delineated. It represents a significant improvement when $\lambda_{s+1} \ll \lambda_1$. The proofs of these claims are straightforward; for details, see Section 2.8.6.

Randomized Nyström preconditioner

It is expensive to compute the best rank-s approximation $\lfloor A \rfloor_s$ accurately. In contrast, we can compute the rank-s randomized Nyström approximation $\hat{A}_{\text{nys}}$ efficiently (Algorithm 13). Furthermore, we have seen that $\hat{A}_{\text{nys}}$ approximates A nearly as well as the optimal rank-s approximation (Corollary 2.2.3). These facts lead us to study the randomized Nyström preconditioner, proposed in [108, Sec. 17] without a complete justification.

Consider the eigenvalue decomposition $\hat{A}_{\text{nys}} = U\hat{\Lambda}U^T$, and write $\hat{\lambda}_s$ for its sth eigenvalue. The randomized Nyström preconditioner and its inverse take the form

$$\begin{aligned} P &= \frac{1}{\hat{\lambda}_s + \mu} U(\hat{\Lambda} + \mu I)U^T + (I - UU^T); \\ P^{-1} &= (\hat{\lambda}_s + \mu) U(\hat{\Lambda} + \mu I)^{-1}U^T + (I - UU^T). \end{aligned} \tag{2.17}$$

Like the optimal preconditioner $P_\star$, the randomized Nyström preconditioner (2.17) is cheap to apply and to store. We may hope that it damps the condition number of the preconditioned system $P^{-1/2}A_\mu P^{-1/2}$ nearly as well as the optimal preconditioner $P_\star$. We will support this intuition with a rigorous bound (Proposition 2.5.3).

2.5.2 Nyström PCG

We can obviously use the randomized Nyström preconditioner within the framework of PCG. We call this approach Nyström PCG, and we present a basic implementation in Algorithm 3. In the case of very ill-conditioned least-squares problems, it is sometimes preferable to use other Krylov methods such as LSQR [127] over CG. We have not found the need to use such methods as we focus on regularized problems and are preconditioning, so that $\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \ll u^{-1}$, where u is machine precision, Nevertheless, the Nyström precondtioner is easily extended to LSQR, one may use $P^{-1/2}$ as a right-preconditioner with P as in (2.17).

More precisely, Algorithm 3 uses left-preconditioned CG. This algorithm implicitly works with the unsymmetric matrix $P^{-1}A_\mu$, rather than the symmetric matrix $P^{-1/2}A_\mu P^{-1/2}$. The two variants of PCG yield identical sequences of iterates [147], but the former is more efficient. For ease of analysis, our theoretical results are presented in terms of the symmetrically preconditioned matrix.

Complexity of Nyström PCG

Nyström PCG first constructs the randomized Nyström approximation, and then solves the regularized linear system with PCG. We have already discussed the cost of constructing the Nyström approximation (Section 2.2.2). PCG requires $\mathcal{O}(T_{\text{mv}})$ operations per iteration, and it uses a total of $\mathcal{O}(n)$ additional storage.

For the regularized linear system Equation (2.1), Theorem 2.5.1 and Corollary 2.5.2 demonstrate

Algorithm 3 Nyström PCG

```
Input: Psd matrix $A$, righthand side $b$, initial guess $x_0$, regularization parameter $\mu$, sketch size $s$, solution tolerance $\eta$
$[U, \hat{\Lambda}] = \mathrm{RandNysAppx}(A, s)$
$r_0 = b - (A + \mu I)x_0$
$z_0 = P^{-1}r_0$ {using (2.17)}
$p_0 = z_0$
while $\|r\|_2 > \eta$ do
    $v = (A + \mu I)p_0$
    $\alpha = (r_0^T z_0)/(p_0^T v_0)$ {compute step size}
    $x = x_0 + \alpha p_0$ {update solution}
    $r = r_0 - \alpha v$ {update residual}
    $z = P^{-1}r$ {find search direction via (2.17)}
    $\beta = (r^T z)/(r_0^T z_0)$
    $x_0 \leftarrow x$, $r_0 \leftarrow r$, $p_0 \leftarrow z + \beta p_0$, $z_0 \leftarrow z$
end while
Output: Approximate solution $\hat{x}$ to regularized system Equation (2.1)
```

that it suffices to choose the sketch size $s = 2\lceil 1.5 d_{\text{eff}}^{\mu}(A)\rceil + 1$. In this case with high probability, the overall runtime needed for Nyström PCG to obtain ϵ-relative error in the A_μ-norm is

$$\mathcal{O}\left(d_{\text{eff}}^{\mu}(A)^2 n + T_{\text{mv}}(d_{\text{eff}}^{\mu}(A) + \log(1/\epsilon))\right) \quad \text{operations.}$$

When the effective dimension $d_{\text{eff}}^{\mu}(A)$ is modest, Nyström PCG is very efficient.

In contrast, Section 2.4.2 shows that the running time for Nyström sketch-and-solve has the same form— with $d_{\text{eff}}^{\epsilon\mu}(A)$ in place of $d_{\text{eff}}^{\mu}(A)$. That is, Nyström PCG can produce solutions whose residual norm is close to machine precision, whereas it is impossible to obtain high precision solutions efficiently with Nyström sketch-and-solve.

When is Nyström PCG faster than Conjugate Gradient?

The preceding discussion shows that when the preconditioner is constructed appropriately, Nyström PCG converges at a linear rate, independent of the condition number of A_μ. It is therefore tempting to conclude that when $d_{\text{eff}}^{\mu}(A)$ is modest, Nyström PCG is faster than CG, whose asymptotic cost is

$$\mathcal{O}\left(T_{\text{mv}}\sqrt{\kappa}\log\left(\frac{1}{\epsilon}\right)\right).$$

However, for problems where $d_{\text{eff}}^{\mu}(A)$ is modest, the classic analysis of CG is not sharp. Instead, a tight analysis from [11] combined with Lemma 2.5.4 shows the cost of CG is at most

$$\mathcal{O}\left(T_{\text{mv}} d_{\text{eff}}^{\mu}(A) + \log\left(\frac{1}{\epsilon}\right)\right).$$

In detail, CG runs for $d_{\text{eff}}^{\mu}(A)$ iterations, during which it eliminates the eigenvalues larger than μ. Once these large outlying eigenvalues have been eliminated, the remaining eigenvalues are highly clustered, so CG converges at a fast linear rate independent of the condition number.

Thus, when using a Gaussian test matrix to construct the preconditioner, Nyström PCG is not asymptotically faster than CG. Nevertheless, it is often faster in practice, owing to how the computation is organized. The computation of the sketch $A\Omega$ is performed as a matrix-matrix product or in batch, which exploits the massive parallelism of modern computing architectures. In contrast, CG cannot batch its matvecs, as it is inherently sequential—it computes one matvec per iteration. Consequently, even though both algorithms use the same number of matvecs asymptotically, Nyström PCG can be significantly faster in practice. The situation is similar to the LSRN solver [113] for highly overdetermined and underdetermined least-squares problems. When the data matrix is highly overdetermined, [113] uses a Gaussian sketch with a sketch size proportional to the number of columns—even though in exact arithmetic CG produces the exact solution when it is run for this many iterations. Thus, LSRN does not improve over CG asymptotically in the number of matvecs, but it is faster in practice thanks to parallelism and how it organizes the computation.

Figure 2.2 gives an example of the power of parallelism with the covtype dataset from LIBSVM [29]. A random features transform is performed, leading to a design matrix X of size $(581012, 5000)$. The ridge regression problem with regularization $n\mu = 1$ is then solved via CG and Nyström PCG with a rank $r = 500$ preconditioner. Note that to compute the Nyström preconditioner, Nyström PCG performs 1000 matvecs. To reach the required tolerance, Nyström PCG incurs an additional 80 matvecs for 80 iterations in PCG, for a total matvec expenditure of 1080. In contrast, to solve this problem, CG expends 1760 matvecs. Thus, the number of matvecs used is comparable in terms of orders of magnitude. Despite this, Figure 2.2 shows that Nyström PCG runs 20× faster than CG on this example, which demonstrates the power of parallelism.

Finally, it is worth noting that there are settings where Nyström PCG exhibits better asymptotic performance than the sharp analysis of CG from [11]. When we have entrywise access to A, the cost of the sketch can be reduced to $\mathcal{O}(n^2 \log(d_{\text{eff}}^{\mu}(A)))$ by using a structured sketching map [40]. This implies a total cost of $\mathcal{O}\left(n^2 \log\left(\frac{d_{\text{eff}}^{\mu}(A)}{\epsilon}\right) + n d_{\text{eff}}^{\mu}(A)^2\right)$, which offers a significant improvement over the refined analysis of CG when $d_{\text{eff}}^{\mu}(A) = \mathcal{O}(\sqrt{n})$.

Comparison to other randomized preconditioning methods

Here we discuss Nyström PCG in the context of prior work on randomized preconditioning [10, 69, 95, 113, 143] based on sketch-and-precondition and related ideas. All these prior methods were developed for least squares problems. We summarize the complexity of each method for regularized least-squares problems in Table 3.1.

The time to construct the sketch-and-precondition preconditioner is always larger than that of the Nyström preconditioner, since $d_{\text{eff}}^{\mu} < d$ and $\gamma < 1$. Indeed, constructing the preconditioner for

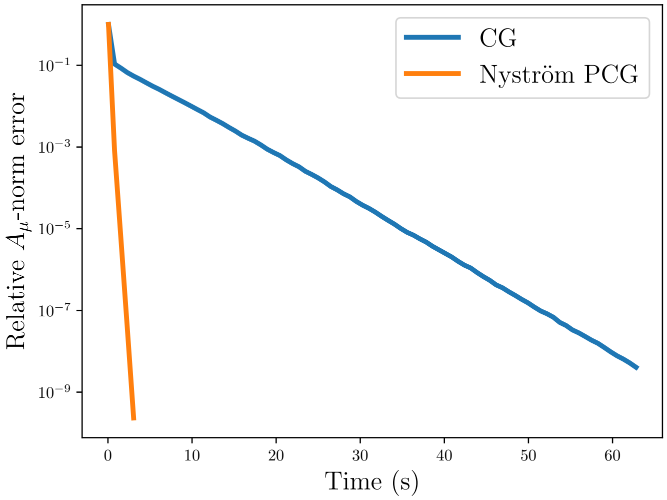


Figure 2.2: **Comparable matvecs does not mean comparable runtime.** On the ridge regression with the covtype random features problem, Nyström PCG uses 1080 matvecs in total to solve the problem, while CG uses 1760 matvecs. Despite the number of matvecs performed by both algorithms being on the same order of magnitude, Nyström PCG is about 20× faster than CG, as the 1000 matvecs used to construct the preconditioner are parallelized.

sketch-and-precondition costs $\Omega(d^3)$, which is the same as a direct method when $d = \Omega(n)$ and is prohibitive for high-dimensional problems. In contrast, Nyström PCG is amenable to problems with large d and runs much faster than sketch-and-precondition whenever $d_{\text{eff}}^{\mu} \ll d$. We note the analysis of sketch-and-precondition can likely be improved to require only a sketch size of $\mathcal{O}(d_{\text{eff}}^{\mu}/\gamma)$. However, this improvement by itself is only of theoretical value as d_{eff}^{μ} is almost never known beforehand. Thus, without an adaptive scheme or method to estimate d_{eff}^{μ}, the best a priori sketch size one can select with sketch-and-precondition methods is $\mathcal{O}(d/\gamma)$. The Nyström preconditioner also enjoys wider applicability then sketch-precondition: it applies to square-ish systems, whereas the others only work for strongly overdetermined or underdetermined problems. Nyström PCG also improves slightly on the complexity of AdaIHS: while both scale linearly in d, Nyström PCG removes unnecessary logarithmic factors and the constant $\rho < 0.18$.

Of the methods presented in Table 3.1, sketched preconditioned SVRG [69] is closest to our approach. The authors of [69] construct a preconditioner from a randomized low-rank approximation to be deployed with the SVRG algorithm [85]. However, while both use a randomized low-rank approximation to construct the preconditioner, the methods differ significantly. In particular, [69] constructs the randomized preconditioner using the randomized block Krylov scheme in [118], which is significantly more expensive than Algorithm 13 used for Nyström PCG. Indeed, the randomized block Krylov scheme requires s-matvecs with A $\mathcal{O}(\log(n))$ times and $\mathcal{O}(\log(n))$ costly orthogonalizations,

Table 2.1: **Regularized least-squares: Complexity of prior randomized preconditoning methods vs. Nyström PCG**. The table compares the complexity of Nyström PCG and state-of-the-art randomized preconditioning methods in the overdetermined case $n \geq d$, assuming we can access A only via matrix-vector products. The sketch-and-precondition preconditioner is constructed from a sketch SA, where $S \in \mathbb{R}^{s\times n}$ is a $(1 \pm \gamma)$ Gaussian subspace embedding with sketch size $s = \Omega(d/\gamma)$ and $\gamma \in (0, 1)$. The time to compute the sketch is $\mathcal{O}(T_{\mathrm{mv}}d/\gamma)$ and the iteration complexity follows from the argument in [171, Sec 2.6]. For AdaIHS, we use a sketch constructed from a Gaussian subspace embedding with sketch size $\mathcal{O}(d_{\mathrm{eff}}^{\mu}/\rho)$ where $\rho \in (0, 0.18)$. The complexity of AdaIHS follows from [95, Theorem 5]. Similarly, the construction in [69] uses a Gaussian test matrix. Let $\tilde{\kappa}_s = \left(s\lambda_s + \sum_{j>s}\lambda_j\right)/\mu$. Then the overall runtime of sketched-preconditioned SVRG follows from [69, Theorem 1] and the runtime of randomized block Krylov method used to construct the preconditioner [108, 118]. The complexity of Nyström PCG is derived from Theorem 2.5.1 and Corollary 2.5.2.

Method	**Complexity**	**References**
Sketch-and-precondition	$\mathcal{O}\left(T_{\mathrm{mv}}d/\gamma + d^3/\gamma + T_{\mathrm{mv}}\frac{\log(1/\epsilon)}{\log(1/\gamma)}\right)$	[10, 113, 143]
AdaIHS	$\mathcal{O}\left((T_{\mathrm{mv}}d_{\mathrm{eff}}^{\mu}/\rho + d(d_{\mathrm{eff}}^{\mu})^2/\rho^2)\log(d_{\mathrm{eff}}^{\mu}/\rho) + T_{\mathrm{mv}}\frac{\log(1/\epsilon)}{\log(1/\rho)}\right)$	[95]
Sketched preconditioned SVRG	$\mathcal{O}\left(T_{\mathrm{mv}}s\log(n) + ds^2\log^2(n)\right)$ $+O\left(T_{\mathrm{mv}} + \tilde{\kappa}_s + d^2\right)\log(1/\epsilon)\big)$	[69]
Nyström PCG	$\mathcal{O}\left(T_{\mathrm{mv}}d_{\mathrm{eff}}^{\mu} + d(d_{\mathrm{eff}}^{\mu})^2 + T_{\mathrm{mv}}\log(1/\epsilon)\right)$	This work

which are needed for numerical stability [108]. Hence sketched preconditioned SVRG is considerably slower than Nyström PCG, see Table 3.1. Moreover, the theory in [69] also lacks any connection with the effective dimension and provides no theoretical or practical guidance for selecting the rank s. Last, note SVRG (unlike PCG) is typically used in settings where a full pass through the data, i.e. a matvec, is too expensive. A preconditioner that requires multiple full passes through the data is an odd choice in this setting.

In the context of kernel ridge regression (KRR), the random features method of [9] may be viewed as a randomized preconditioning technique. [9] prove convergence guarantees for the polynomial kernel with a (large) sketch size $s = \mathcal{O}\left((d_{\mathrm{eff}}^{\mu})^2\right)$. In contrast, Nyström PCG can be used for KRR with any kernel and requires only the smaller sketch size $s = \mathcal{O}(d_{\mathrm{eff}}^{\mu})$ to obtain fast convergence.

Finally, in a pure statistical learning setting, where the primary concern is test-set error and not residual tolerance, fast approximate methods for KRR are also available. The current state of the art is the Falkon algorithm from [146], which shares important commonalities with Nyström-sketch-and-solve. Instead of working with full kernel, it works with $K_{ns} \in \mathbb{R}^{n\times s}$, where K_{ns} is computed with respect to s-centers randomly sampled from the training set. Let D_{eff}^{μ} denote the effective dimension of the kernel covariance operator. Then under appropriate conditions and with $s = \mathcal{O}\left(D_{\mathrm{eff}}^{\mu}\right)$, [146] shows Falkon obtains generalization error comparable to that of exact methods, with runtime $\mathcal{O}(nD_{\mathrm{eff}}^{\mu}\log(n) + (D_{\mathrm{eff}}^{\mu})^3)$. Moreover, it can be shown under the same hypotheses that $D_{\mathrm{eff}}^{\mu} = \mathcal{O}\left(d_{\mathrm{eff}}^{\mu}\right)$. Thus, in principle, Falkon should run much faster than Nyström PCG or random features PCG from [9], and yield nearly identical statistical performance on the test set. We have found Falkon does run faster, but there are gaps in performance relative to Nyström PCG that cannot be improved by increasing the number of centers. That is, to obtain the best statistical performance, there is still a benefit to solving the problem to modest accuracy. See Section 2.6.5 for numerical comparison and further discussion. Furthermore, Falkon only applies to vanilla KRR and kernelized logistic regression [109], and not to Gaussian processes, an application where Nyström PCG might prove useful. Moreover, the Gaussian processes literature [61, 168] has found exact inference yields better learning performance than approximate methods.

In summary, Nyström PCG applies to a wider class of problems than prior randomized preconditioners and enjoys stronger theoretical guarantees for regularized problems. Nyström PCG also outperforms other randomized preconditioners numerically (Section 2.6).

Block Nyström PCG

The Nyström preconditioner can also precondition the block CG algorithm [125] to solve regularized linear systems with multiple right-hand sides, as appear in applications to approximate cross validation [157], influence functions [91], and hyperparameter optimization [106]. Blocking provides advantages both in convergence rate and in memory management. The orthogonalization preprocessing proposed in [53] ensures numerical stability for Block Nyström PCG without further orthogonalization steps during the iteration.

2.5.3 Analysis of Nyström PCG

We now turn to the analysis of the randomized Nyström preconditioner P. Theorem 2.5.1 provides a bound for the rank s of the Nyström preconditioner that reduces the condition number of A_μ to a constant. In this case, we deduce that Nyström PCG converges rapidly (Corollary 2.5.2).

Theorem 2.5.1 (Nyström preconditioning)**.** *Suppose we construct the Nyström preconditioner P in Equation* (2.17) *using Algorithm 13 with sketch size $s = 2\lceil 1.5\, d_{\text{eff}}^{\mu}(A)\rceil + 1$. Using P to precondition the regularized matrix A_μ results in the condition number bound*

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_\mu P^{-1/2})\big] < 28.$$

The proof of Theorem 2.5.1 may be found in Section 2.5.3.

Theorem 2.5.1 has several appealing features. Many other authors have noticed that the effective dimension controls sample size requirements for particular applications such as discriminant analysis [33], ridge regression [95], and kernel ridge regression [3, 13]. In contrast, our result holds for any regularized linear system.

Our argument makes the role of the effective dimension conceptually simpler, and it leads to explicit, practical parameter recommendations. Indeed, the effective dimension $d_{\text{eff}}^{\mu}(A))$ is essentially the same as the sketch size s that makes the approximation error $\|A - \hat{A}_{\text{nys}}\|$ proportional to μ. In previous arguments, such as those in [3, 13, 33], the effective dimension arises because the authors reduce the analysis to approximate matrix multiplication [36], which produces inscrutable constant factors.

We also note Theorem 2.5.1 easily extends to the column sampling schemes using Proposition 2.5.3 and results from [3] to control $\|E\|$. This is particularly attractive for kernel problems, as the Nyström preconditioner maybe constructed in $\mathcal{O}(ns^2)$ operations. For the case of uniform column sampling, the key quantity is the maximal marginal degrees of freedom

$$d_{\text{mof}}^{\mu}(A) = n\|\text{diag}(A(A + n\mu I)^{-1})\|_\infty.$$

Clearly, $d_{\text{mof}}^{\mu}(A) \geq d_{\text{eff}}^{\mu}(A)$, and is generally significantly larger. Combining our results with those from [3], we can conclude a similar result to Theorem 2.5.1 using a rank of size $s = \mathcal{O}\left(d_{\text{mof}}^{\mu}(A)\log(n)\right)$. Thus the guarantees for uniform column sampling are considerably worse than those of random projection. In practice we have found the bound on s for uniform column sampling to be very pessimistic, see Section 2.6.5 for corroborating numerical evidence.

Theorem 2.5.1 ensures that Nyström PCG converges quickly.

Corollary 2.5.2 (Nyström PCG: Convergence)**.** *Define P as in Theorem 2.5.1, and condition on the event $\left\{\kappa_2\left(P^{-1/2}A_\mu P^{-1/2}\right) \leq 56\right\}$. If we initialize Algorithm 3 with initial iterate $w_0 = 0$, then the relative error δ_t in the iterate x_t satisfies*

$$\delta_t = \frac{\|w_t - w_\star\|_{A_\mu}}{\|w_\star\|_{A_\mu}} < 2 \cdot (0.77)^t \quad \textit{where } A_\mu w_\star = b.$$

In particular, after $t = \lceil 3.8\log(2/\epsilon)\rceil$ iterations, we have relative error $\delta_t < \epsilon$.

The proof of Corollary 2.5.2 is an immediate consequence of the standard convergence result for CG [160, Theorem 38.5, p. 299]. See Section 2.8.7. Note Corollary 2.5.2 also immediately implies the total number of matvecs required to reach an ϵ-accurate solution in the A-norm.

Analyzing the condition number

The first step in the proof of Theorem 2.5.1 is a deterministic bound on how the preconditioner (2.17) reduces the condition number of the regularized matrix A_μ. Let us emphasize that this bound is valid for any rank-s Nyström approximation, regardless of the choice of test matrix.

Proposition 2.5.3 (Nyström preconditioner: deterministic bound)**.** *Let $\hat{A} = U\hat{\Lambda}U^T$ be any rank-s Nyström approximation, with sth largest eigenvalue $\hat{\lambda}_s$, and let $E = A - \hat{A}$ be the approximation error. Construct the Nyström preconditioner P as in* (2.17)*. Then the condition number of the* *preconditioned matrix* $P^{-1/2}A_\mu P^{-1/2}$ *satisfies*

$$\begin{aligned}\max\left\{\frac{\hat\lambda_s+\mu}{\lambda_n+\mu},1\right\} &\leq \kappa_2(P^{-1/2}A_\mu P^{-1/2})\\ &\leq \left(\hat\lambda_s+\mu+\|E\|\right)\min\left\{\frac{1}{\mu},\ \frac{\hat\lambda_s+\lambda_n+2\mu}{(\hat\lambda_s+\mu)(\lambda_n+\mu)}\right\}.\end{aligned} \tag{2.18}$$

For the proof of Proposition 2.5.3 see Section 2.8.6. It turns out the only properties of the Nyström approximation we require in Proposition 2.5.3 is that $\hat{A}$ is psd and $E = A - \hat{A} \succeq 0$. Thus, Proposition 2.5.3 also applies to any preconditioner of the form (2.17) constructed from a matrix $\hat{A}$ that possesses these two properties.

To interpret the result, recall the expression Equation (2.16) for the condition number induced by the optimal preconditioner. Proposition 2.5.3 shows that the Nyström preconditioner may reduce the condition number almost as well as the optimal preconditioner. Equation (2.18) shows the price we pay for using an efficiently computable preconditioner, is the condition number of the preconditioned system depends upon our approximation error $\|E\|$. This is natural given the preconditioner is constructed from $\hat{A}$, a perturbed version of A. Hence we expect P to behave like a perturbed version of $P_\star$, which is precisely the content of Proposition 2.5.3.

In particular, when $\|E\| = \mathcal{O}(\mu)$, the condition number of the preconditioned system is bounded by a constant, independent of the spectrum of A. This follows as $\hat\lambda_s \leq \lambda_s$ and $\|E\|$ dominates λ_s. In this setting, Nyström PCG is guaranteed to converge quickly.

The effective dimension and sketch size selection

How should we choose the sketch size s to guarantee that $\|E\| = \mathcal{O}(\mu)$? Corollary 2.2.3 shows how the error in the rank-s randomized Nyström approximation depends on the spectrum of A through the eigenvalues of A and the tail stable rank. In this section, we present a lemma which demonstrates that the effective dimension $d^\mu_{\text{eff}}(A)$ controls both quantities. As a consequence of this bound, we will be able to choose the sketch size s proportional to the effective dimension $d^\mu_{\text{eff}}(A)$.

Recall from Equation (2.4) that the effective dimension of the matrix A is defined as

$$d^\mu_{\text{eff}}(A) = \operatorname{tr}(A(A+\mu I)^{+}) = \sum_{j=1}^{n}\frac{\lambda_j(A)}{\lambda_j(A)+\mu}. \tag{2.19}$$

As previously mentioned, $d^\mu_{\text{eff}}(A)$ may be viewed as a smoothed count of the eigenvalues larger than μ. Thus, one may expect that $\lambda_k(A) \lesssim \mu$ for $k \gtrsim d^\mu_{\text{eff}}(A)$. This intuition is correct, and it forms the content of Lemma 2.5.4.

Lemma 2.5.4 (Effective dimension)**.** *Let* $A \in \mathbb{S}_n^+(\mathbb{R})$ *with eigenvalues* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. *Let* $\mu > 0$ *be regularization parameter, and define the effective dimension as in Equation* (2.19). *The* *following statements hold.*

1. *Fix $\gamma > 0$. If $j \geq (1+\gamma^{-1})d_{\text{eff}}^{\mu}(A)$, then $\lambda_j \leq \gamma\mu$.*

2. *If $k \geq d_{\text{eff}}^{\mu}(A)$, then $k^{-1}\sum_{j>k}\lambda_j \leq (d_{\text{eff}}^{\mu}(A)/k)\cdot\mu$.*

The proof of Lemma 2.5.4 may be found in Section 2.8.6.

Lemma 2.5.4, Item 1 captures the intuitive fact that there are no more than $2d_{\text{eff}}^{\mu}(A)$ eigenvalues larger than μ. Similarly, Item 2 states that the effective dimension controls the sum of all the eigenvalues whose index exceeds the effective dimension. It is instructive to think about the meaning of these results when $d_{\text{eff}}^{\mu}(A)$ is small.

Proof of Theorem 2.5.1

We are now prepared to prove Theorem 2.5.1. The key ingredients in the proof are Proposition 2.2.2, Proposition 2.5.3, and Lemma 2.5.4.

Proof of Theorem 2.5.1. Fix the sketch size $s = 2\lceil 1.5\, d_{\text{eff}}^{\mu}(A)\rceil + 1$. Construct the rank-$s$ randomized Nyström approximation $\hat{A}_{\text{nys}}$ with eigenvalues $\hat{\lambda}_j$. Write $E = A - \hat{A}_{\text{nys}}$ for the approximation error. Form the preconditioner P via Equation (2.17). We must bound the expected condition number of the preconditioned matrix $P^{-1/2}A_{\mu}P^{-1/2}$

First, we apply Proposition 2.5.3 to obtain a deterministic bound that is valid for any rank-s Nyström preconditioner:

$$\kappa_2(P^{-1/2}A_{\mu}P^{-1/2}) \leq \frac{\hat{\lambda}_s + \mu + \|E\|}{\mu} \leq 2 + \frac{\|E\|}{\mu}.$$

The second inequality holds because $\hat{\lambda}_s \leq \lambda_s \leq \mu$. This is a consequence of Lemma 2.2.1, Item 4 and Lemma 2.5.4, Item 1 with $\gamma = 1$. We rely on the fact that $s \geq 2\, d_{\text{eff}}^{\mu}(A)$.

Decompose $s = 2p - 1$ where $p = \lceil 1.5\, d_{\text{eff}}^{\mu}(A)\rceil + 1$. Take the expectation, and invoke Corollary 2.2.3 to obtain

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_{\mu}P^{-1/2})\big] \leq 2 + \left(3 + \frac{4\mathrm{e}^2}{p}\mathrm{sr}_p(A)\right)(\lambda_p/\mu).$$

By definition, $\mathrm{sr}_p(A)\cdot\lambda_p = \sum_{j\geq p}\lambda_j$. To complete the bound, apply Lemma 2.5.4 twice. We use Item 1 with $\gamma = 2$ and Item 2 with $k = p - 1 = \lceil 1.5\, d_{\text{eff}}^{\mu}(A)\rceil$ to reach

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_{\mu}P^{-1/2})\big] \leq 2 + \frac{3\cdot 2\mu + 4\mathrm{e}^2\cdot 2\mu/3}{\mu} < 2 + 26 = 28,$$

which is the desired result. □

2.5.4 Practical parameter selection

In practice, we may not know the regularization parameter μ in advance, and we rarely know the effective dimension $d_{\text{eff}}^{\mu}(A)$. As a consequence, we cannot enact the theoretical recommendation for the rank of the Nyström preconditioner: $s = 2\,\lceil 1.5\, d_{\text{eff}}^{\mu}(A)\rceil + 1$. Instead, we need an adaptive method for choosing the rank s. Below, we outline three strategies.

Strategy 1: Adaptive rank selection by a posteriori error estimation

The first strategy uses the posterior condition number estimate adaptively in a procedure the repeatedly doubles the sketch size s as required. Recall that Proposition 2.5.3 controls the condition number of the preconditioned system:

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq \frac{\hat{\lambda}_s + \mu + \|E\|}{\mu} \quad \text{where } E = A - \hat{A}_{\text{nys}}. \tag{2.20}$$

We get $\hat{\lambda}_s$ for free from Algorithm 13 and we can compute the error $\|E\|$ inexpensively with the randomized power method [94]; see Algorithm 4 in Section 2.10.1. Thus, we can ensure the condition number is small by making $\|E\|$ and $\hat{\lambda}_s$ fall below some desired tolerance. The adaptive strategy proceeds to do this as follows. We compute a randomized Nyström approximation with initial sketch size s_0, and we estimate the error $\|E\|$ using randomized powering. If $\|E\|$ is smaller than a prescribed tolerance, we accept the rank-s_0 approximation. If the tolerance is not met, then we double the sketch size, update the approximation, and estimate $\|E\|$ again. The process repeats until the estimate for $\|E\|$ falls below the tolerance or it breaches a threshold $s_{\max}$ for the maximum sketch size. Algorithm 5 uses the following stopping criterions $\|E\| \leq \text{Tol}_{\text{Err}}$ and $\hat{\lambda}_s \leq \text{Tol}_{\text{Rat}}$ for tolerances Tol_{Err} and Tol_{Rat}. The stopping criterion on $\hat{\lambda}_s$ does not seem to be necessary in practice, as it is usually an order of magnitude small than $\|E\|$, but it is needed for Theorem 2.5.5. Based on numerical experience, we recommend the choices $\text{Tol}_{\text{Err}} = \tau\mu, \text{Tol}_{\text{Rat}} = \tau\mu/10$ for $\tau \in [1, 100]$. For full algorithmic details of adaptive rank selection by estimating $\|E\|$, see Algorithm 5 in Section 2.10.

The following theorem shows that with high probability, Algorithm 5 terminates with a modest sketch size in at most a logarithmic number of steps, and PCG with the resulting preconditioner converges rapidly.

Theorem 2.5.5. *Run Algorithm 5 with initial sketch size s_0, tolerances $\text{Tol}_{\text{Err}} = \tau\mu, \text{Tol}_{\text{Rat}} = \tau\mu/11$ where $\tau \geq 1$, and let $\tilde{s} = 2\lceil 2 d_{\text{eff}}^{\delta\tau\mu/11}(A)\rceil + 1$. Then with probability at least $1 - \delta$:*

1. *Algorithm 5 doubles the sketch size at most $\lceil \log_2\left(\frac{\tilde{s}}{s_0}\right)\rceil$ times.*

2. *The final sketch size s satisfies*

$$s \leq 4\lceil 2 d_{\text{eff}}^{\delta\tau\mu/11}(A)\rceil + 2.$$

3. *With the preconditioner constructed from Algorithm 5, Nyström PCG converges in at most* $\lceil \frac{\log(2/\epsilon)}{\log(1/\tau_0)} \rceil$ *iterations, where* $\tau_0 = \frac{\sqrt{1+12\tau/11}-1}{\sqrt{1+12\tau/11}+1}$.

Theorem 2.5.5 immediately implies the following concrete guarantee.

Corollary 2.5.6. *Set* $\tau = 44$ *and* $\delta = 1/4$ *in Algorithm 5 then with probability at least* $3/4$*:*

1. *Algorithm 5 doubles the sketch size at most* $\lceil \log_2 \left(\frac{\tilde{s}}{s_0}\right) \rceil$ *times.*
2. *The final sketch size* s *satisfies*
$$s \leq 4\lceil 2d_{\text{eff}}^{\mu}(A) \rceil + 2.$$
3. *With the preconditioner constructed from Algorithm 5, Nyström PCG converges in at most* $\lceil 3.48 \log(2/\epsilon) \rceil$ *iterations.*

Strategy 2: Adaptive rank selection by monitoring $\hat{\lambda}_s/\mu$

The second strategy is almost identical to the first, except we monitor the ratio $\hat{\lambda}_s/\mu$ instead of $\|E\|/\mu$. Strategy 2 doubles the approximation rank until $\hat{\lambda}_s/\mu$ falls below some tolerance (say, 10) or the sample size reaches the threshold $s_{\max}$. The approach is justified by the following proposition which shows that once the rank s is sufficiently large, with high probability, the exact condition number differs from the empirical condition number $(\hat{\lambda}_s + \mu)/\mu$ by at most a constant.

Proposition 2.5.7. *Let* $\tau \geq 0$ *denote the tolerance and* $\delta > 0$ *a given failure probability. Suppose the rank of the randomized Nyström approximation satisfies* $s \geq 2\lceil 2d_{\text{eff}}^{\tau\mu}(A)) \rceil + 1$. *Then*

$$\mathbb{P}\left\{\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_s + \mu}{\mu}\right)_+ \leq \frac{\tau}{\delta}\right\} \geq 1 - \delta, \tag{2.21}$$

where $(\cdot)_+ = \max\{\cdot, 0\}$.

This strategy has the benefit of saving a bit of computation and is preferable when a moderately small residual is sufficient, eg, in machine learning problems where training error only loosely predicts test error.

Strategy 3: Choose s as large as the user can afford

The third strategy is to choose the rank s as large as the user can afford. This approach is coarse, and it does not yield any guarantees on the cost of the Nyström PCG method.

Nevertheless, once we have constructed a rank-s Nyström approximation we can combine the posterior estimate of the condition number used in strategy 1 with the standard convergence theory of PCG to obtain an upper bound for the iteration count of Nyström PCG. This gives us advance warning about how long it may take to solve the regularized linear system. As in strategy 1 we compute the error $\|E\|$ in the condition number bound inexpensively with the randomized power method.

2.6 Applications and experiments

In this section, we study the performance of Nyström PCG on real world data from three different applications: ridge regression, kernel ridge regression, and approximate cross-validation. The experiments demonstrate the effectiveness of the preconditioner and our strategies for choosing the rank s compared to other algorithms in the literature: on large datasets, we find that our method outperforms competitors by a factor of 5–10 (Table 2.3 and Table 2.8).

2.6.1 Preliminaries

We implemented all experiments in MATLAB R2019a and MATLAB R2021a on a server with 128 Intel Xeon E7-4850 v4 2.10GHz CPU cores and 1056 GB. Except for the very large scale datasets ($n \geq 10^5$), every numerical experiment in this section was repeated twenty times; tables report the mean over the twenty runs, and the standard deviation (in parentheses) when it is non-zero. We highlight the best-performing method in a table in bold.

We select hyperparameters of competing methods by grid search to optimize performance. This procedure tends to be very charitable to the competitors, and it may not be representative of their real-world performance. Indeed, grid search is computationally expensive, and it cannot be used as part of a practical implementation. A detailed overview of the experimental setup for each application may be found in the appropriate section of Section 2.11, and additional numerical results in Section 2.12.

2.6.2 Ridge regression

In this section, we solve the ridge regression problem (2.7) described in Section 2.1.3 on some standard machine learning data sets (Table 2.2) from OpenML [167] and LIBSVM [29]. The effective dimension d_{eff}^{μ} and the numerical rank of these matrices provide insight into the difficulty of each problem. These are reported in Table 2.2. We compare Nyström PCG to standard CG and two randomized preconditioning methods, the sketch-and-precondition method of Rokhlin and Tygert (R&T) [143] and the Adaptive Iterative Hessian Sketch (AdaIHS) [95].

Experimental overview

We perform two sets of experiments: computing regularization paths on CIFAR-10 and Guillermo, and random features regression [135,137] on shuttle, smallNORB, Higgs, YearMSD, and covtype with specified values of μ. The values of μ may be found in Section 2.11.1. We use the Euclidean norm

Table 2.2: **Ridge regression: Dataset statistics.** The table reports the effective dimension and numerical rank (in double precision) of each dataset. For CIFAR-10 and Guillermo, we report d_{eff}^{μ} using the value of μ on the regularization path that yields the best test error. The numerical rank (NumRank) of a matrix $A \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_k$ is $\max\{k : \lambda_k \geq \lambda_1 \epsilon\}$, the number of eigenvalues larger than machine precision ϵ scaled by the spectral norm of A.

Dataset	**n**	**d**	d_{eff}^{μ}	**NumRank**
CIFAR-10	40,000	3,072	1,258	3,072
Guillermo	16,000	4,297	1,885	2,000
smallNorb-rf	24,300	10,000	1,806	6,812
shuttle-rf	43,300	10,000	439	853
Higgs-rf	800,000	10,000	936	10,000
YearMSD-rf	463,715	15,000	7,262	15,000
covtype-binary	464,810	15,000	14,629	15,000

$\|r\|_2$ of the residual as our stopping criteria and declare convergence when $\|r\|_2 \leq 10^{-10}$. For both sets of experiments, we use Nyström PCG with adaptive rank selection (Algorithm 5 in Section 2.10). For experimental details, see Section 2.11.1.

The regularization path experiments solve Equation (2.7) over a regularization path $\mu = 10^{j}$ where $j = 3, \cdots, -6$. We first solve the problem for the largest μ and then solve for progressively smaller μ by warm starting from the previous solution. We allow every method at most 500 iterations to reach the desired tolerance, for each value of μ.

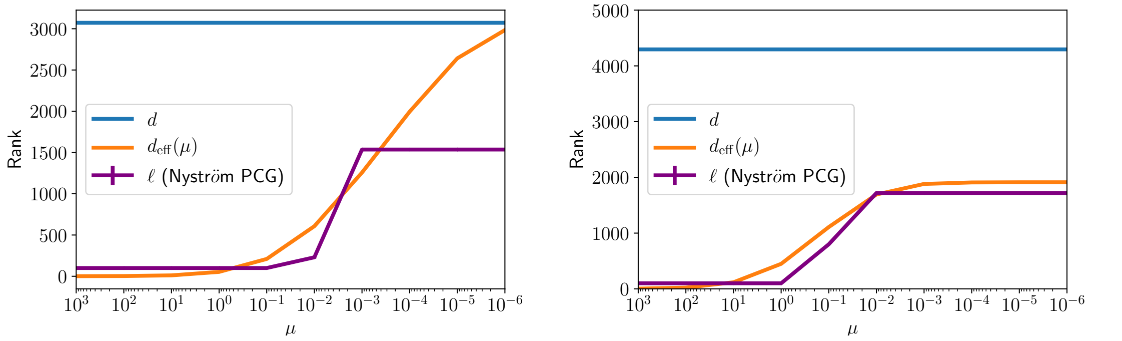
We report the test error achieved on each dataset in Section 2.9.1. We also compare to the test-error obtained by a sketch-and-solve approach that approximates the inverse using the Nyström preconditioner, and which is known to admit good learning guarantees under appropriate conditions [3, 13].

Computing the regularization path

Figure 2.3 shows how the effective dimension d_{eff}^{μ} varies with the regularization parameter μ on two small datasets. We see that the effective dimension reaches our chosen maximum sketch size, $s_{\max} = 0.5d$ for CIFAR-10 and $s_{\max} = 0.4d$ for Guillermo, when μ is small enough. For CIFAR-10, Nyström PCG chooses a rank much smaller than the effective dimension for small values of μ, yet the method still performs well (Figure 2.4).

Figure 2.4 show the effectiveness of each method for computing the entire regularization path. Nyström PCG is the fastest almost uniformly. The one exception is on CIFAR-10, where R&T performs better for the smallest regularization parameter, for which $d_{\text{eff}}^{\mu} \approx d$. That is, the $\mathcal{O}(d^3)$ cost of forming the R&T preconditioner is not worthwhile unless $d_{\text{eff}}^{\mu} \approx d$ and the regularization is negligible.

AdaIHS is rather slow. It increases the sketch size parameter several times along the regularization path. Each time, AdaIHS must form a new sketch of the matrix, approximate the Hessian, and



(a) CIFAR 10

(b) Guillermo

Figure 2.3: **Ridge regression: Adaptive sketch size selection.** Nyström PCG with adaptive rank selection (Algorithm 5) selects a preconditioner whose rank is less than or equal to the effective dimension. Error bars for the rank selected by the adaptive algorithm are so small that they are not visible in the graph: the behavior of the adaptive algorithm is essentially deterministic across runs. See Section 2.6.2.

compute a Cholesky factorization.

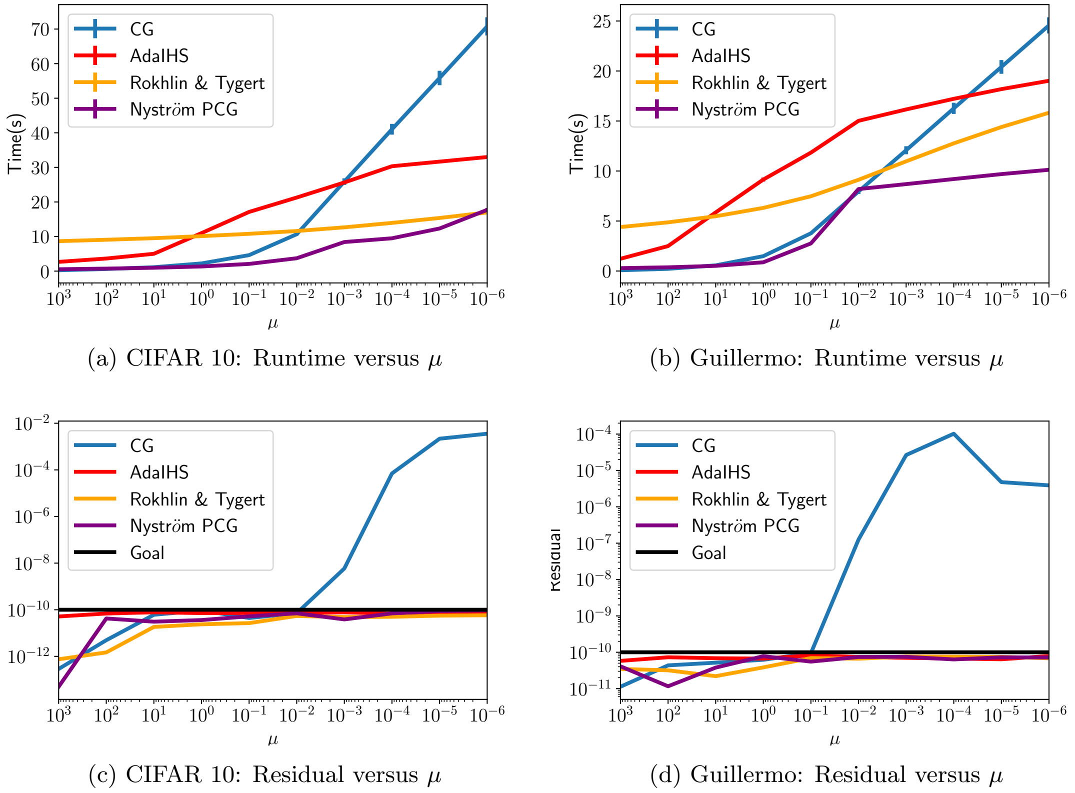
Random features regression

Table 2.3 compares the performance of Nyström PCG, AdaIHS, and R&T PCG for random features regression. Table 2.3 shows that Nyström PCG performs best on all datasets for all metrics. The most striking feature is the difference between sketch sizes: AdaIHS and R&T require much larger sketch sizes than Nyström PCG, leading to greater computation time and higher storage costs. Table 2.3, in conjunction with Table 2.2, shows the adaptive scheme in Section 2.5.4 effectively selects a rank on the order of d_{eff}^{μ} when the effective dimension is small or moderate.

Nyström PCG also works well for sketch sizes smaller than the effective dimension. For example, on YearMSD-rf, Nyström PCG converges quickly despite a rank three times smaller than d_{eff}^{μ}. For covtype-rf, where $d_{\text{eff}}^{\mu} \sim d$, the convergence is no longer as fast, but it still outperforms R&T, owing to the expensive $\mathcal{O}(d^3)$ cost of constructing the preconditioner. Thus, even in settings where $d_{\text{eff}}^{\mu} \sim d$, Nyström PCG may still be faster than R&T when d is large enough. For a discussion on the statistical performance of Nyström PCG and the test set error obtained on all datasets, see Section 2.9.1.

2.6.3 Approximate cross-validation

In this subsection we use our preconditioner to compute approximate leave-one-out cross-validation (ALOOCV), which requires solving a large linear system with multiple right-hand sides.



(a) CIFAR 10: Runtime versus μ

(b) Guillermo: Runtime versus μ

(c) CIFAR 10: Residual versus μ

(d) Guillermo: Residual versus μ

Figure 2.4: **Ridge regression: Runtime and residual.** Nyström PCG is either the fastest method, or it is competitive with the fastest method, for all values of the regularization parameter μ. CG is generally the slowest method. All the methods reliably achieve the target residual along the entire regularization path, except for ordinary CG at small values of μ. See Section 2.6.2.

Background

Cross-validation is an important machine-learning technique to assess and select models and hyperparameters. Generally, it requires re-fitting a model on many subsets of the data, so can take quite a long time. The worst culprit is leave-one-out cross-validation (LOOCV), which requires running an expensive training algorithm n times. Recent work has developed approximate leave-one-out cross-validation (ALOOCV), a faster alternative that replaces model retraining by a linear system solve [64, 134, 170]. In particular, these techniques yield accurate and computationally tractable approximations to LOOCV.

To present the approach, we consider the infinitesimal jacknife (IJ) approximation to LOOCV [64, 156]. The IJ approximation computes

$$\tilde{\theta}_{\mathrm{IJ}}^{n/j} = \hat{\theta} + \frac{1}{n} H^{-1}(\hat{\theta}) \nabla_{\theta} l(\hat{\theta}, x_j), \tag{2.22}$$

Table 2.3: **Ridge regression: Nyström PCG versus AdaIHS and R&T PCG.** Nyström PCG outperforms AdaIHS and R&T PCG in iteration (#iters) and runtime for all datasets. Nyström PCG also requires much less storage (s_{final}). For Nyström PCG, the estimated condition number of the preconditioned system κ_{PCG} is computed using the upper bound in Proposition 2.5.3.

Dataset	**Method**	$\mathbf{s_{final}}$	$\kappa_{\mathbf{PCG}}$	**#iters**	**Runtime (s)**
shuttle-rf	AdaIHS	10,000	-	66.9 (0.933)	66.9 (5.27)
	R&T PCG	20,000	-	60.15	242.6 (12.24)
	NysPCG	800	4.17 (0.161)	**13.1 (1.47)**	**9.78 (0.943)**
smallNORB-rf	AdaIHS	12,800	-	38.7 (1.42)	41.0 (2.46)
	R&T PCG	20,000	-	34.5 (1.31)	181.5 (6.53)
	NysPCG	800	18.5 (0.753)	**31.5 (0.489)**	**6.67 (0.372)**
YearMSD-rf	AdaIHS	30,000	-	44	1,327.3
	R&T PCG	30,000	-	49	766.5
	NysPCG	2,000	22.7	**22**	**209.7**
Higgs-rf	AdaIHS	6,400	-	55	1,052.7
	R&T PCG	20,000	-	53	607.4
	NysPCG	800	23.8	**28**	**91.26**
covtype-rf	AdaIHS	30,000	-	211	1,633.5
	R&T PCG	30,000	-	**50**	846.4
	NysPCG	2000	2.12e+4	430	**540.05**

Table 2.4: **ALOOCV: Datasets and experimental parameters.** For each dataset we consider two values of μ, we aso report the exact effective dimension

Dataset	**n**	**d**	**%nz(A)**	$\boldsymbol{\mu}$	$s_{\mathbf{init}}$	$\mathbf{d}^{\mu}_{\text{eff}}$
Gisette	6,000	5,000	99.1%	1 1e-4	850	116 948
real-sim	72,308	20,958	0.245%	1e-4 1e-8	500	891 6,686
rcv1.binary	20,242	47,236	0.157%	1e-4 1e-8	500	779 3,463
SVHN	73,257	3,072	100%	1 1e-4	850	10 674

where $H(\hat{\theta}) \in \mathbb{R}^{d\times d}$ is the Hessian of the loss function at the solution $\hat{\theta}$, for each datapoint x_j. The main computational challenge is computing the inverse Hessian vector product $H^{-1}(\hat{\theta})\nabla_{\theta} l(\hat{\theta}, x_j)$. When n is very large, we can also subsample the data and average Equation (2.22) over the subsample to estimate ALOOCV. Since ALOOCV solves the same problem with several right-hand sides, blocked PCG methods (here, Nyström blocked PCG) are the tool of choice to efficiently solve for multiple right-hand sides at once. To demonstrate the idea, we perform numerical experiments on ALOOCV for l_2-regularized logistic regression. The datasets we use are all from LIBSVM [29]; see Table 2.4.

Experimental overview

We perform two sets of experiments in this section. The first set of experiments uses Gisette and SVHN to test the efficacy of Nyström sketch-and-solve. These datasets are small enough that we can factor $H(\theta)$ using a direct method. We also compare to block CG and block PCG with the computed

Table 2.5: **ALOOCV: Small datasets.** The error for a given value of μ is the maximum relative error on 100 randomly sampled datapoints, averaged over 20 trials.

Dataset	μ	**Nyström sketch-and-solve**	**Block CG**	**Block Nyström PCG**
Gisette	1	4.99e–2	2.68e–11	2.58e–12
Gisette	1e–4	1.22e–0	1.19e–11	6.59e–12
SVHN	1	9.12e–5	2.80e–13	1.26e–13
SVHN	1e–4	3.42e–1	2.01e–10	1.41e–11

Nyström approximation as a preconditioner. To assess the error due to an inexact solve for datapoint x_j, let $w_\star(x_j) = H^{-1}(\theta)\nabla_\theta l(\hat{\theta}, x_j)$. For any putative solution $\hat{w}(x_j)$, we compute the relative error $\|\hat{w}(x_j) - w_\star(x_j)\|_2/\|w_\star(x_j)\|_2$. We average the relative error over 100 data-points x_j.

The second set of experiments uses the larger datasets real-sim and rcv1.binary and small values of μ, the most challenging setting for ALOOCV. We restrict our comparison to block Nyström PCG versus the block CG algorithm, as Nyström sketch-and-solve is so inaccurate in this regime. We employ Algorithm 5 to construct the preconditioner for block Nyström PCG.

Nyström sketch-and-solve

As predicted, Nyström sketch-and-solve works poorly (Table 2.5). When $\mu = 1$, the approximate solutions are modestly accurate, and the accuracy degrades as μ decreases to 10^{-4}. The experimental results agree with the theoretical analysis presented in Section 2.4, which indicate that sketch-and-solve degrades as μ decreases. In contrast, block CG and block Nyström PCG both provide high-quality solutions for each datapoint for both values of the regularization parameter.

2.6.4 Large-scale ALOOCV experiments

Table 2.6 summarizes results for block Nyström PCG and block CG on the larger datasets. When $\mu = 10^{-4}$, block Nyström PCG offers little or no benefit over block CG because the data matrices are very sparse (see Table 2.4) and the rcv1 problem is well-conditioned (see Table 2.11).

For $\mu = 10^{-8}$, block Nyström PCG reduces the number of iterations substantially, but the speedup is negligible. The data matrix A is sparse, which reduces the benefit of the Nyström method. Block CG also benefits from the presence of multiple right-hand sides just as block Nyström PCG. Indeed, O'Leary proved that the convergence of block CG depends on the ratio $(\lambda_s + \mu)/(\lambda_n + \mu)$, where s is is the number of right-hand sides [125]. Consequently, multiple right-hand sides precondition block CG and accelerate convergence. We expect bigger gains over block CG when A is dense.

Table 2.6: **ALOOCV: Large datasets.** Block Nyström PCG outperforms block CG for small μ.

Dataset	μ	Method	#iters	Runtime (s)
rcv1	1e–4	Block CG	12	**11.06 (0.874)**
	1e–4	Block Nyström PCG	**10**	11.87 (0.767)
rcv1	1e–8	Block CG	52	39.03 (2.97)
	1e–8	Block Nyström PCG	**15**	**24.1 (1.79)**
realsim	1e–4	Block CG	12	23.04 (2.04)
	1e–4	Block Nyström PCG	**8**	**19.05 (1.10)**
realsim	1e–8	Block CG	90	163.7 (12.3)
	1e–8	Block Nyström PCG	**32**	**68.9 (5.30)**

2.6.5 Kernel ridge regression

Our last application is kernel ridge regression (KRR), a supervised learning technique that uses a kernel to model nonlinearity in the data. KRR leads to large dense linear systems that are challenging to solve.

Background

We briefly review KRR [151]. Given a dataset of inputs $x_i \in \mathcal{D}$, their corresponding outputs $b_i \in \mathbb{R}$ for $i = 1, \dots, n$, and a kernel function $\mathcal{K}(x, y)$, KRR finds a function $f_\star : \mathcal{D} \to \mathbb{R}$ in the associated reproducing kernel Hilbert space $\mathcal{H}$ that best predicts the outputs for the given inputs. The solution $f_\star \in \mathcal{H}$ minimizes the square error subject to a complexity penalty:

$$f_\star = \operatorname*{argmin}_{f \in \mathcal{H}} \quad \frac{1}{2n} \sum_{i=1}^{n} (f(x_i) - b_i)^2 + \frac{\mu}{2} \|f\|_{\mathcal{H}}^2, \tag{2.23}$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm on $\mathcal{H}$. Define the kernel matrix $K \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = \mathcal{K}(x_i, x_j)$. The representer theorem [153] states the solution to (2.23) is

$$f_\star(x) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(x, x_i),$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$ solves the linear system

$$(K + n\mu I)\alpha = b. \tag{2.24}$$

Solving the linear system (2.24) is the computational bottleneck of KRR. Direct factorization methods to solve (2.24) are prohibitive for large n as their costs grow as n^3; for $n > 10^4$ or so, iterative methods are generally preferred. However, K is often extremely ill-conditioned, even with the regularization term $n\mu I$. As a result, CG for Problem (2.24) converges slowly.

Experimental overview

We use Nyström PCG (NysPCG) to solve several KRR problems derived from classification problems on real world datasets from [29, 167]. For all experiments, we use the Gaussian kernel $\mathcal{K}(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$. Following [9], we take a machine learning perspective: the objective is to minimize test set error rather than to achieve the smallest possible residual. We compare our method to random features PCG (rfPCG), proposed in [9]. We do not compare to vanilla CG as it is much slower than NysPCG and rfPCG. We also compare to Falkon [146], a state-of-the-art scalable approximation method for kernel ridge regression. For Falkon, we use the author's Matlab implementation provided here: `https://github.com/LCSL/FALKON_paper`. This implementation is more optimized than the implementations of NysPCG and rfPCG, making use of C++ for several important steps. Thus, the comparison to rfPCG and NysPCG made here is very favorable to Falkon.

All datasets either come with specified test sets, or we create one from a random 80–20 split. The PCG tolerance, σ, and μ were all chosen to achieve good performance on the test sets (see Table 2.8 below). In particular, the test set error on a given dataset saturates or increases if PCG (either rfPCG or NysPCG) is not stopped after reaching the selected tolerance. Both rfPCG and NysPCG were allowed to run for a maximum of 500 iterations. We report statistics for each dataset and experimental parameters in Table 2.7.

In addition, Table 2.7 also reports estimates of the effective dimension and the numerical rank for each kernel matrix. For these KRR systems, computing the exact effective dimension and numerical rank is too expensive, even in single-precision. Instead, we use procedures described in [111] to estimate the effective dimension and numerical rank (in single precision) of the kernel matrix, and report only a lower bound on the effective dimension or numerical rank if the estimate exceeds $\lceil 0.25n \rceil$.

We run two sets of experiments. For the datasets with $n < 10^5$, the "oracle" method uses the a posteriori best parameters for rfPCG (the rank of random features approximation used to construct the preconditioner) and NysPCG (the sketch size s), chosen by grid search, which we call Or-rfPCG and Or-NysPCG respectively. We also compare to the adaptive NysPCG algorithm (Ada-NysPCG) described in Section 2.5.4. We restrict values for s and the rank of the random features approximation to be less than $10,000$ to ensure the preconditioners are cheap to apply and store. Ada-NysPCG for each dataset was initialized at $s = 2,000$, which is smaller than $0.05n$ for all datasets. For the datasets with $n \geq 10^5$, we restrict both s and the rank of the random features approximation to $1,000$, which corresponds to less than $0.01n$. This fixed-rank setting allows us to see how both methods perform in the situation where the size of the preconditioner is restricted owing to memory constraints. We then run both algorithms until they reach the desired tolerance or the maximum number of iterations. Falkon's main hyperparameter is the number of centers, which is typically taken to be a small fraction of the training set. For our experiments, we selected the number of centers via grid search using the grid $\{\lceil 0.01n \rceil, \lceil 0.025n \rceil, \lceil 0.05n \rceil, \lceil 0.075n \rceil, \lceil 0.1n \rceil\}$. The number of

Table 2.7: **Kernel ridge regression: Datasets and experimental parameters.** The table shows experimental parameters and estimates of the effective dimension and numerical rank (see Table 2.2) of each kernel in single precision. These estimates are computed using sketching methods as described in [111].

Dataset	**n**	**d**	**$\mathbf{n}_{\text{classes}}$**	**$\boldsymbol{\mu}$**	**$\boldsymbol{\sigma}$**	**Tol**	**$d_{\text{eff}}^{n\mu}$**	**NumRank**
ijcnn1	49,990	49	2	1e-6	0.5	1e-3	5,269	> 12,498
MNIST	60,000	784	10	1e-7	5	1e-4	$> 15,000$	> 15,000
Sensorless	48,509	48	11	1e-8	0.8	1e-4	1,948	2,331
SensIT	78,823	100	3	1e-8	3	1e-3	8,186	9,216
MiniBooNE	104,052	50	2	1e-7	5	1e-4	522	1,065
EMNIST	105,280	784	47	1e-6	8	1e-3	17,079	> 26,320
Santander	160,000	200	2	1e-6	7	1e-3	> 40,000	> 40,000

iterations used for solving the Falkon linear system is fixed at 20, matching the setting used by the authors in `https://github.com/LCSL/FALKON_paper` for datasets satisfying $n \lesssim 10^6$.

We use column sampling to construct the Nyström preconditioner for all KRR problems. On these problems, random projection takes longer and yields similar performance (with somewhat lower variance).

Experimental results

Table 2.8 summarizes the results for the KRR experiments. Table 2.8 shows that both versions of Nyström PCG perform better than random features preconditioning on all the datasets considered. Nyström PCG also uses less storage. In the fixed-rank setting with the larger scale datasets, Nyström PCG performs better than random features PCG. The second column in Table 2.8 shows the adaptive strategy proposed in Section 2.5.4 to select the sketch size s works very well. In contrast, it is difficult to choose the rank for random features preconditioning: the authors of [9] provide no guidance except for the polynomial kernel. Moreover, the success of Nyström PCG is robust to reaching the effective dimension. Indeed, on MNIST, EMNIST, and Santander, Table 2.7 shows s is much smaller than d_{eff}^{μ}, yet Nyström PCG still converges quickly using the constructed preconditioner. This robustness is important from the viewpoint of practice, for as Table 2.7 reveals, the effective dimension d_{eff}^{μ} is often large.

Table 2.8 shows that good test set error is obtained on all datasets. Significantly, Nyström PCG yields lower test set error than approximate methods such as Falkon and a sketch-and-solve style method that simply applies the inverse of the Nyström preconditioner to the righthand side (NysPrec). However, Falkon and NysPrec run considerably faster as they work with only a subsample of the kernel matrix. We also see that Falkon generally outperforms NysPrec. The gap between Nyström PCG and Falkon can be quite large, such as with EMNIST where Nyström PCG obtains an error of 15.00% compared to the 17.57% obtained by Falkon. Furthermore, we found this gap persisted even as we varied the number of centers from $0.01n$ to $0.5n$, at which point Falkon becomes more expensive than Nyström PCG, see Figure 2.5. Our observation that exact methods outperform approximate methods is consistent with findings in [9], which noted a similar performance gap between random features PCG and the basic random features method of [135]. Thus, even in the statistical learning setting, solving the problem more accurately using the full data does yield improved performance.

Table 2.8: **Kernel ridge regression: ranks, iteration count, and total runtime.** We denote random features PCG and Nyström PCG by rfPCG and NysPCG respectively. The prefixes Or and Ada stand for oracle and adaptive. NysPrec is a sketch-and-solve style method that applies the inverse of the Nyström preconditioner to the righthand side. The total runtime for for rfPCG and both variants of NysPCG also includes the the time required compute the kernel matrix. All variants of NysPCG uniformly outperforms rfPCG, in both runtime and number of iterations. NysPCG also attains the best test error across all problem instances.

Dataset	**Method**	**Rank or #centers**	**#iters**	**Total Runtime (s)**	**Test error**
icjnn1	Or-rfPCG	3,000	63.8(2.66)	56.2(2.33)	**1.25**%
	Ada-NysPCG	2,000	43.7(1.77)	49.9(1.47)	
	Or-NysPCG	3,000	**31.8**(**0.835**)	51.2(1.60)	
	NysPrec	3,000	-	6.09(0.151)	7.06%
	Falkon	4,999	-	4.60(0.122)	1.39%
MNIST	Or-rfPCG	9,000	314.5(2.88)	291.4(6.93)	**1.22**%
	Ada-NysPCG	6,000 (1,716)	78.5(17.65)	185.8(46.39)	
	Or-NysPCG	4,000	**77.9**(**2.08**)	129.4(2.08)	
	NysPrec	4,000	-	31.36(0.427)	34.57%
	Falkon	6,000	-	7.01(0.288)	1.98%
Sensorless	Or-rfPCG	5,000	55.4(2.35)	56.5(3.96)	**2.05**%
	Ada-NysPCG	2,000	22.0(0.510)	40.0(1.26)	
	Or-NysPCG	2,000	**21.7**(**0.571**)	39.3(1.63)	
	NysPrec	2,000	-	9.21(0.248)	3.91%
	Falkon	3,639	-	3.12(0.214)	2.16%
SensIT	Or-rfPCG	7,000	68.0(4.31)	146.0(6.19)	**12.83**%
	Ada-NysPCG	2,000	**47.8**(**1.72**)	120.9(2.43)	
	Or-NysPCG	2,000	48.7(3.40)	112.4(6.41)	
	NysPrec	2,000	-	14.10(0.321)	22.55%
	Falkon	7,883	-	17.55(0.494)	13.05%
MiniBooNE	rfPCG	1,000	92	240.8	**7.93**%
	NysPCG	1,000	**72**	224.0	
	NysPrec	1,000	-	9.06	8.90%
	Falkon	10,046	-	43.03	7.96%
EMNIST	rfPCG	1,000	154	753.1	**15**%
	NysPCG	1,000	**32**	386.3	
	NysPrec	1,000	-	9.21	26.90%
	Falkon	10,528	-	24.5	17.57%
Santander	rfPCG	1,000	160	1019.7	**8.90**%
	NysPCG	1,000	**31**	374.1	
	NysPrec	1,000	-	13.66	19.24%
	Falkon	16,000	-	43.06	9.26%

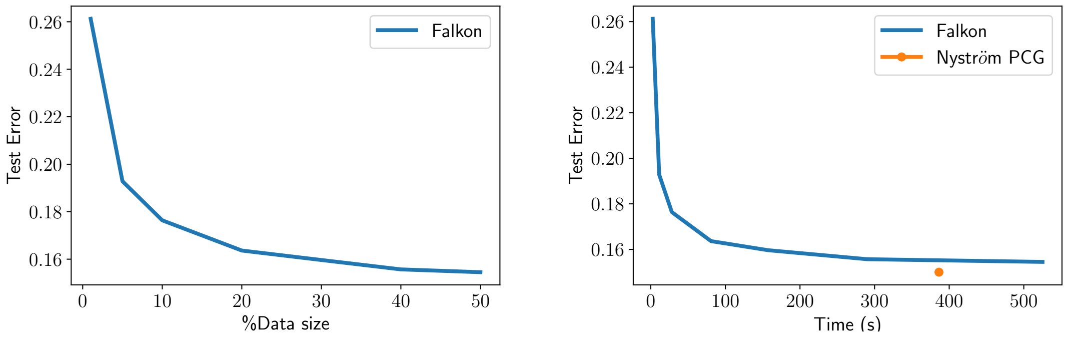


Figure 2.5: **Falkon saturates.** Falkon's performance saturates as the number of centers increases from $0.01n$ to $0.5n$, and always underperforms Nyström PCG. Furthermore, once the number of centers reaches $0.5n$, Falkon runs slower than Nyström PCG.

2.7 Conclusion

We have shown that Nyström PCG delivers a strong benefit over standard CG both in the theory and in practice, thanks to the ease of parameter selection, on a range of interesting large-scale computational problems including ridge regression, kernel ridge regression, and ALOOCV. In our experience, Nyström PCG outperforms all generic methods for solving large-scale dense linear systems with spectral decay. It is our hope that this paper motivates further research on randomized preconditioning for solving large scale linear systems and offers a useful speedup to practitioners.

2.8 Proofs not appearing in the main chapter

2.8.1 Proof of Proposition 2.2.2

Useful facts about Gaussian random matrices

In this subsection we record some useful results about Gaussian random matrices that are necessary for the proof of Proposition 2.2.2. The proof of Proposition 2.2.2 follows in Section 2.8.2.

Proposition 2.8.1 ([76, 119])**.** *Let G be $(s-p)\times s$ standard Gaussian matrix with $s \geq 4$ and $2 \leq p \leq s-2$. Then*

$$\left(\mathbb{E}\|G^{\dagger}\|_F^2\right)^{1/2} = \sqrt{\frac{s-p}{p-1}}, \tag{2.25}$$

and

$$\left(\mathbb{E}\|G^{\dagger}\|^2\right)^{1/2} \leq \mathrm{e}\sqrt{\frac{s}{p^2-1}}. \tag{2.26}$$

Remark 2.8.2. The first display in Proposition 2.8.1 appears in [76], while the second display is due to [119].

We also require one new result, which strengthens the improved version of Chevet's theorem due to Gordon [70]

Proposition 2.8.3 (Squared Chevet). *Fix matrices $S \in \mathbb{R}^{r\times m}$ and $T \in \mathbb{R}^{n\times s}$ and let $G \in \mathbb{R}^{m\times n}$ be a standard Gaussian matrix. Then*

$$\mathbb{E}\|SGT\|^2 \leq \left(\|S\|\|T\|_F + \|S\|_F\|T\|\right)^2.$$

We defer the proof of Proposition 2.8.3 to Section 2.8.3.

Remark 2.8.4. Chevet's theorem states that [76]

$$\mathbb{E}\|SGT\| \leq \|S\|\|T\|_F + \|S\|_F\|T\|.$$

Proposition 2.8.3 immediately implies Chevet's theorem by Hölder's Inequality.

2.8.2 Proof of Proposition 2.2.2

Proof. Proposition 11.1 in [108, Sec. 11] and the argument of Theorem 11.4 in [108, Sec. 11] shows that

$$\|A - \hat{A}_{\text{nys}}\| \leq \|\Sigma_{s-p+1}\|^2 + \|\Sigma_{s-p+1}\Omega_2\Omega_1^\dagger\|^2.$$

Taking expectations and using $\|\Sigma_{s-p+1}\|^2 = \lambda_{s-p+1}$ gives

$$\mathbb{E}\|A - \hat{A}_{\text{nys}}\| \leq \lambda_{s-p+1} + \mathbb{E}\|\Sigma_{s-p+1}\Omega_2\Omega_1^\dagger\|^2.$$

Using the law of total expectation, the second term may be bounded as follows

$$\begin{aligned}
\mathbb{E}\|\Sigma_{s-p+1}\Omega_2\Omega_1^\dagger\|^2 &= \mathbb{E}\left(\mathbb{E}_{\Omega_1}\left[\|\Sigma_{s-p+1}\Omega_2\Omega_1^\dagger\|^2\right]\right) \\
&\overset{(a)}{\leq} \mathbb{E}\left(\|\Sigma_{s-p+1}\|\,\|\Omega_1^\dagger\|_F + \|\Sigma_{s-p+1}\|_F\,\|\Omega_1^\dagger\|\right)^2 \\
&\overset{(b)}{\leq} 2\|\Sigma_{s-p+1}\|^2\mathbb{E}\|\Omega_1^\dagger\|_F^2 + 2\|\Sigma_{s-p+1}\|_F^2\mathbb{E}\|\Omega_1^\dagger\|^2 \\
&\overset{(c)}{\leq} \frac{2(s-p)}{p-1}\lambda_{s-p+1} + \frac{2\mathrm{e}^2 s}{p^2-1}\left(\sum_{j>s-p}\lambda_j\right),
\end{aligned}$$

where in step (a) we use Squared Chevet (Proposition 2.8.3). In step (b) we invoke the elementary identity $(a+b)^2 \leq 2a^2 + 2b^2$, and in step (c) we apply the bounds from Proposition 2.8.1. Inserting the above display into the bound for $\mathbb{E}\|A - \hat{A}_{\text{nys}}\|$ yields

$$\mathbb{E}\|A - \hat{A}_{\text{nys}}\| \leq \left(1 + \frac{2(s-p)}{p-1}\right)\lambda_{s-p+1} + \frac{2\mathrm{e}^2 s}{p^2-1}\left(\sum_{j>s-p}\lambda_j\right).$$

As the bound above holds for any $2 \leq p \leq s-2$, we may take the minimum over admissible p to conclude the result. □

2.8.3 Proof of Squared Chevet

In this subsection we provide a proof of Proposition 2.8.3. The proof is based on a Gaussian comparison inequality argument, a standard technique in the high dimensional probability literature.

Proof. Let

$$U = \{S^T a : \|a\|_2 = 1\} \subset \mathbb{R}^m$$

$$V = \{Tb : \|b\|_2 = 1\} \subset \mathbb{R}^n$$

and for $u \in U$, $v \in V$ consider the Gaussian processes

$$Y_{uv} = \langle u, Gv\rangle + \|S\|\,\|v\|\gamma \qquad \text{and} \qquad X_{uv} = \|S\|\langle h, v\rangle + \|v\|\langle g, u\rangle,$$

where

- $G \in \mathbb{R}^{m\times n}$ is a Gaussian random matrix,
- g, h are Gaussian random vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively,
- and γ is $N(0,1)$ in $\mathbb{R}$.

Furthermore, G, g, h and γ are all independent.

A standard calculation shows that the conditions of Slepian's lemma [99, Corollary 3.12, p. 72] are satisfied. Hence we conclude that

$$\mathbb{P}\left(\max_{u,v} Y_{uv} > t\right) \leq \mathbb{P}\left(\max_{u,v} X_{uv} > t\right). \tag{2.27}$$

We are now ready to prove Proposition 2.8.3. Throughout the argument below, we use the notation $X_+ = \max\{X, 0\}$.

We first observe by Jensen's inequality with respect to γ and the variational characterization of the singular values that

$$\begin{aligned}\mathbb{E}\max_{u,v}(Y_{uv})_+^2 &= \mathbb{E}\max_{\|a\|=1,\|b\|=1}\left(\langle S^Ta, GTb\rangle + \|S\|\|Tb\|\gamma\right)_+^2 \\ &\geq \mathbb{E}_G\max_{\|a\|=1,\|b\|=1}\left(\langle S^Ta, GTb\rangle^2\right)_+ = \mathbb{E}_G\|SGT\|^2.\end{aligned}$$

Hence $\mathbb{E}_G\|SGT\|^2$ is majorized by $\mathbb{E}\max_{u,v}(Y_{uv})_+^2$. For X_{uv}, we note that

$$\begin{aligned}\mathbb{E}\max_{u,v}(X_{uv})_+^2 \leq \mathbb{E}\max_{u,v}X_{uv}^2 &= \mathbb{E}\max_{\|a\|=1,\|b\|=1}\left(\|S\|\langle h, Tb\rangle + \|Tb\|\langle g, S^Ta\rangle\right)^2 \\ &\overset{(a)}{\leq} \mathbb{E}\left(\|S\|^2\|T^Th\|^2 + 2\|S\|\|T\|\|T^Th\|\|Sg\| + \|T\|^2\|Sg\|^2\right) \\ &\overset{(b)}{\leq} \|S\|^2\|T\|_F^2 + 2\|S\|\|T\|\|S\|_F\|T\|_F + \|T\|^2\|S\|_F^2 \\ &= (\|S\|\|T\|_F + \|T\|\|S\|_F)^2,\end{aligned}$$

where in step (a) we expand the quadratic and use Cauchy-Schwarz. Step (b) follows from a straightforward calculation and Hölder's inequality.

To conclude, we use integration by parts and (2.27) to obtain

$$\begin{aligned}\mathbb{E}_G\|SGT\|^2 \leq \mathbb{E}\max_{u,v}(Y_{uv})_+^2 &= \int_0^\infty t\mathbb{P}\left(\max_{u,v}(Y_{uv})_+ > t\right)dt \\ &= \int_0^\infty t\mathbb{P}\left(\max_{u,v}Y_{uv} > t\right)dt \leq \int_0^\infty t\mathbb{P}\left(\max_{u,v}X_{uv} > t\right)dt \\ &= \int_0^\infty t\mathbb{P}\left(\max_{u,v}(X_{uv})_+ > t\right)dt = \mathbb{E}\max_{u,v}(X_{uv})_+^2 \\ &\leq (\|S\|\|T\|_F + \|T\|\|S\|_F)^2,\end{aligned}$$

completing the proof. □

2.8.4 Proof of Proposition 2.3.1

We require the following fact from [18, Chapter X] ,

Lemma 2.8.5 ([18] Lemma X.1.4.). *Let A, B be psd matrices. Then*

$$|||(A+I)^{-1} - (A+B+I)^{-1}||| \leq |||B(B+I)^{-1}|||$$

for every unitarily invariant norm.

Proof of Proposition 2.3.1. We first prove (2.11). Under the hypotheses of Proposition 2.3.1, we may strengthen Lemma 2.8.5 by scaling the identity to deduce

$$\|(\hat{A}+\mu I)^{-1}-(A+\mu I)^{-1}\| \leq \frac{1}{\mu}\|(E+\mu I)^{-1}E\|.$$

Recall that the function $f(t)=\frac{t}{t+\mu}$ is matrix monotone, so that $A \preceq B$ implies $f(A) \preceq f(B)$. As $E \preceq \|E\|I$, it follows that

$$\|(\hat{A}+\mu I)^{-1}-(A+\mu I)^{-1}\| \leq \frac{1}{\mu}\frac{\|E\|}{\|E\|+\mu}.$$

Hence we have established the desired inequality.

Next we show the bound is attained when $\hat{A}=[A]_s$. Applying the Woodbury identity, we may write

$$([A]_s+\mu I)^{-1} = V_s(\hat{\Lambda}_s+\mu I)^{-1}V_s^T+\frac{1}{\mu}(I-V_sV_s^T).$$

Using the eigendecomposition of $A=V_s\Lambda_sV_s^T+V_{n-s}\Lambda_{n-s}V_{n-s}^T$, we obtain

$$\begin{aligned}([A]_s+\mu I)^{-1}-(A+\mu I)^{-1} &= \frac{1}{\mu}(I-V_sV_s^T)-V_{n-s}(\Lambda_{n-s}+\mu I)^{-1}V_{n-s}^T\\ &= \frac{1}{\mu}V_{n-s}(I-(\Lambda_{n-s}+\mu I)^{-1})V_{n-s}^T\\ &= \frac{1}{\mu}V_{n-s}\left(\Lambda_{n-s}(\Lambda_{n-s}+\mu I)^{-1}\right)V_{n-s}^T.\end{aligned}$$

Hence

$$\|(A+\mu I)^{-1}-([A]_s+\mu I)^{-1}\| = \frac{\lambda_{s+1}}{\mu(\lambda_{s+1}+\mu)}.$$

□

2.8.5 Proof Theorem 2.4.2

This result contains the analysis of the Nyström sketch-and-solve method. We begin with Equation (2.13), which provides an error bound that compares the regularized inverse of a psd matrix A with the regularized inverse of the randomized Nyström approximation $\hat{A}_{\text{nys}}$. Since $0 \preceq \hat{A}_{\text{nys}} \preceq A$, we can apply Proposition 2.3.1 to obtain a deterministic bound for the discrepancy:

$$\|(\hat{A}_{\text{nys}}+\mu I)^{-1}-(A+\mu I)^{-1}\| \leq \frac{1}{\mu}\frac{\|E\|}{\|E\|+\mu} \quad \text{where } E=A-\hat{A}_{\text{nys}}.$$

The function $f(t)=t/(t+\mu)$ is concave, so we can take expectations and invoke Jensen's inequality to obtain

$$\mathbb{E}\|(\hat{A}_{\text{nys}}+\mu I)^{-1}-(A+\mu I)^{-1}\| \leq \frac{1}{\mu}\frac{\mathbb{E}\|E\|}{\mathbb{E}\|E\|+\mu}.$$

Inserting the bound Equation (2.10) on $\mathbb{E}\|E\|$ from Corollary 2.2.3 gives

$$\mathbb{E}\|(\hat{A}_{\text{nys}} + \mu I)^{-1} - (A + \mu I)^{-1}\| \leq \frac{1}{\mu} \cdot \frac{(3 + 4\mathrm{e}^2 \mathrm{sr}_p(A)/p)\lambda_p}{\mu + (3 + 4\mathrm{e}^2 \mathrm{sr}_p(A)/p)\lambda_p}.$$

To conclude, observe that the denominator of the second fraction exceeds $\mu + \lambda_p$.

Now, let us establish Equation (2.14), the error bound for Nyström sketch-and-solve. Introduce the solution $\hat{x}$ to the Nyström sketch-and-solve problem and the solution $x_\star$ to the regularized linear system:

$$(\hat{A}_{\text{nys}} + \mu I)\hat{w} = b \quad \text{and} \quad (A + \mu I)w_\star = b.$$

We may decompose the regularized matrix as $A + \mu I = \hat{A}_{\text{nys}} + \mu I + E$. Subtract the two equations in the last display to obtain

$$(\hat{A}_{\text{nys}} + \mu I)(\hat{w} - w_\star) - wx_\star = 0.$$

Rearranging to isolate the error in the solution, we have

$$\hat{w} - w_\star = (\hat{A}_{\text{nys}} + \mu I)^{-1} E w_\star.$$

Take the norm, apply the operator norm inequality, and use the elementary bound $\|(\hat{A}_{\text{nys}} + \mu I)^{-1}\| \leq \mu^{-1}$. We obtain

$$\frac{\|\hat{w} - w_\star\|_2}{\|w_\star\|_2} \leq \frac{\|E\|}{\mu}.$$

Finally, take the expectation and repeat the argument used to control $\mathbb{E}\|E\|/\mu$ in the proof of Theorem 2.5.1. □

2.8.6 Proof of statements for the optimal low-rank preconditioner $P_\star$

We show that $P_\star$ from Section 2.5 is the best symmetric positive definite preconditioner that acts as a multiple of the identity off V_s.

Lemma 2.8.6. *Let $\mathcal{P} = \{P : P = V_s M V_s^T + \beta(I - V_s V_s^T)$ where $\beta > 0$ and $M \in \mathbb{S}_s^+(\mathbb{R})\}$. With this parametrization, define $P_\star$ by setting $M = \frac{1}{\lambda_{s+1}+\mu}(\Lambda_s + \mu I)$ and $\beta = 1$. Then for any symmetric psd matrix A and $\mu \geq 0$,*

$$\min_{P \in \mathcal{P}} \kappa_2(P^{-1/2} A_\mu P^{-1/2}) = \frac{\lambda_{s+1} + \mu}{\lambda_n + \mu}, \tag{2.28}$$

$$P_\star = \operatorname*{argmin}_{P \in \mathcal{P}} \kappa_2(P^{-1/2} A_\mu P^{-1/2}). \tag{2.29}$$

Proof. We first prove the lefthand side of (2.28) is always at least as large as the righthand side, and then show the bound is attained by $P_\star$. Given $P \in \mathcal{P}$, we have

$$P^{-1/2}A_\mu P^{-1/2} = V_s M^{-1/2}(\Lambda_s + \mu I)M^{-1/2}V_s^T + \frac{1}{\beta}V_{n-s}(\Lambda_{n-s} + \mu I)V_{n-s}^T.$$

For any $1 \leq i, j \leq n$,

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) = \frac{\lambda_1(P^{-1/2}A_\mu P^{-1/2})}{\lambda_n(P^{-1/2}A_\mu P^{-1/2})} \geq \frac{\lambda_i(P^{-1/2}A_\mu P^{-1/2})}{\lambda_j(P^{-1/2}A_\mu P^{-1/2})}.$$

From our expression for $P^{-1/2}A_\mu P^{-1/2}$, we see that $(\lambda_{s+1} + \mu)/\gamma, (\lambda_n + \mu)/\gamma$ are eigenvalues of $P^{-1/2}A_\mu P^{-1/2}$. Hence for any $P \in \mathcal{P}$, the following inequality holds:

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \geq \frac{\lambda_{s+1} + \mu}{\lambda_n + \mu},$$

proving (2.28). Using the definition of $P_\star$, we see

$$\begin{aligned} P_\star^{-1/2}A_\mu P^{-1/2} &= (\lambda_{s+1} + \mu)V_s V_s^T + V_{n-s}(\Lambda_{n-s} + \mu I)V_{n-s}^T, \\ \kappa_2(P_\star^{-1/2}A_\mu P_\star^{-1/2}) &= (\lambda_{s+1} + \mu)/(\lambda_n + \mu). \end{aligned}$$

□

Proof of Proposition 2.5.3

Let $\hat{A} = U\hat{\Lambda}U^T$ be an rank-s Nyström approximation constructed from an arbitrary test matrix, whose sth eigenvalue is $\hat{\lambda}_s$. Proposition 2.5.3 provides a deterministic bound on the condition number of the regularized matrix A_μ after preconditioning with

$$P = \frac{1}{\hat{\lambda}_s + \mu}U(\hat{\Lambda} + \mu I)U^T + (I - UU^T).$$

We remind the reader that this argument is completely deterministic.

First, note that the preconditioned matrix $P^{-1/2}A_\mu P^{-1/2}$ is psd, so

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) = \frac{\lambda_1(P^{-1/2}A_\mu P^{-1/2})}{\lambda_n(P^{-1/2}A_\mu P^{-1/2})}.$$

Let us begin with the upper bound on the condition number. We have the decomposition

$$P^{-1/2}A_\mu P^{-1/2} = P^{-1/2}(\hat{A} + \mu I)P^{-1/2} + P^{-1/2}EP^{-1/2}, \tag{2.30}$$

owing to the relation $A_\mu = \hat{A} + \mu I + E$. Recall that the error matrix E is psd, so the matrix $P^{-1/2}EP^{-1/2}$ is also psd.

First, we bound the maximum eigenvalue. Weyl's inequalities imply that

$$\lambda_1(P^{-1/2}A_\mu P^{-1/2}) \leq \lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) + \lambda_1(P^{-1/2}EP^{-1/2}).$$

To determine $\lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2})$, we write $\hat{A}+\mu I = U(\hat{\Lambda}+\mu I)U^T + \mu U_\perp U_\perp^T$, where $U_\perp$ is an orthonormal basis for the eigenvectors orthogonal to U. From this and the definition of P^{-1}, we obtain.

$$P^{-1/2}(\hat{A}+\mu I)P^{-1/2} = (\hat{\lambda}_s+\mu)UU^T + \mu U_\perp U_\perp^T.$$

The preceding display immediately yields $\lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) = \hat{\lambda}_s + \mu$. We now turn to bounding $\lambda_1(P^{-1/2}EP^{-1/2})$. When $s < n$, we have $\lambda_1(P^{-1}) = 1$. Therefore,

$$\lambda_1(P^{-1/2}EP^{-1/2}) = \lambda_1(P^{-1}E) \leq \lambda_1(P^{-1})\lambda_1(E) = \lambda_1(E) = \|E\|.$$

In summary,

$$\lambda_1(P^{-1/2}A_\mu P^{-1/2}) \leq \hat{\lambda}_s + \mu + \|E\|. \tag{2.31}$$

For the minimum eigenvalue, we first assume that $\mu > 0$. Apply Weyl's inequality to Equation (2.30) to obtain to obtain

$$\begin{aligned}\lambda_n(P^{-1/2}A_\mu P^{-1/2}) &\geq \lambda_n(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) + \lambda_n(P^{-1/2}EP^{-1/2}) \\ &\geq \lambda_n(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) = \mu.\end{aligned} \tag{2.32}$$

Combining Equation (2.31) and Equation (2.32), we reach

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq \frac{\hat{\lambda}_s + \mu + \|E\|}{\mu}.$$

This gives a bound for the maximum in case $\mu > 0$.

If we only have $\mu \geq 0$, then a different argument is required for the smallest eigenvalue. Assume that A is positive definite, in which case $\hat{\lambda}_s > 0$. As $P^{-1/2}A_\mu P^{-1/2}$ is symmetric positive definite we have

$$\lambda_n(P^{-1/2}A_\mu P^{-1/2}) = \frac{1}{\lambda_1(P^{1/2}A_\mu^{-1}P^{1/2})}.$$

Conjugating by $A_\mu^{1/2}P^{-1/2}$ and using similarity, we obtain the equality

$$\lambda_1(P^{1/2}A_\mu^{-1}P^{1/2}) = \lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2}).$$

Hence it suffices to produce an upper bound for $\lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2})$. To that end, we expand

$$\begin{aligned}\lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2}) &= \lambda_1\left(A_\mu^{-1/2}\left(\frac{1}{\hat{\lambda}_s+\mu}(\hat{A}+\mu UU^T)+(I-UU)^T\right)A_\mu^{-1/2}\right)\\ &\leq \frac{1}{\hat{\lambda}_s+\mu}\lambda_1\left(A_\mu^{-1/2}(\hat{A}+\mu UU^T)A_\mu^{-1/2}\right)\\ &+\lambda_1\left(A_\mu^{-1/2}\left(I-UU^T\right)A_\mu^{-1/2}\right).\end{aligned}$$

The second inequality is Weyl's. Since $\hat{A} \preceq A$, we have $\hat{A}+\mu UU^T \preceq A_\mu$. The last display simplifies to

$$\lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2}) \leq \frac{1}{\hat{\lambda}_s+\mu}+\frac{1}{\lambda_n+\mu}.$$

Putting the pieces together with Equation (2.31), we obtain

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq (\hat{\lambda}_s+\mu+\|E\|)\left(\frac{1}{\hat{\lambda}_s+\mu}+\frac{1}{\lambda_n+\mu}\right).$$

Thus,

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq \left(\hat{\lambda}_s+\mu+\|E\|\right)\min\left\{\frac{1}{\mu},\ \frac{\hat{\lambda}_s+\lambda_n+2\mu}{(\hat{\lambda}_s+\mu)(\lambda_n+\mu)}\right\}.$$

This formula is valid when A is positive definite or when $\mu > 0$.

We now prove the lower bound on $\kappa_2(P^{-1/2}A_\mu P^{-1/2})$. Returning to Equation (2.30) and invoking Weyl's inequalities yields

$$\lambda_1(P^{-1/2}A_\mu P^{-1/2}) \geq \lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2})+\lambda_n(P^{-1/2}EP^{-1/2}) \geq \hat{\lambda}_s+\mu.$$

For the smallest eigenvalue we observe that

$$\lambda_n(P^{-1/2}A_\mu P^{-1/2}) = \lambda_n(A_\mu P^{-1}) \leq \lambda_n(A_\mu)\lambda_1(P^{-1}) = \lambda_n+\mu.$$

Where the last inequality in the preceding display follows from the identity

$$\lambda_j(AB) \leq \lambda_j(A)\lambda_1(B),$$

which holds for symmetric positive definite matrices A and B. Combining the last two displays, we obtain

$$\frac{\hat{\lambda}_s+\mu}{\lambda_n+\mu} \leq \kappa_2(P^{-1/2}A_\mu P^{-1/2}).$$

Condition numbers always exceed one, so

$$\max\left\{\frac{\hat{\lambda}_s+\mu}{\lambda_n+\mu}, 1\right\} \leq \kappa_2(P^{-1/2}A_\mu P^{-1/2}).$$

This point concludes the argument.

Proof of Lemma 2.5.4

Lemma 2.5.4 establishes the central facts about the effective dimension. First, we prove Item 1. Fix a parameter $\gamma \geq 1$, and set $j_\star = \max\{1 \leq j \leq n : \lambda_j > \gamma\mu\}$. We can bound the effective dimension below by the following mechanism.

$$d_{\mathrm{eff}}^{\mu} = \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_j+\mu} \geq \sum_{j=1}^{j_\star} \frac{\lambda_j}{\lambda_j+\mu} \geq j_\star \cdot \frac{\lambda_{j_\star}}{\lambda_{j_\star}+\mu}.$$

We have used the fact that $t \mapsto t/(1+t)$ is increasing for $t \geq 0$, Solving for $j_\star$, we determine that

$$j_\star \leq (1+\mu/\lambda_{j_\star})d_{\mathrm{eff}}^{\mu} < (1+\gamma^{-1})d_{\mathrm{eff}}^{\mu}.$$

The last inequality depends on the definition of $j_\star$. This is the required result.

Item 2 follows from a short calculation:

$$\begin{aligned}\frac{1}{k}\sum_{j>k}\lambda_j &= \frac{\lambda_k+\mu}{k}\sum_{j>k}\frac{\lambda_j}{\lambda_k+\mu} \leq \frac{\lambda_k+\mu}{k}\sum_{j>k}\frac{\lambda_j}{\lambda_j+\mu} \\ &= \frac{\lambda_k+\mu}{k}\left(d_{\mathrm{eff}}^{\mu} - \sum_{j=1}^{k}\frac{\lambda_j}{\lambda_j+\mu}\right) \leq \frac{\lambda_k+\mu}{k}\left(d_{\mathrm{eff}}^{\mu} - \frac{k\lambda_k}{\lambda_k+\mu}\right) \\ &= \frac{\mu\, d_{\mathrm{eff}}^{\mu}}{k} + \lambda_k\left(\frac{d_{\mathrm{eff}}^{\mu}}{k}-1\right) \leq \frac{\mu\, d_{\mathrm{eff}}^{\mu}}{k}.\end{aligned}$$

The last inequality depends on the assumption that $k \geq d_{\mathrm{eff}}^{\mu}$.

2.8.7 Proof of Corollary 2.5.2

This result gives a bound for the relative error δ_t in the iterates of PCG. Recall the standard convergence bound for CG [160, Theorem 38.5]:

$$\delta_t \leq 2\left(\frac{\sqrt{\kappa_2(P^{-1/2}A_\mu P^{-1/2})}-1}{\sqrt{\kappa_2(P^{-1/2}A_\mu P^{-1/2})}+1}\right)^t.$$

We conditioned on the event that $\{\kappa(P^{-1/2}A_\mu P^{-1/2}) \leq 56\}$. On this event, the relative error must satisfy

$$\delta_t < 2\left(\frac{\sqrt{56}-1}{\sqrt{56}+1}\right)^t \leq 2\cdot(0.77)^t.$$

Solving for t, we see that $\delta_t < \epsilon$ when $t \geq \lceil 3.8\log(1/\epsilon)\rceil$. This concludes the proof.

2.8.8 Proof of Theorem 2.5.5

Theorem 2.5.5 establishes that with high probability Algorithm 5 terminates in a logarithmic number of steps, the sketch size remains $\mathcal{O}(d_{\text{eff}}^{\delta\tau\mu})$, and PCG with the preconditioner constructed from the output converges fast.

Proof. We first recall with the tolerances chosen in Theorem 2.5.5 that Algorithm 5 terminates when the event

$$\mathcal{E} = \{\|E\| \leq \tau\mu\} \cap \left\{\hat{\lambda}_s \leq \frac{\tau\mu}{11}\right\}$$

holds. Observe that conditioned on $\mathcal{E}$, Proposition 2.5.3 yields

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq \frac{\hat{\lambda}_s + \mu + \|E\|}{\mu} \leq 1 + \left(1 + \frac{1}{11}\right)\tau = 1 + \frac{12}{11}\tau.$$

Statement 3 now follows from the above display and the standard convergence theorem for CG.

Now, if Algorithm 5 terminates with $N \leq \lceil \log_2(\tilde{s}/s_0)\rceil - 1$ steps of sketch size doubling, then $\mathcal{E}$ holds with probability 1. Statement 3 then follows by our initial observation, while statements 1 and 2 hold trivially. Hence statements 1-3 all hold if the algorithm terminates in $N \leq \lceil \log_2(\tilde{s}/s_0)\rceil - 1$ steps.

Thus to conclude the proof, it suffices to show that if $N \geq \lceil \log_2(\tilde{s}/s_0)\rceil$, then $\mathcal{E}$ holds with probability at least $1-\delta$, which implies that statements 1-3 hold with probability at least $1-\delta$, as above.

We now show that $\mathcal{E}$ holds with probability at least $1-\delta$ when $N = \lceil \log_2(\tilde{s}/s_0)\rceil$. To see this note that when $N = \lceil \log_2(\tilde{s}/s_0)\rceil$, we have $s \geq \tilde{s}$. Consequently, we may invoke Proposition 2.2.2 with $p = \lceil 2d_{\text{eff}}^{\delta\tau\mu/11}\rceil + 1$ and Lemma 2.5.4 to show

$$\begin{aligned}
\mathbb{E}[\|E\|] &\overset{(1)}{\leq} 3\lambda_p + \frac{2\mathrm{e}^2}{p}\left(\sum_{j=p}^{n}\lambda_j\right) \\
&\overset{(2)}{\leq} 3\frac{\delta\tau\mu}{11} + 2\mathrm{e}^2\frac{d_{\text{eff}}(\delta\tau\mu/11)}{p}\frac{\delta\tau\mu}{11} \\
&\overset{(3)}{\leq} \frac{3}{11}\delta\tau\mu + \frac{2\mathrm{e}^2}{2}\frac{\delta\tau\mu}{11} = \left(\frac{3+\mathrm{e}^2}{11}\right)\delta\tau\mu \leq \delta\tau\mu.
\end{aligned}$$

Where step (1) uses Proposition 2.2.2, step (2) uses items 1 and 2 of Lemma 2.5.4 with $\gamma = 1$, and step (3) follows from $p \geq 2d_{\mathrm{eff}}^{\delta\tau\mu}$. Thus,

$$\mathbb{E}[\|E\|] \leq \delta\tau\mu.$$

By Markov's inequality,

$$\mathbb{P}\{\|E\| > \tau\mu\} \leq \delta.$$

Hence $\{\|E\| \leq \tau\mu\}$ holds with probability at least $1-\delta$. Furthermore, by Lemma 2.5.4 we have $\{\hat{\lambda}_s \leq \delta\tau\mu/11\}$ with probability 1 as $\hat{\lambda}_s \leq \lambda_s \leq \lambda_p$. Thus when $N = \lceil \log_2 (\tilde{s}/s_0) \rceil$, $\mathcal{E}$ holds with probability at least $1-\delta$, this immediately implies statements 1 and 3. Statement 2 follows as

$$s = 2^N s_0 \leq 2^{\log_2(\tilde{s}/s_0)+1} s_0 = 2\tilde{s} = 4\lceil 2d_{\mathrm{eff}}^{\delta\tau\mu/11} \rceil + 2,$$

where in the first inequality we used $\lceil x \rceil \leq x+1$, this completes the proof. □

2.8.9 Proof of Proposition 2.5.7

Proposition 2.5.7 shows once $s = \Omega\left(d_{\mathrm{eff}}(\tau\mu)\right)$, then with high probability $\kappa_2(P^{-1/2}A_\mu P^{-1/2})$ differs from $(\hat{\lambda}_s + \mu)$ by at most a constant.

Proof. Proposition 2.5.3 implies that

$$\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_s + \mu}{\mu}\right)_+ \leq \frac{\|E\|}{\mu}.$$

Combining the previous display with Markov's inequality yields

$$\mathbb{P}\left\{\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_s + \mu}{\mu}\right)_+ > \frac{\tau}{\delta}\right\} \leq \frac{\delta}{\tau}\frac{\mathbb{E}[\|E\|]}{\mu}.$$

Now, our choice of s combined with Proposition 2.2.2 and Lemma 2.5.4 implies that $\mathbb{E}[\|E\|] \leq \tau\mu$. Hence we have

$$\mathbb{P}\left\{\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_s + \mu}{\mu}\right)_+ > \frac{\tau}{\delta}\right\} \leq \delta,$$

which implies the desired claim. □

Table 2.9: **Ridge regression: Test set error.** We report the relative error for regression tasks and the misclassification rate for classification tasks. Nyström PCG outperforms the Nyström preconditioner on nearly all the datasets.

Dataset	Method	Test set error
CIFAR-10	Nyström Preconditioner	9.51%
	Nyström PCG	**8.67**%
Guillermo	Nyström Preconditioner	**32.5**%
	Nyström PCG	32.6%
shuttle-rf	Nyström Preconditioner	**0.20**%
	Nyström PCG	0.22%
smallnorb-rf	Nyström Preconditioner	57.92%
	Nyström PCG	**16.14**%
YearMSD-rf	Nyström Preconditioner	5.48e–3
	Nyström PCG	**4.55e–3**
Higgs-rf	Nyström Preconditioner	3.49%
	Nyström PCG	**0.05**%
Covtype-rf	Nyström Preconditioner	20.76%
	Nyström PCG	**9.39**%

2.9 Additional numerical results

2.9.1 Ridge regression experiments

Here, we report the test error obtained on datasets considered in Section 2.6.2 and the implications of these results.

Table 2.9 compares the test error obtained using the Nyström PCG solution with that of a sketch-and-solve approach we call *Nyström preconditioner*, which uses P^{-1} (the inverse of the Nyström preconditioner) to approximate $(1/nX^TX + \mu I)^{-1}$.
Nyström PCG outperforms, especially on the larger datasets $n \geq 10^5$, and even for datasets where the effective dimension is small, such as Higgs-rf. Hence even in the statisical learning setting, where one only cares about test error, solving the ridge regression problem accurately improves statistical performance.

2.10 Adapative rank selection via a-posteriori error estimation

2.10.1 Randomized powering algorithm

The pseudo-code for estimating $\|E\|$ by the randomized power method is given in Algorithm 4

The pseudocode for adaptive rank selection by a-priori error estimation is given in Algorithm 5. The code is structured to reuse use the previously computed Ω and Y, resulting in significant computational savings. The error $\|E\|$ is estimated from q iterations of the randomized power method on the error matrix $A - U\hat{\Lambda}U^T$.

Algorithm 4 Randomized Power method for estimating $\|E\|$

```
Input: symmetric PSD matrix $A \in \mathbb{R}^{n\times n}$, approximate eigenvectors $U$, approximate eigenvalues $\hat{\Lambda}$, and number of power iterations $q$.
$g = \mathrm{randn}(n, 1)$
$v_0 = \frac{g}{\|g\|_2}$
for $i = 1, \dots, q$ do
    $v = Av_0 - U(\hat{\Lambda}(U^T v_0))$
    $\hat{E} = v_0^T v$
    $v = \frac{v}{\|v\|_2}$
    $v_0 \leftarrow v$
end for
Output: estimate $\hat{E}$ of $\|E\|$
```

2.11 Additional experimental details

Here we provide additional details on the experimental procedure and the methods we compared to.

2.11.1 Ridge regression experiments

All of the datasets used in our ridge regression experiments are classification datasets. We converted them to regression problems by using a one-hot vector encoding. The target vector b was constructed by setting $b_i = 1$ if example i has the first label and 0 otherwise. We did no data pre-processing except on CIFAR-10, where we scaled the matrix by 255 so that all entries lie in $[0, 1]$.

We now give an overview of the hyperparameters of each method. The R&T preconditioner has only one hyperparameter: the sketch size s_{RT}. AdaIHS has five hyperparameters: $\rho, \lambda_\rho, \Lambda_\rho, \mu_{\mathrm{gd}}(\rho)$, and $c_{\mathrm{gd}}(\rho)$. The hyperparameter $\rho \in (0, 1)$ controls the remaining four hyperparameters, which are set to the values recommended in [95]. For the regularization path experiments, s_{RT} and ρ were chosen by grid search to minimize the time taken to solve the linear systems over the regularization path. We chose s_{RT} from the linear grid jd, where $j \in \{1, \dots, 8\}$. Additionally, we restrict $j \leq 4$ for Guillermo as $jd \geq n$ when $j \geq 5$, and hence no benefit is gained over a direct method. For AdaIHS, ρ was chosen from the linear grid $\rho = j \times 10^{-1}$ where $j \in \{1, \dots, 9\}$. We set the initial sample size for AdaIHS to $s = 100$ for both sets of experiments.

We reused computation as much as possible for both R&T and AdaIHS, which we now detail. To construct the R&T preconditioner, we incur a $\mathcal{O}(nd\log(n) + s_{\mathrm{RT}}d^2)$ to cache the Gram matrix and pay an $\mathcal{O}(d^3)$ to update the preconditioner for each value of μ. In the case of AdaIHS, for each value of μ we cache the sketch SA and the corresponding Gram matrix. We then use them for the next value of μ on the path until the adapativity criterion of the algorithm deems a new sketch necessary. For AdaIHS computing the sketch only costs $O(nd\log(n))$.

We now give the parameters for the random features experiments. For Shuttle-rf we used random features corresponding to a Gaussian kernel with bandwidth parameter $\sigma = 0.75$, we set $\mu = 10^{-8}/n$.

For smallNORB-rf we used ReLU random features with $\mu = 6 \times 10^{-4}$. We selected the AdaIHS parameter ρ from the same grid used for the ridge regression experiments. We also capped the sketch size for AdaIHS at $s_{\max} = 10,000$.

Finally, we give the details of our implementation of Nyström PCG. For both sets of experiments we used Algorithm 5 initialized at $s = 100$, with an error tolerance of 30μ, and $q = 5$ power iterations. To avoid trivialities, the rank of the preconditioner is capped at $s_{\max} = 0.5d$ for CIFAR-10 and $s_{\max} = 0.4d$ for Guillermo. For the random features experiments we capped s at $s_{\max} = 2000$. In the regularization path experiments, we keep track of the latest estimate $\hat{E}$ of $\|E\|$, and do not compute a new Nyström approximation unless $\hat{E}$ is larger than the error tolerance for the new regularization parameter. When we compute the new Nyström approximation, the adaptive algorithm is initialized with a target rank of twice the old one.

The values of hyperparameters used for all experiments are summarized in Table 2.10.

Table 2.10: **Ridge regression: Experimental parameters.**

Dataset	**(R&T) sketch size**	**AdaIHS rate**	**Initial AdaIHS sketch size**	**Initial Nyström rank**
CIFAR-10	$3d$	$\rho = 0.3$	100	100
Guillermo	d	$\rho = 0.3$	100	100
shuttle-rf	NA	$\rho = 0.1$	100	100
smallNORB-rf	NA	$\rho = 0.3$	100	100

2.11.2 ALOOCV

The datasets were chosen so that n and d are both large, the challenging regime for ALOOCV. The first three datasets are binary classification problems, while SVHN has multiple classes. For SVHN we created a binary classification problem by looking at the first class vs. remaining classes.

For the large scale problems the adaptive algorithm for Nyström PCG was initialized at $s_0 = 500$ and is capped at $s_{\max} = 4000$. We set the solve tolerances for both algorithms to 10^{-10}. As before, we sample 100 points randomly from each dataset.

2.11.3 Kernel ridge regression

We converted the binary classification problem to a regression problem by constructing the target vector as follows: We assign +1 to the first class and -1 to the second class. For multi-class problems, we do one-vs-all classification; this formulation leads to multiple right hand sides, so we use block PCG for both methods. We did no data pre-processing except for MNIST, whose data matrix was scaled by 255 so that its entries lie in $[0, 1]$. The number of random features, m_{rf} from the linear grid

$m_{\text{rf}} = j \times 10^3$ for $j = 1, \ldots, 9$. For adaptive Nyström PCG we capped the maximum rank for the preconditioner at $s_{\max} = \lfloor 0.1n \rfloor$ and used a tolerance of 40 for the ratio $\hat{\lambda}_s/n\mu$ on all datasets.

2.12 Additional numerical results

Here we include some additional numerical results not appearing in the main paper.

2.12.1 ALOOCV

Table 2.11 contains more details about the preconditioner and preconditioned system for the large scale ALOOCV experiments in Section 2.6.3. The original condition number in Table 2.11 below is estimated as follows. First we compute the top eigenvalue of the Hessian using Matlab's eigs() command, then we divide this by μ.

Dataset	**Nyström rank**	**Preconditioner construction time(s)**	**Condition number estimate**	**Preconditioned condition number estimate**
rcv1 ($\mu = 1 \times 10^{-4}$)	1000	19.5 (0.523)	21.6	2.98 (0.081)
rcv1 ($\mu = 1 \times 10^{-8}$)	4000	100.6 (3.46)	5.70e+3	17.2 (0.218)
realsim ($\mu = 1 \times 10^{-4}$)	3100 (1.41e+3)	82.01(2.04)	10.0	1.70 (0.2324)
realsim ($\mu = 1 \times 10^{-8}$)	4000	108.3 (6.21)	2.13e+4	62.4 (0.945)

Table 2.11: **ALOOCV: additional details for large-scale experiments.** For $\mu = 10^{-4}$ the Hessian is well-conditioned for both datasets, so there is little value to preconditioning. For $\mu = 10^{-8}$, the ill-conditioning of the Hessian increases significantly, making preconditioning more valuable. Furthermore, as ALOOCV uses Block PCG on at least several batches of data points, the cost of constructing the preconditioner is negligible compared to the cost of solving the linear systems (see Table 2.6 in Section 2.6.3).

Algorithm 5 Adaptive Randomized Nyström Approximation

```
**Input:** symmetric psd matrix $A$, initial rank $s_0$, maximum rank $s_{\max}$, number of power iterations
  for estimating $E$, error tolerance $\mathrm{Tol}_{\mathrm{Err}}$, ratio tolerance $\mathrm{Tol}_{\mathrm{Rat}}$
$Y = [\ ], \Omega = [\ ]$, $E = \mathrm{Inf}$, and $\hat{\lambda}_s = \mathrm{Inf}$
$m = s_0$
**while** $E > \mathrm{Tol}_{\mathrm{Err}}$ and $\hat{\lambda}_s/\mu > \mathrm{Tol}_{\mathrm{Rat}}$ **do**
  Generate Gaussian test matrix $\Omega_0 \in \mathbb{R}^{n \times m}$
  $[\Omega_0, \sim] = \mathrm{qr}(\Omega_0, 0)$
  $Y_0 = A\Omega_0$
  $\Omega = [\Omega\ \ \Omega_0]$ and $Y = [Y\ \ Y_0]$
  $\nu = \sqrt{n}\,\mathrm{eps}(\mathrm{norm}(Y, 2))$
  $Y_\nu = Y + \nu\Omega$,
  $C = \mathrm{chol}(\Omega^T Y_\nu)$
  $B = Y_\nu / C$
  Compute $[U, \Sigma, \sim] = \mathrm{svd}(B, 0)$
  $\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$ {remove shift}
  $E = \mathrm{RandomizedPowerErrEst}(A, U, \hat{\Lambda}, q)$ {estimate error}
  $m \leftarrow s_0$, $s_0 \leftarrow 2s_0$ {double rank if tolerances are not met}
  **if** $s_0 > s_{\max}$ **then**
    $s_0 = s_0 - m$ {when $s_0 > s_{\max}$, reset to $s_0 = s_{\max}$}
    $m = s_{\max} - s_0$
    Generate Gaussian test matrix $\Omega_0 \in \mathbb{R}^{n \times m}$
    $[\Omega_0, \sim] = \mathrm{qr}(\Omega_0, 0)$
    $Y_0 = A\Omega_0$
    $\Omega = [\Omega\ \ \Omega_0]$ and $Y = [Y\ \ Y_0]$
    $\nu = \sqrt{n}\,\mathrm{eps}(\mathrm{norm}(Y, 2))$ {compute final approximation and break}
    $Y_\nu = Y + \nu\Omega$,
    $C = \mathrm{chol}(\Omega^T Y_\nu)$
    $B = Y_\nu / C$
    Compute $[U, \Sigma, \sim] = \mathrm{svd}(B, 0)$
    $\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$
    **break**
  **end if**
**end while**
**Output:** Nyström approximation $(U, \hat{\Lambda})$
```

Chapter 3

NysADMM

3.1 Introduction

Consider the composite convex optimization problem

$$\text{minimize}_{w \in \mathbb{R}^d} \ \ell(Xw; b) + r(w). \tag{3.1}$$

We assume that ℓ and r are convex and ℓ is smooth. In machine learning, generally ℓ is a loss function, r is a regularizer, $X \in \mathbb{R}^{n \times d}$ is a feature matrix, and $b \in \mathbb{R}^n$ is the label or response. Throughout the paper we assume that a solution to (3.1) exists. A canonical example of (3.1) is the lasso problem,

$$\text{minimize } \frac{1}{2}\|Xw - b\|_2^2 + \gamma\|w\|_1, \tag{3.2}$$

where $\ell(Xw; b) = \frac{1}{2}\|Xw - b\|_2^2$ and $r(x) = \gamma\|w\|_1$. We discuss more applications of (3.1) in Section 3.3.

The alternating directions method of multipliers (ADMM) is a popular algorithm to solve optimization problems of the form (3.1). However, when the matrix X is large, each iteration of ADMM requires solving a large subproblem. For example, consider the lasso where the loss ℓ is quadratic. At each iteration, ADMM solves a regularized least-squares problem at a cost of $\mathcal{O}(nd^2)$ flops. On the other hand, it is not necessary to solve each subproblem exactly to ensure convergence: ADMM strategies that solve the subproblems inexactly are called inexact ADMM, and can be shown to converge when the sequence of errors is summable [48]. Unfortunately, it can be challenging even to satisfy this relaxed criterion. Consider again the lasso problem. At each iteration, inexact ADMM solves the regularized least-squares subproblem (3.4) approximately, for example, using the iterative method of conjugate gradients (CG). We call this method inexact ADMM with CG. The number of CG iterations required to achieve accuracy ϵ increases with the square root of the condition number κ_2 of the regularized Hessian, $\mathcal{O}\left(\sqrt{\kappa_2}\log(\frac{\kappa_2}{\epsilon})\right)$. Alas, the condition number of large-scale data matrices is generally high, and later iterations of inexact ADMM require high accuracy, so inexact ADMM with CG still converges too slowly to be practical.

In this work we show how to speed up inexact ADMM using preconditioned conjugate gradients (PCG) as a subproblem solver. We will precondition with the randomized Nyström preconditioner from Chapter 2. We call the resulting algorithm NysADMM ("nice ADMM"): inexact ADMM with PCG using randomized Nyström preconditioning. As shown in Chapter 2, the Nyström preconditioner reduces the number of iterations required to solve the subproblem to ϵ-accuracy to $\mathcal{O}\left(\log(\frac{1}{\epsilon})\right)$, independent of the condition number. To make Nyström PCG applicable when ℓ is not quadratic, NysADMM uses linearized inexact ADMM to transform the subproblem into a linear system solve.

3.1.1 Contributions

1. We provide a general algorithmic framework for solving large scale lasso, l_1-regularized logistic regression, and SVM problems.

2. Our theory shows that at each iteration, modulo logarithmic factors, only a constant number of matrix vector products (matvecs) are required to solve the ADMM subproblem, provided we have constructed the preconditioner appropriately. If the loss function is quadratic, modulo logarithmic factors, only a constant number of matvecs are required to achieve convergence.

3. We develop a practical adaptive algorithm that increases the rank until the conditions of our theory are met, which ensures the theoretical benefits of the method can be realized in practice.

4. Even a preconditioner with lower rank often succeeds in speeding up inexact ADMM with PCG. Our analysis is also able to explain this phenomenon.

5. Our algorithm beats standard solvers such as glmnet, SAGA, and LIBSVM on large dense problems like lasso, logistic regression, and kernalized SVMs: it yields equally accurate solutions and often runs 2–4 times faster.

3.1.2 Related work

Our work relies on recent advancements in RandNLA for solving regularized least squares problems $(X^TX + \mu I)w = X^Tb$ for w, given a design matrix $X \in \mathbb{R}^{n\times d}$, righthand side $b \in \mathbb{R}^n$, and regularization $\mu \in \mathbb{R}$, using a *sketch* of the design matrix A [95]. NysADMM adapts the randomized Nyström preconditioner from Chapter 2. Recall, these algorithms begin by forming a *sketch* $Y = X\Omega$ of X (or X^T) with a random dimension reduction map $\Omega \in \mathbb{R}^{d\times s}$ [108, 171]. For example, Ω may be chosen to have iid Gaussian entries. These algorithms obtain significant computational speedups by using a sketch size $s \ll \min\{n, d\}$ and working with the sketch in place of the original matrix to construct a preconditioner for the linear system. Chapter 2 along with the work [95] show that these randomized preconditioners work well when the sketch size grows with the *effective dimension*

(Equation (2.19)) of the Gram matrix (assuming, for [95] that we have access to a matrix square root). As the effective dimension is never larger than d and often significantly smaller, these results substantially improve on prior work in randomized preconditioning [113, 143] that requires a sketch size $s \gtrsim d$. Many applications require even smaller sketch sizes: for example, for NysADMM, a fixed sketch size $s = 50$ suffices even for extremely large problems.

We are not the first to use RandNLA to accelerate iterative optimization. [71, 131] both use iterative sketching to accelerate Newton's method, while [32] use randomized preconditioning to accelerate interior point methods for linear programmming. The approach taken here is closest in spirit to [32], as we also use randomized preconditioning. However, the preconditioner used in [32] requires the data matrix to have many more columns than rows, while ours can handle any (sufficiently large) dimensions.

NysADMM can solve many traditional machine learning problems, such as lasso, regularized logistic regression, and support vector machines (SVMs). In contrast, standard solvers for these problems use a wider variety of convex optimization techniques. For example, one popular lasso solver, glmnet [60], relies on coordinate descent (CD), while solvers for SVMs, such as LIBSVM [29], more often use sequential minimal optimization [132], a kind of pairwise CD on the dual problem. For regularized logistic regression, especially for l_1-regularization, stochastic gradient algorithms are most commonly used [38, 150]. Other authors propose to solve lasso with ADMM [23, 177]. Our work, motivated by the ADMM quadratic programming framework of [154], is the first to accelerate ADMM with randomized preconditioning, thereby improving on the performance of standard CD or stochastic gradient solvers for each of these important classes of machine learning problems on large-scale dense data. Unlike [154], our work relies on inexact ADMM and can handle non-quadratic loss functions, which allows NysADMM to solve problems such as regularized logistic regression.

3.1.3 Organization of the paper

Section 3.2 introduces the NysADMM algorithm. Section 3.3 lists a variety of applied problems that can be solved by NysADMM. Section 3.4 states the theoretical guarantees for NysADMM. Section 3.5 compares NysADMM and standard optimization solvers numerically on several applied problems. Section 4.6 summarizes the results of the paper and discusses directions for future work.

3.1.4 Notation and preliminaries

We call a matrix psd if it is positive semidefinite. The notation $a \gtrsim b$ means that $a \geq Cb$ for some absolute constant C. Given a matrix H, we denote its spectral norm by $\|H\|$. We denote the Moore-Penrose pseudoinverse of a matrix M by $M^{\dagger}$. For $\rho > 0$ and a symmetric psd matrix H, we define $H_\rho = H + \rho I$. We say a positive sequence $\{\varepsilon^k\}_{k=1}^{\infty}$ is summable if $\sum_{k=1}^{\infty} \varepsilon^k < \infty$. We denote the Loewner ordering on the cone of symmetric psd matrices by $\preceq$, that is $A \preceq B$ if and only if $B - A$ is psd.

3.2 Algorithm

3.2.1 Inexact linearized ADMM

To solve problem (3.1), we apply the ADMM framework. Algorithm 6 presents the standard ADMM updates.

Algorithm 6 ADMM

```
**Input:** design matrix $X$, response $b$, loss function $\ell$, regularization $r$, stepsize $\rho$
**repeat**
    $w^{k+1} = \mathrm{argmin}_w\{\ell(Xw; b) + \frac{\rho}{2}\|w - z^k + u^k\|_2^2\}$
    $z^{k+1} = \mathrm{argmin}_z\{r(z) + \frac{\rho}{2}\|w^{k+1} - z + u^k\|_2^2\}$
    $u^{k+1} = u^k + w^{k+1} - z^{k+1}$
**until** convergence
**Output:** solution $w_\star$ of problem (3.1)
```

In each iteration, two subproblems are solved sequentially to update variables w and z. The z-subproblem often has a closed-form solution. For example, if $r(w) = \|w\|_1$, the z-subproblem is the soft thresholding, and if r is the indicator function of a convex set $\mathcal{C}$, the z-subproblem is projection onto the set $\mathcal{C}$.

There is usually no closed-form solution for the w-subproblem. Instead, it is usually solved inaccurately by an iterative scheme, especially for large-scale applications. To simplify the subproblem, inspired by linearized ADMM, we assume ℓ is twice differentiable and notice that the w update is close to the minimum of a quadratic function given by the Taylor expansion of ℓ at the current iterate:

$$\tilde{w}^{k+1} = \mathrm{argmin}_w \left\{(w - \tilde{w}^k)^T X^T \nabla\ell(X\tilde{w}^k; b) + \frac{1}{2}(w - \tilde{w}^k)^T X^T \nabla^2\ell(A\tilde{w}^k; b) X (w - \tilde{w}^k) + \frac{\rho}{2}\|w - \tilde{z}^k + \tilde{u}^k\|_2^2\right\}. \tag{3.3}$$

Here $\nabla^2\ell$ denotes the Hessian of ℓ. We assume throughout the chapter that $\nabla^2\ell$ is psd matrices, this is a very minor assumption, and is satisfied by all the applications we consider. The solution to this quadratic minimization may be obtained by solving the linear system

$$(X^T\nabla^2\ell(X\tilde{w}^k; b)X + \rho I)w = r^k \tag{3.4}$$

$$\text{where} \quad r^k = \rho(\tilde{z}^k - \tilde{u}^k) + X^T\nabla^2\ell(X\tilde{w}^k; b)X\tilde{w}^k - X^T\nabla\ell(A\tilde{w}^k; b). \tag{3.5}$$

The inexact ADMM algorithm we propose solves (3.4) approximately at each iteration.

Algorithm 7 Inexact ADMM

```
Input: design matrix $X$, response $b$, loss function $\ell$, regularization $r$, stepsize $\rho$, positive summable sequence $\{\varepsilon^k\}_{k=0}^{\infty}$
repeat
    find $\tilde{w}^{k+1}$ that solves (3.4) within tolerance $\varepsilon^k$
    $\tilde{z}^{k+1} = \mathrm{argmin}_z\{r(z) + \frac{\rho}{2}\|\tilde{w}^{k+1} - z + \tilde{u}^k\|_2^2\}$
    $\tilde{u}^{k+1} = \tilde{u}^k + \tilde{w}^{k+1} - \tilde{z}^{k+1}$
until convergence
Output: solution $w_\star$ of problem (3.1)
```

For a quadratic loss ℓ, when $\sum_{k=0}^{\infty} \varepsilon^k < \infty$ and under various other conditions, if optimization problem (3.1) has an optimal solution, the $\{\tilde{w}^k\}_{k=0}^{\infty}$ sequence generated by Algorithm 7 converges to the optimal solution of (3.1) [48,49]. From [23], quantity $r_d^{k+1} = \rho(\tilde{z}^k - \tilde{z}^{k+1})$ can be regarded as the dual residual and $r_p^{k+1} = \tilde{w}^{k+1} - \tilde{z}^{k+1}$ can be viewed as the primal residual at iteration $k+1$. This suggests that we can terminate the ADMM iterations when the primal and dual residuals become very small. The primal and dual tolerances can be chosen based on an absolute and relative criterion, such as

$$\|r_p^k\|_2 \leq \epsilon^{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \max\{\|\tilde{x}^k\|_2, \|\tilde{z}^k\|_2\}$$
$$\|r_d^k\|_2 \leq \epsilon^{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \|\rho \tilde{u}^k\|_2.$$

The relative criteria ϵ^{rel} might be 10^{-3} or 10^{-4} in practice. The choice of absolute criteria ϵ^{abs} depends on the scale of the variable values. More details can be found in [23].

3.2.2 Solving the w-subproblem with Nyström PCG

To efficiently solve the linearized w-subproblem in (3.4), we apply the Nyström PCG algorithm (Algorithm 3) introduced in Chapter 2. Our selection of Nyström PCG is motivated by the fact that the Hessian matrix in (3.4) has the form:

$$H = X^T \nabla^2 \ell(X\tilde{w}^k; b)X$$

that is, the Hessian is formed from the design matrix X. As most design matrices exhibit fast spectral decay, we expect the Hessian to enjoy approximate low-rank structure, which makes Nyström PCG a natural candidate for solving (3.4) efficiently.

3.2.3 NysADMM

Integrating Nyström PCG with inexact ADMM, we obtain NysADMM, presented in Algorithm 8.

Algorithm 8 NysADMM

```
**Input:** design matrix $X$, response $b$, loss function $\ell$, regularization $r$, stepsize $\rho$, positive summable sequence $\{\varepsilon^k\}_{k=0}^{\infty}$
$[U, \hat{\Lambda}] = \text{RandNyströmApprox}(X^T \nabla^2 \ell(X\tilde{w}^0; b)X, s)$ {use Algorithm 13}
**repeat**
    use Nyström PCG (Algorithm 3) to find $\tilde{w}^{k+1}$ that solves (3.4) within tolerance $\varepsilon^k$
    $\tilde{z}^{k+1} = \text{argmin}_z\{r(z) + \frac{\rho}{2}\|\tilde{w}^{k+1} - z + \tilde{u}^k\|_2^2\}$
    $\tilde{u}^{k+1} = \tilde{u}^k + \tilde{w}^{k+1} - \tilde{z}^{k+1}$
**until** convergence
**Output:** solution $w_\star$ of problem (3.1)
```

Our theory for Algorithm 8, shows that if the sketch size $s \gtrsim d_{\text{eff}}^{\rho}$, then with high probability subproblem (3.4) will be solved to ϵ-accuracy in $\mathcal{O}\left(\log(\frac{1}{\epsilon})\right)$ iterations (Corollary 3.4.2). When the loss ℓ is quadratic and the sequence of tolerances $\{\varepsilon^k\}_{k=0}^{\infty}$ is decreasing with $\sum_{k=0}^{\infty} \varepsilon^k < \infty$, NysADMM is guaranteed to converge as $k \to \infty$ with only a constant number of matvecs per iteration (Theorem 3.4.3). Table 3.1 compares the complexity of inexact ADMM with CG vs. NysADMM for K iterations under the hypotheses of Theorem 3.4.3. NysADMM achieves a significant decrease in runtime over inexact ADMM with CG, as the iteration complexity no longer depends on the condition number κ_2.

Table 3.1: **Complexity comparison for a quadratic loss with Hessian H.** Here T_{mv} is the time to compute a matrix vector product with H, $\kappa(H_\rho)$ is the condition number of $\kappa(H_\rho)$, and ε^k is the precision of the kth subproblem solve (3.4).

Method	**Complexity**
Inexact ADMM with CG	$\mathcal{O}\left(\sum_{k=1}^{K} T_{\text{mv}}\sqrt{\kappa(H_\rho)}\log\left(\frac{\kappa(H_\rho)}{\varepsilon^k}\right)\right)$
NysADMM	$\mathcal{O}\left(T_{\text{mv}} d_{\text{eff}}^{\rho}\right) +$ $\sum_{k=1}^{K} T_{\text{mv}}\left(1 + \left\lceil 2\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}R}{\varepsilon^k \rho}\right)\right\rceil\right)$

3.2.4 AdaNysADMM

Two practical problems remain in realizing the success predicted by the theoretical analysis of Table 3.1. These bounds are achieved by selecting the sketch size to be d_{eff}^{ρ}, but the effective dimension is 1) seldom known in practice, and 2) often larger than required to achieve good convergence of NysADMM. Fortunately, a simple adaptive strategy for choosing the sketch size, inspired by [58], can achieve the same guarantees as in Table 3.1. This strategy chooses a tolerance ϵ and doubles the sketch size s until the empirical condition number $\frac{\hat{\lambda}_s+\rho}{\rho}$ satisfies

$$\frac{\hat{\lambda}_s+\rho}{\rho} \leq 1+\epsilon. \tag{3.6}$$

Theorem 3.4.4 guarantees that (3.6) holds when $s \geq d_{\text{eff}}^{\rho}$ and that when (3.6) holds, the true condition number is on the order of $1+\epsilon$ with high probability. We refer to (3.6) as the empirical condition number as it provides an estimate of the true condition number of the preconditoned system (Theorem 3.4.4).

Thus, to enjoy the guarantees of Theorem 3.4.4 in practice, we may employ the adaptive version of NysADMM, which we call AdaNysADMM. We provide the pseudocode for AdaNysADMM in Algorithm 9 in Section 3.8. Furthermore, as we use a Gaussian test matrix, it is possible to construct a larger sketch from a smaller one. Hence the total computational work needed by the adaptive strategy is not much larger than if the effective dimension were known in advance. Indeed, AdaNysADMM differs from NysADMM only in the construction of the preconditioner. The dominant cost in forming the precondition is computing the sketch is $H\Omega$, which costs $\mathcal{O}(T_{\text{mv}}d_{\text{eff}}^{\rho})$. As AdaNysADMM reuses computation, the dominant complexity for constructing the Nyström preconditioner remains $\mathcal{O}(T_{\text{mv}}d_{\text{eff}}^{\rho})$. Consequently, the overall complexity of AdaNysADMM is the same as NysADMM in Table 3.1.

3.3 Applications

Here we discuss various applications that can be reformulated as instances of (3.1) and solved by Algorithm 8.

3.3.1 Elastic net

Elastic net generalizes lasso and ridge regression by adding both the l_1 and l_2 penalty to the least squares problem:

$$\text{minimize} \quad \frac{1}{2}\|Xw-b\|_2^2+\frac{1}{2}(1-\gamma)\|w\|_2^2+\gamma\|w\|_1 \tag{3.7}$$

Parameter $\gamma > 0$ interpolates between the l_1 and l_2 penalties. NysADMM applies with $\ell(Xw;b) = \frac{1}{2}\|Xw-b\|_2^2$, $r(w)=\frac{1}{2}(1-\gamma)\|w\|_2^2+\gamma\|w\|_1$. The Hessian matrices for ℓ is are X^TX.

3.3.2 Regularized logistic regression

Regularized logistic regression minimizes a logistic loss function together with an l_1-regularizer:

$$\text{minimize} \quad -\sum_i \left(b_i(Xw)_i-\log(1+\exp((Xw)_i))\right)+\gamma\|w\|_1 \tag{3.8}$$

NysADMM applies with $\ell(Aw;b) = -\sum_i (b_i(Aw)_i - \log(1+\exp((Aw)_i)))$ and $h(w) = \gamma\|w\|_1$. The inexact ADMM update chooses $\tilde{w}^{k+1}$ to minimize a quadratic approximation of the log-likelihood,

$$\text{minimize} \ \frac{1}{2}\sum_i \alpha_i^k (q_i^k - (Ax)_i)^2 + \frac{\rho}{2}\|x - \tilde{z}^k + \tilde{u}^k\|_2^2,$$

where α_i^k and q_i^k depend on the current estimate $\tilde{x}^k$ as

$$\begin{aligned} \alpha_i^k &= \frac{1}{2 + \exp(-(X\tilde{w}^k)_i) + \exp((X\tilde{w}^k)_i)} \\ q_i^k &= (X\tilde{w}^k)_i + \frac{b_i - \frac{1}{1+\exp(-(X\tilde{w}^k)_i)}}{\alpha_i^k}. \end{aligned}$$

Therefore, the solution of the x-subproblem can be approximated by solving the linear system

$$(X^T\text{diag}(\alpha^k)X + \rho I)w = \rho(\tilde{z}^k - \tilde{u}^k) + X^T\text{diag}(\alpha^k)q^k.$$

Here α^k and q^k are the vectors for α_i^k and q_i^k. The Hessian matrix of ℓ is given by $X^T\text{diag}(\alpha^k)X$.

3.3.3 Support vector machine

To reformulate the SVM problem for solution with NysADMM, consider the dual SVM problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}w^T\text{diag}(b)K\text{diag}(b)w - \mathbf{1}^T w \\ \text{subject to} & w^T b = 0 \\ & 0 \leq w \leq C. \end{array} \tag{3.9}$$

Variable w is the dual variable, b is the label or response, and C is the penalty parameter for misclassification. For linear SVM, $K = XX^T$ where X is the design matrix; and for nonlinear SVM, K is the corresponding kernel matrix. The SVM problem can be reformulated as (3.1) by setting $\ell(Kw;b) = \frac{1}{2}w^T\text{diag}(b)K\text{diag}(b)w - \mathbf{1}^T w$ and taking r to be the indicator function for convex constraint set $w^T b = 0,\ 0 \leq w \leq C$. The Hessian matrix for ℓ is $\text{diag}(b)K\text{diag}(b)$.

3.4 Convergence analysis

This section provides a convergence analysis for NysADMM. All proofs for the results in this section may be found in Section 3.7. First we show Nyström PCG can solve any quadratic problem in a constant number of iterations.

Theorem 3.4.1. *Let H be a symmetric positive semidefinite matrix, $\rho > 0$ and set $H_\rho = H + \rho I$. Suppose we construct the randomized Nyström preconditioner with sketch size $s \geq 8\left(\sqrt{d_{\text{eff}}^{\rho}} + \sqrt{8\log(\frac{16}{\delta})}\right)^2$. Then*

$$\kappa_2(P^{-1/2}H_\rho P^{-1/2}) \leq 8 \tag{3.10}$$

with probability at least $1-\delta$.

Theorem 3.4.1 strengthens Theorem 2.5.1 from Chapter 2, which provides a sharp expectation bound on the condition number of the preconditioned system, but gives loose high probability bounds based on Markov's inequality. Theorem 3.4.1 tightens this bound, showing that Nyström PCG enjoys an exponentially small failure probability.

As an immediate corollary, we can solve (3.4) with a few iterations of PCG using the Nyström preconditioner.

Corollary 3.4.2. *Instate the hypotheses of Theorem 3.4.1 and let* $\tilde{w}_\star$ *denote the solution of* (3.4). *Then with probability at least* $1-\delta$, *the iterates* $\{w_t\}_{t\geq 1}$ *produced by Nyström PCG on problem* (3.4) *satisfy*

$$\frac{\|w_t-\tilde{w}_\star\|_2}{\|\tilde{w}_\star\|_2} \leq \sqrt{\kappa(H_\rho)}\left(\frac{1}{2}\right)^{t-1}. \tag{3.11}$$

Thus, after $t \geq 1+\left\lceil \frac{\log\left(\frac{\sqrt{\kappa(H_\rho)}\|\tilde{w}_\star\|_2}{\epsilon}\right)}{\log(2)}\right\rceil$ *iterations,*

$$\|w_t-\tilde{w}_\star\|_2 \leq \epsilon. \tag{3.12}$$

Corollary 3.4.2 ensures that we can efficiently solve the sub-problem to the necessary accuracy at each iteration. This result allows us to prove convergence of NysADMM in the case of the quadratic loss.

Theorem 3.4.3. *Consider the problem in* (3.1) *with quadratic loss* $\ell(Xw;b)=\frac{1}{2}\|Xw-b\|_2^2$. *Define initial iterates* $\tilde{w}^0$, $\tilde{z}^0$ *and* $\tilde{u}^0 \in \mathbb{R}^d$, *stepsize* $\rho>0$, *and summable tolerance sequence* $\{\varepsilon^k\}_{k=0}^{\infty} \subset \mathbb{R}_+$. *Assume at* k*th ADMM iteration, the norm of the righthand side of the linear system* r^k *is bounded by constant* R *for all* k. *Construct the Nyström preconditioner with sketch size*

$$s \geq 8\left(\sqrt{d_{\text{eff}}^{\rho}}+\sqrt{8\log\left(\frac{16}{\delta}\right)}\right)^2$$

and solve problem (3.1) *with NysADMM, using* $T^k = 1+\left\lceil 2\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho R}}{\varepsilon^k\rho}\right)\right\rceil$. *iterations for PCG at the* k*th ADMM iteration. Then with probability at least* $1-\delta$,

1. *For all* $k \geq 0$, *each iterate* $\tilde{x}^{k+1}$ *satisfies*

$$\|\tilde{w}^{k+1}-w^{k+1}\|_2 \leq \varepsilon^k, \tag{3.13}$$

where w^{k+1} *is the exact solution of* (3.4).

2. *As* $k \to \infty$, $\{\tilde{w}^k\}_{k=0}^{\infty}$ *converges to a solution of the primal* (3.1) *and* $\{\rho\tilde{u}^k\}_{k=0}^{\infty}$ *converges to a solution of the dual problem of* (3.1).

Theorem 3.4.3 establishes convergence of NysADMM for a quadratic loss. The quadratic loss already covers many applications of interest including the lasso, elastic-net, and SVMs. Convergence for general convex losses is established in the follow-up work [59] by the current author along with collaborators. [59] goes beyond merely establishing convergence, it also proves explicit convergence rates for NysADMM and other approximate ADMM schemes under standard regularity assumptions.

The next result makes rigorous the claims made in Section 3.2.4: it shows we can determine whether or not we have reached the effective dimension by monitoring the empirical condition number $(\hat{\lambda}_s + \rho)/\rho$.

Theorem 3.4.4. *Suppose, for some user defined tolerance $\epsilon > 0$, the sketch size satisfies*

$$s \geq 8\left(\sqrt{d_{\text{eff}}^{\epsilon\rho/6}} + \sqrt{8\log\left(\frac{16}{\delta}\right)}\right)^2.$$

Then the empirical condition number of the Nyström preconditioned system $P^{-1/2}H_rP^{-1/2}$ satisfies

$$\frac{\hat{\lambda}_s + \rho}{\rho} \leq 1 + \frac{\epsilon}{42}. \tag{3.14}$$

Furthermore, with probability at least $1 - \delta$,

$$\left|\kappa_2(P^{-1/2}H_\rho P^{-1/2}) - \frac{\hat{\lambda}_s + \rho}{\rho}\right| \leq \epsilon. \tag{3.15}$$

Theorem 3.4.4 shows that once the empirical condition number is sufficiently close to 1, so too is the condition number of the preconditioned system. Hence it is possible to reach the effective dimension by doubling the sketch size of the Nyström approximation until the empirical condition number falls below the desired tolerance. Theorem 3.4.4 ensures the true condition number is close to this empirical estimate with high probability.

Theorem 3.4.4 also helps explain why sketch sizes much smaller than the effective dimension can succeed in practice. The point is best illustrated by instantiating an explicit parameter selection in Theorem 3.4.4, which yields the following corollary.

Corollary 3.4.5. *Instate the hypotheses of Theorem 3.4.4 with $\epsilon = 100$. Then with a sketch size of $s \gtrsim d_{\text{eff}}^{16\rho}$ the following holds*

1. $(\hat{\lambda}_s + \rho)/\rho \leq 1 + \frac{100}{42}$.

2. *With probability at least $1 - \delta$,*

$$\left|\kappa_2(P^{-1/2}H_\rho P^{-1/2}) - 1 - \frac{100}{42}\right| \leq 100.$$

Corollary 3.4.5 shows that for a coarse tolerance of $\epsilon = 100$, a sketch size of $s \gtrsim d_{\mathrm{eff}}^{16\rho}$ suffices to ensure that the condition number of $P^{-1/2}H_\rho P^{-1/2}$ is no more than around 100. Two practical observations cement the importance of this corollary. First, $d_{\mathrm{eff}}^{16\rho}$ is often significantly smaller than d_{eff}^{ρ}, possibly by an order of magnitude or more. Second, with a condition number around 100, PCG is likely to converge very quickly. In fact, for modest condition numbers, PCG is known to converge much faster in practice than the theory would suggest [160]. It is only when the condition number reaches around 10^3, that convergence starts to slow. Thus, Corollary 3.4.5 helps explain why it is not necessary for the sketch size to equal the effective dimension in order for NysADMM to obtain significant accelerations.

3.5 Numerical experiments

Table 3.2: **Statistics of experiment datasets.**

Name	instances n	features d	nonzero %
STL-10	13000	27648	96.3
CIFAR-10	60000	3073	99.7
CIFAR-10-rf	60000	60000	100.0
smallNorb-rf	24300	30000	100.0
E2006.train	16087	150348	0.8
sector	6412	55197	0.3
p53-rf	16592	20000	100.0
connect-4-rf	16087	30000	100.0
realsim-rf	72309	50000	100.0
rcv1-rf	20242	30000	100.0
cod-rna-rf	59535	60000	100.0

In this section, we evaluate the performance of NysADMM on different large-scale applications: lasso, ℓ_1-regularized logistic regression, and SVM. For each type of problems, we compare NysADMM with popular standard solvers. We run all experiments on a server with 128 Intel Xeon E7-4850 v4 2.10GHz CPU cores and 1056GB. We repeat every numerical experiment ten times and report the mean solution time. We highlight the best-performing method in bold. The tolerance of NysADMM at each iteration is chosen as the geometric mean $\varepsilon^{k+1} = \sqrt{r_p^k r_d^k}$ of the ADMM primal residual r_p and dual residual r_d at the previous iteration, as in [154]. See [23] for more motivation and details. An alternative is to choose the tolerance sequence as any decaying sequence with respect to the righthand side norm as the number of NysADMM iteration increases, e.g., $\varepsilon^k = \|r^k\|_2/k^\beta$, where β is a predefined factor. These two strategies perform similarly; our experiments use the first strategy.

We choose a sketch size $s = 50$ to compute the Nyström approximation throughout our experiments. Inspired by Theorem 3.4.4 and Corollary 3.4.5, even if the sketch size is much smaller than the effective dimension, NysADMM can still achieve significant acceleration in practice.

To support experiments with standard solvers, for each problem class we use the same stopping criterion and other parameter settings as the standard solver. These experiments use datasets with $n > 10,000$ or $d > 10,000$ from LIBSVM [29], UCI [45], and OpenML [165], with statistics summarized in Table 3.2. We use a random feature map [136,138] to generate features for the data sets CIFAR-10, smallnorb, realsim, rcv1, and cod-rna, which increases both predictive performance and problem dimension.

3.5.1 Lasso

This subsection demonstrates the performance of NysADMM to solve the standard lasso problem (3.2). Here we compare NysADMM with three standard lasso solvers, SSNAL [102], mfIPM [55], and glmnet [60]. SSNAL is a Newton method based solver; mfIPM is an interior point method based solver and glmnet is a coordinate descent based solver. In practice, these three solvers and NysADMM rely on different stopping criteria. In order to make a fair comparison, in our experiments, the accuracy of a solution w for (3.2) is measured by the following relative Karush–Kuhn–Tucker (KKT) residual [102]:

$$\eta = \frac{\|w - \mathrm{prox}_{\gamma\|\cdot\|_1}(w - X^T(Xw - b))\|}{1 + \|w\| + \|Xw - b\|}. \tag{3.16}$$

For a given tolerance ϵ, we stop the tested algorithms when $\eta < \epsilon$. Note that stopping criterion (3.16) is rather strong: if $\eta \leq 10^{-2}$ for NysADMM, then the primal and dual gaps for ADMM are $\lesssim 10^{-4}$, which suffices for most applications. Indeed, for many machine learning problems, lower bounds on the statistical performance of the estimator [104] imply an unavoidable level of statistical error that is greater than this optimization error for most applications. Optimizing the objective beyond the level of statistical error [1,105] does not improve generalization. For standard lasso experiments, we fix the regularization parameter at $\gamma = 1$.

Table 3.3: **Results for low precision lasso experiment.**

Task	Time for $\epsilon = 10^{-1}$ (s)			
	NysADMM	mfIPM	SSNAL	glmnet
STL-10	**165**	573	467	278
CIFAR-10-rf	**251**	655	692	391
smallNorb-rf	**219**	552	515	293
E2006.train	**313**	875	903	554
sector	**235**	678	608	396
realsim-rf	**193**	–	765	292
rcv1-rf	**226**	563	595	273
cod-rna-rf	**208**	976	865	324

Table 3.3 and Table 3.4 show results for lasso experiments. The average solution time for NysADMM, mfIPM, SSNAL, and glmnet with $\epsilon = 10^{-1}, 10^{-2}$ on different tasks are provided. Here

Table 3.4: **Results for high precision lasso experiment.**

Task	Time for $\epsilon = 10^{-2}$ (s)			
	NysADMM	mfIPM	SSNAL	glmnet
STL-10	**406**	812	656	831
CIFAR-10-rf	**715**	1317	1126	1169
smallNorb-rf	**596**	896	768	732
E2006.train	1657	1965	**1446**	2135
sector	957	1066	**875**	1124
realsim-rf	**732**	–	1035	922
rcv1-rf	**593**	853	715	736
cod-rna-rf	**715**	1409	1167	997

mfIPM fails to solve the realsim-rf instance since it requires $n < d$. For precision of $\epsilon = 10^{-1}$, NysADMM is faster than all other solvers and at least 3 times faster than both mfIPM and SSNAL. For precision of $\epsilon = 10^{-2}$, NysADMM is still faster than all other solvers for all instances except E2006.train and sector. The results are fair since both SSNAL and mfIPM are second-order solvers and can reach high precision. NysADMM and glmnet are first-order solvers; they reach low precision quickly, but improve accuracy more slowly than a second order method. In practice, for large-scale machine learning problems, a low precision solution usually suffices, as decreasing optimization error beyond the statistical noise in the problem does not improve generalization. Further, our algorithm achieves bigger improvements on dense datasets compared with sparse datasets, as the factors of the Nyström approximation are dense even for sparse problems. To further illustrate the results, we vary the value of ϵ from 1.0 to 10^{-3} on STL-10 task and plot the average solution time for four methods in Figure 3.1. We can see NysADMM is as least as fast as other solvers when $\epsilon > 10^{-3}$, and often

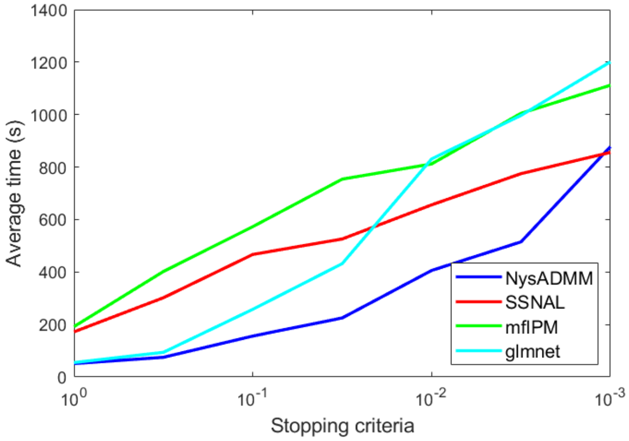


Figure 3.1: Solution times for varying tolerance ϵ on STL-10.

twice as fast for many practical values of ϵ.

3.5.2 l_1-Regularized logistic regression

This subsection demonstrates the performance of NysADMM on l_1-regularized logistic regression, (3.8) from Section 3.3.2. We test the method on binary classification problems using the same random feature map as in Section 3.5.1.

The l_1-regularized logistic regression experiments compare NysADMM with the SAGA algorithm, a stochastic average gradient like algorithm [38] implemented in sklearn, and the accelerated proximal gradient (APG) algorithm [16, 122, 126]. For the purpose of fair comparison, all the algorithms are stopped when the maximum relative change in the problem variable (that is, the regression coefficients) $\frac{\|w_k - w_{k+1}\|_\infty}{\|w_k\|_\infty}$ is less than the tolerance. The tolerance is set to 10^{-3}; other settings match the default settings of the sklearn logistic regression solver.

An overview of l_1-regularized logistic regression experiment results are provided in Table 3.5. NysADMM uniformly out performs SAGA, solving each problem at least twice as fast. Similarly, NysADMM is at least twice as fast as APG on all datasets except STL-10, where it performs comparably. In the cases of p53-rf and connect-4-rf, NysADMM runs significantly faster than its competitors, being four times faster than SAGA and three times faster than APG. These large performance gains are due to the size of the problem instances and their conditioning. From [38], the convergence speed of SAGA depends on the problem instance size and condition number. Our test cases have large instance sizes and condition numbers, which lead to slow convergence of SAGA. The situation with APG is similar. Indeed, although ADMM and proximal gradient methods generally have the same $\mathcal{O}(1/t)$-convergence rate [16, 78], NysADMM is less sensitive ill-conditioning than APG.

Table 3.5: **Results for l_1-regularized logistic regression experiment.**

Task	NysADMM (s)	SAGA (s)	APG (s)
STL-10	3012	6083	**2635**
CIFAR-10-rf	**7884**	21256	17292
p53-rf	**528**	2116	1880
connect-4-rf	**866**	4781	7365
smallnorb-rf	**1808**	6381	4408
rcv1-rf	**1237**	3988	2759
con-rna-rf	**7528**	21513	16361

3.5.3 Support vector machine

This subsection demonstrates the performance of NysADMM on kernel SVM problem for binary classification, (3.9) from Section 3.3.3. The SVM experiments compare NysADMM with the LIBSVM solver [29]. LIBSVM uses sequential minimal optimization (SMO) to solve the dual SVM problem. We use the same stopping criteria as the LIBSVM solver, which stops the NysADMM method when the ADMM dual gap reaches 10^{-4} level. All SVM experiments use the RBF kernel. Table 3.6 shows the results of SVM experiments. On these problems, NysADMM is at least 3 times faster (and up to 58 times faster) than the LIBSVM solver. Consider problem formulation (3.9), with the RBF kernel. The Gram matrix $\mathrm{diag}(b)K\mathrm{diag}(b)$ is dense and approximately low rank: exactly the setting in which NysADMM should be expected to perform well. In constrast, the SMO-type decomposition in LIBSVM solver works better for sparse problems, as it updates only two variables at each iteration.

Table 3.6: **Results of SVM experiment.**

Task	NysADMM time (s)	LIBSVM time (s)
STL-10	**208**	11573
CIFAR-10	**1636**	8563
p53-rf	**291**	919
connect-4-rf	**7073**	42762
realsim-rf	**17045**	52397
rcv1-rf	**564**	32848
cod-rna-rf	**4942**	36791

3.6 Conclusion

In this paper, we have developed a scalable new algorithm, NysADMM, that combines inexact ADMM and the randomized low-rank Nyström approximation to accelerate composite convex optimization. We show that NysADMM exhibits strong benefits both in theory and in practice. Our theory shows that when the Nyström preconditioner is constructed with an appropriate rank, NysADMM requires only a constant number of matvecs to solve the ADMM subproblem. We have also provided an adaptive strategy for selecting the rank that possesses a similar computational profile to the non-adaptive algorithm, and allows us to realize the theoretical benefits in practice. Further, numerical results demonstrate that NysADMM is as least twice as fast as standard methods on large dense lasso, regularized logistic regression, and kernalized SVM problems. More broadly, this paper shows the promise of recent advances in RandNLA to provide practical accelerations for important large-scale optimization algorithms.

3.7 Proofs not appearing in the main chapter

In this section we give the proofs for the main results of the paper: Theorem 3.4.1, Theorem 3.4.3, and Theorem 3.4.4.

3.7.1 Preliminaries

We start by recalling some useful background information and technical results that are useful for proving the main theorems. In order to obtain the exponentially small failure probabilities in Theorem 3.4.1 and Theorem 3.4.4 we take a different approach from the one in Chapter 2, which was based off of Markov's inequality. The proofs here are based on regularized Schur complements and approximate matrix multiplication. Our arguments are inspired by the techniques used to establish statistical guarantees for approximate kernel ridge regression via column sampling schemes [3, 13].

Regularized Nyström approximation: Properties

We start by recalling some important properties of the Nyström approximation and its regularized variant. Recall that $\Omega \in \mathbb{R}^{d\times s}$ denotes the test matrix from which we construct the Nyström approximation. Given $\sigma > 0$, the regularized Nyström approximation with respect to Ω is defined as

$$H\langle\Omega\rangle_\sigma = (H\Omega)(\Omega^T H\Omega + \sigma I)^{-1}(H\Omega)^T. \tag{3.17}$$

Furthermore, let $H = V\Lambda V^T$ be the eigendecomposition of H and define $D_\sigma = H(H+\sigma I)^{-1} = \Lambda(\Lambda+\sigma I)^{-1}$. We shall see below that D_σ plays a crucial role in the analysis. The following lemma summarizes the properties of the regularized Nyström approximation.

Lemma 3.7.1 (Lemma 3.7.2 [3]). *Let H be a symmetric psd matrix, $\sigma > 0$. Define $E = H - H\langle\Omega\rangle$ and $E_\sigma = H - H\langle\Omega\rangle_\sigma$. Then the following hold.*

1. $H\langle\Omega\rangle_\sigma \preceq H\langle\Omega\rangle \preceq H$.

2. $0 \preceq E \preceq E_\sigma$.

3. *If $\|D_\sigma^{1/2}V^T(\frac{1}{s}\Omega\Omega^T)VD_\sigma^{1/2} - D_\sigma\| \leq \eta < 1$, then*

$$0 \preceq E_\sigma \preceq \frac{\sigma}{1-\eta}I. \tag{3.18}$$

Lemma 3.7.1 relates $H\langle\Omega\rangle_\sigma$ to $H\langle\Omega\rangle$ and H. In particular, item 2 implies that $\|E\| \leq \|E_\sigma\|$, so controlling E_σ controls E. Item 3 shows that E_σ can be controlled by the spectral norm of the matrix

$$D_\sigma^{1/2}V^T\frac{1}{s}\Omega\Omega^T VD_\sigma^{1/2} - D_\sigma. \tag{3.19}$$

The spectral norm of (3.19) can be bounded by observing

$$\mathbb{E}\left[D_\sigma^{1/2}V^T\frac{1}{s}\Omega\Omega^T VD_\sigma^{1/2}\right] = \tag{3.20}$$

$$D_\sigma^{1/2}V^T\mathbb{E}\left[\frac{1}{s}\Omega\Omega^T\right]VD_\sigma^{1/2} = \tag{3.21}$$

$$D_\sigma^{1/2}V^TVD_\sigma^{1/2} = D_\sigma. \tag{3.22}$$

Thus, $D_\sigma^{1/2}V^T\frac{1}{s}\Omega\Omega^TVD_\sigma^{1/2}$ is an unbiased estimator of D_σ, and may be viewed as approximating the product of the matrices $D_\sigma^{1/2}V^T$ and $VD_\sigma^{1/2}$. Hence results from randomized linear algebra can bound the spectral norm of this difference. In particular, it suffices to take a sketch size that scales with the effective dimension, using results on approximate matrix multiplication in terms of stable rank [36].

Approximate matrix multiplication in terms of the effective dimension

The condition in item 3 of Lemma 3.7.1 follows immediately from theorem 1 of [36]. Unfortunately, the analysis in that paper does not yield explicit constants. Instead we use a special case of their results due to [96] that provides explicit constants. Theorem 3.7.2 simplifies Theorem 5.2 in [96].

Theorem 3.7.2 (Simplified Theorem 5.2 [96])**.** *Let $\Psi \in \mathbb{R}^{s\times d}$ be a matrix with i.i.d. $N(0,\frac{1}{s})$ entries. Given $\delta > 0$, and $\tau \in (0,1)$ it holds with probability at least $1-\delta$ that*

$$\sup_{v\in\mathbb{S}^{d-1}}\langle v,(D_\sigma^{1/2}V^T\Psi^T\Psi VD_\sigma^{1/2}-D_\sigma)v\rangle \leq \tau + 2\sqrt{\tau}, \tag{3.23}$$

$$\inf_{v\in\mathbb{S}^{d-1}}\langle v,(D_\sigma^{1/2}V^T\Psi^T\Psi VD_\sigma^{1/2}-D_\sigma)v\rangle \geq \tau - 2\sqrt{\tau}, \tag{3.24}$$

provided $s \geq \frac{\left(\sqrt{d_{\text{eff}}^\sigma}+\sqrt{8\log(16/\delta)}\right)^2}{\tau}$.

Setting $\Psi = \frac{1}{\sqrt{s}}\Omega^T$, where $\Omega \in \mathbb{R}^{d\times s}$ has i.i.d. $N(0,1)$ entries, Theorem 3.7.2 yields the following corollary.

Corollary 3.7.3. *Let $\Omega \in \mathbb{R}^{d\times s}$ be a matrix with i.i.d. $N(0,1)$ entries. Given $\delta > 0$, and $\tau \in (0,1)$ it holds with probability at least $1-\delta$ that*

$$\left\|D_\sigma^{1/2}V^T\frac{1}{s}\Omega\Omega^TVD_\sigma^{1/2}-D_\sigma\right\| \leq \tau + 2\sqrt{\tau} \tag{3.25}$$

provided $s \geq \frac{\left(\sqrt{d_{\text{eff}}^\sigma}+\sqrt{8\log(16/\delta)}\right)^2}{\tau}$.

3.7.2 Proofs of Theorem 3.4.1 and Corollary 3.4.2

We start by establishing the following lemma, from which Theorem 3.4.1 follows easily.

Lemma 3.7.4. *Let $\epsilon > 0$ and $E = H - H\langle\Omega\rangle$. Suppose we construct a randomized Nyström approximation from a standard Gaussian random matrix Ω with sketch size $s \geq 8\left(\sqrt{d_{\text{eff}}^{\epsilon}} + \sqrt{8\log(\frac{16}{\delta})}\right)^2$. Then the event*

$$\mathcal{E} = \{\|E\| \leq 6\epsilon\}, \tag{3.26}$$

holds with probability at least $1-\delta$.

Proof. Let $\Omega_s = \frac{1}{\sqrt{s}}\Omega$ and observe that $H\langle\Omega_s\rangle = H\langle\Omega\rangle$. Now the conditions of Corollary 3.7.3 are satisfied with $\sigma = \epsilon$ and $\tau = 1/8$. Consequently with probability at least $1-\delta$,

$$\left\|D_{\epsilon}^{1/2}V^T\frac{1}{s}\Omega\Omega^TVD_{\epsilon}^{1/2} - D_{\epsilon}\right\| \leq \frac{1}{8} + \frac{\sqrt{2}}{2}.$$

Hence applying Lemma 3.7.1 with $\sigma = \epsilon$ and $\eta = \frac{1}{8} + \frac{\sqrt{2}}{2}$, we obtain

$$\left\|H - H\langle\Omega_s\rangle_{\epsilon}\right\| \leq 6\epsilon,$$

with probability at least $1-\delta$. Recalling our initial observation, we conclude the desired result. □

Proof of Theorem 3.4.1

Proof. As $s \geq 8\left(\sqrt{d_{\text{eff}}^{\rho}} + \sqrt{8\log(\frac{16}{\delta})}\right)^2$ we have that $\|E\| \leq 6\rho$ with probability at least $1-\delta$ by Lemma 3.7.4. Furthermore, $\hat{\lambda}_s \leq \frac{\rho}{7}$ by item 3 of Lemma 2.2.1 and Lemma 2.5.4 with $\gamma = 1/7$. Combining this with Proposition 2.5.3, we conclude with probability at least $1-\delta$,

$$\begin{aligned}\kappa_2(P^{-1/2}H_{\rho}P^{-1/2}) &\leq \frac{\hat{\lambda}_s + \rho + \|E\|}{\rho}\\ &\leq 1 + 6 + \frac{1}{7} \leq 8\end{aligned}$$

as desired. □

Proof of Corollary 3.4.2

Proof. Let $A = P^{-1/2}H_{\rho}P^{-1/2}$ and condition on the event that $\kappa_2(A) \leq 8$, which holds with probability at least $1-\delta$. Then the same argument used in Section 2.8.7 used to establish Corollary 2.5.2 guarantees after t iterations that,

$$\frac{\|w_t - \tilde{w}_{\star}\|_{H_{\rho}}}{\|\tilde{w}_{\star}\|_{H_{\rho}}} \leq 2\left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}\right)^t \tag{3.27}$$

Theorem 3.4.1 guarantees that the Nyström preconditioned matrix satisfies $\kappa_2(A) \leq 8$, so the above display may be majorized as

$$\frac{\|w_t - \tilde{w}_\star\|_{H_\rho}}{\|\tilde{w}_\star\|_{H_\rho}} \leq \left(\frac{1}{2}\right)^{t-1}. \tag{3.28}$$

Now, from the elementary inequality

$$\sqrt{\lambda_d(H_\rho)}\|w\|_2 \leq \|w\|_{H_\rho} \leq \sqrt{\lambda_1(H_\rho)}\|w\|_{H_\rho}, \tag{3.29}$$

we conclude

$$\frac{\|w_t - \tilde{w}_\star\|_2}{\|\tilde{w}_\star\|_2} \leq \sqrt{\kappa_2(H_\rho)}\left(\frac{1}{2}\right)^{t-1}. \tag{3.30}$$

The claimed result now follows from an elementary computation. □

3.7.3 Proof of Theorem 3.4.3

This proof is a natural consequence of the following theorem from [48].

Theorem 3.7.5. *Consider a convex optimization problem in the primal form (P),*

$$\text{minimize } f(w) + r(Mw),$$

where $w \in \mathbb{R}^d$, $M \in \mathbb{R}^{m\times d}$ has full column rank. Pick any y^0, $z^0 \in \mathbb{R}^m$, and $\rho > 0$, and summable sequences

$$\{\varepsilon^k\}_{k=0}^{\infty} \subseteq [0,\infty),\ \sum_{k=0}^{\infty}\varepsilon^k < \infty,$$
$$\{\nu^k\}_{k=0}^{\infty} \subseteq [0,\infty),\ \sum_{k=0}^{\infty}\nu^k < \infty,$$
$$\{\lambda^k\}_{k=0}^{\infty} \subseteq (0,2),\ 0 < \inf \lambda^k \leq \sup \lambda^k < 2.$$

The dual problem (D) of primal problem (P) is

$$\text{maximize}_{y\in\mathbb{R}^m}\ -(f^*(-M^T y) + g^*(y)).$$

Suppose the primal and dual ADMM iterates $\{w^k\}_{k=0}^{\infty}$, $\{z^k\}_{k=0}^{\infty}$, and $\{y^k\}_{k=0}^{\infty}$ satisfy the update *equations to within errors given by conform, for all k to*

$$\begin{aligned} &\Big\| w^{k+1} - \mathrm{argmin}_w \{ f(w) + \langle y^k, Mw \rangle \\ &+ \frac{\rho}{2} \| Mw - z^k \|_2^2 \} \Big\|_2 \le \varepsilon^k, \\ &\Big\| z^{k+1} - \mathrm{argmin}_z \{ r(z) - \langle y^k, z \rangle \\ &+ \frac{\rho}{2} \| \lambda^k M w^{k+1} - z + (1-\lambda^k) z^k \|_2^2 \} \Big\|_2 \le \nu^k, \\ &y^{k+1} = y^k + \rho(\lambda^k M w^{k+1} + (1-\lambda^k) z^k - z^{k+1}). \end{aligned} \tag{3.31}$$

Then if (P) has a Kuhn-Tucker pair, $\{w^k\}$ converges to a solution of (P) and $\{y^k\}$ converges to a solution of (D).

Proof of Theorem 3.4.3

Proof. Consider optimization problem (3.1) and the associated NysADMM algorithm Algorithm 8. Suppose $\{\tilde{w}^k\}_{k=0}^{\infty}$, $\{\tilde{z}^k\}_{k=0}^{\infty}$, and $\{\tilde{u}^k\}_{k=0}^{\infty}$ are generated by NysADMM iterations. Since $\ell(Xw, b)$ is quadratic with respect to w, the x-subproblem of (3.1) is exactly the linear system (3.4).

Let w^{k+1} be the exact solution for the w-subproblem at iteration k. For all $k \ge 0$, NysADMM iterate $\tilde{w}^{k+1}$ satisfies $\|\tilde{w}^{k+1} - w^{k+1}\|_2 \le \varepsilon^k$. Let $M = I$, $\nu^k = 0$, $\lambda^k = 1$, $y^k = \rho\tilde{u}^k$ for all k, and $f(w) = \ell(Xw; b)$. By Theorem 3.7.5, $\{\tilde{w}^k\}_{k=0}^{\infty}$, $\{\tilde{z}^k\}_{k=0}^{\infty}$, and $\{\rho\tilde{u}^k\}_{k=0}^{\infty}$ satisfy condition (3.31). Therefore, if the optimization problem (3.1) has a Kuhn-Tucker pair, $\{\tilde{w}^k\}$ converges to a solution of (3.1) and $\{\rho\tilde{u}^k\}$ converges to a solution of the dual problem of (3.1).

Next, we derive the bound for the number of Nyström PCG iterations T^k required at NysADMM iteration k. As the Hessian of $\ell(Xw; b)$ is constant, we only need to compute the preconditioner for constant linear system matrix $H_\rho = X^T X + \rho I$. The resulting preconditioner can then be used for all NysADMM iterations. Since the Nyström preconditioner is constructed with sketch size $s \ge 8\left(\sqrt{d_{\mathrm{eff}}^{\rho}} + \sqrt{8\log(\frac{16}{\delta})}\right)^2$, by Corollary 3.4.2, with probability at least $1-\delta$, after

$$T^k \ge 1 + \left\lceil \frac{\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}\|\tilde{w}^{k+1}\|_2}{\varepsilon^k}\right)}{\log(2)} \right\rceil$$

Nyström PCG iterations, we have $\|\tilde{w}^{k+1} - w^{k+1}\|_2 \le \varepsilon^k$. Recall the right-hand side of linear system (3.4) r^k. The exact solution for the x-subproblem w^{k+1} at iteration k satisfies $\|w^{k+1}\|_2 \le \frac{\|r^k\|_2}{\rho}$. We have

$$\left\lceil \frac{\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}\|\tilde{w}^{k+1}\|_2}{\varepsilon^k}\right)}{\log(2)} \right\rceil \le \left\lceil \frac{\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}\|r^k\|_2}{\varepsilon^k \rho}\right)}{\log(2)} \right\rceil.$$

Further, by assumption, as $\|r^k\|_2$ is bounded by a constant R for all k, we have

$$\left\lceil \frac{\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}}{\varepsilon^k \rho}\right)}{\log(2)} \right\rceil \leq \left\lceil \frac{\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}R}{\varepsilon^k \rho}\right)}{\log(2)} \right\rceil \leq \left\lceil 2\log\left(\frac{\sqrt{\lambda_1(H_\rho)/\rho}R}{\varepsilon^k \rho}\right) \right\rceil.$$

This gives the bound for the number of PCG iterations T^k required at NysADMM iteration k □

3.7.4 Proof of Theorem 3.4.4

Proof. By hypothesis we have $s > d_{\text{eff}}^{\epsilon\rho/6}$, so Lemma 2.5.4 with $\gamma = 7$ yields

$$\hat{\lambda}_s \leq \lambda_s \leq \frac{1}{7}\frac{\epsilon\rho}{6} = \frac{\epsilon\rho}{42},$$

Thus,

$$\frac{\hat{\lambda}_s + \rho}{\rho} \leq 1 + \frac{\epsilon}{42}.$$

This gives the first statement. For the second statement we use our hypothesis on s to apply Lemma 3.7.4 with tolerance $\epsilon\rho/6$. From this we conclude $\|E\| \leq \epsilon\rho$ with probability at least $1 - \delta$. Combining this with Proposition 2.5.3 yields

$$\kappa_2(P^{-1/2}H_\rho P^{-1/2}) - \frac{\hat{\lambda}_s + \rho}{\rho} \leq \epsilon,$$

with probability at least $1 - \delta$. On the other hand, condition numbers always satisfy

$$\kappa_2(P^{-1/2}H_\rho P^{-1/2}) \geq 1.$$

Combining this with our upper bound on $\hat{\lambda}_s$ gives

$$\begin{aligned} \kappa_2(P^{-1/2}H_\rho P^{-1/2}) - \frac{\hat{\lambda}_s + \rho}{\rho} &\geq 1 - (1 + \epsilon/42) \\ &= -\epsilon/42. \end{aligned}$$

Hence with probability at least $1 - \delta$

$$\left| \kappa_2(P^{-1/2}H_\rho P^{-1/2}) - \frac{\hat{\lambda}_s + \rho}{\rho} \right| \leq \epsilon.$$

□

3.8 AdaNysADMM Algorithm

In this section we give the pseudocode for AdaNysADMM

Algorithm 9 AdaNysADMM

```
Input: design matrix $X$, response $b$, loss function $\ell$, regularization $r$, stepsize $\rho$, positive summable sequence $\{\varepsilon^k\}_{k=0}^{\infty}$
$[U, \hat{\Lambda}] = $ AdaptiveRandNysAppx$(X^T \nabla^2 \ell(A\tilde{w}^k; b)X, s)$ {use Algorithm 5}
repeat
  find $\tilde{w}^{k+1}$ that solves (3.4) within tolerance $\varepsilon^k$ by Nyström PCG
  $\tilde{z}^{k+1} = \text{argmin}_z\{r(z) + \frac{\rho}{2}\|\tilde{w}^{k+1} - z + \tilde{u}^k\|_2^2\}$
  $\tilde{u}^{k+1} = \tilde{u}^k + \tilde{w}^{k+1} - \tilde{z}^{k+1}$
until convergence
```

Chapter 4

SketchySGD

4.1 Introduction

Modern large-scale machine learning requires stochastic optimization: evaluating the full objective or gradient even once is too slow to be useful. Instead, stochastic gradient descent (SGD) and variants are the methods of choice [4,37,85,116,142]. However, stochastic optimizers sacrifice stability for their improved speed. Parameters like the learning rate are difficult to choose and important to get right, with slow convergence or divergence looming on either side of the best parameter choice [120]. Worse, most large-scale machine learning problems are ill-conditioned: typical condition numbers among standard test datasets range from 10^4 to 10^8 (see Table 4.2) or even larger, resulting in painfully slow convergence even given the optimal learning rate. This paper introduces a method, SketchySGD, that uses a principled theory to address ill-conditioning and yields a theoretically motivated learning rate that robustly works for modern machine learning problems. Figure 4.1 depicts the performance of stochastic optimizers using learning rates tuned for each optimizer on a ridge regression problem with the E2006-tfidf dataset (see Section 4.5). SketchySGD improves the objective substantially, while the other stochastic optimization methods (SGD, SVRG, SAGA, and Katyusha) do not.

Second-order optimizers based on the Hessian, such as Newton's method and quasi-Newton methods, are the classic remedy for ill-conditioning. These methods converge at super-linear rates under mild assumptions and are faster than first-order methods both in theory and in practice [24,124]. Alas, it has proved difficult to design second-order methods that can use stochastic gradients. This deficiency limits their use in large-scale machine learning. Stochastic second-order methods using stochastic Hessian approximations but full gradients are abundant in the literature [97, 131, 159]. However, a practical second-order stochastic optimizer must replace both the Hessian and gradient by stochastic approximations. While many interesting ideas have been proposed, existing methods require impractical conditions for convergence: for example, a batch size for the gradient and Hessian that grows with the condition number [145], or that increases geometrically at each iteration [20].

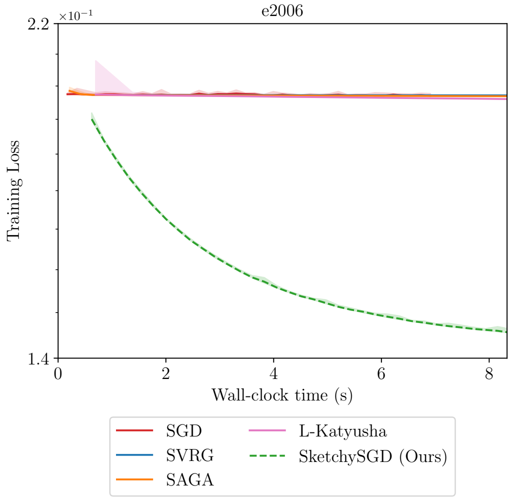


Figure 4.1: SketchySGD outperforms standard stochastic gradient optimizers, even when their parameters are tuned for optimal performance. Each optimizer was allowed 40 full data passes.

These theoretical conditions are impossible to implement in practice. Convergence results without extremely large or growing batch sizes have been established under interpolation, i.e., if the loss is zero at the solution [112]. This setting is interesting for deep learning, but is unrealistic for convex machine learning models. Moreover, most of these methods lack practical guidelines for hyperparameter selection, making them difficult to deploy in real-world machine learning pipelines. Altogether, the literature does not present any stochastic second-order method that can function as a drop-in replacement for SGD, despite strong empirical evidence that — for perfectly chosen parameters — they yield improved performance. One major contribution of this paper is a theory that matches how these methods are used in practice, and therefore is able to offer practical parameter selection rules that make SketchySGD (and even some previously proposed methods) practical. We provide a more detailed discussion of how SketchySGD compares to prior stochastic second-order optimizers in Section 4.3.

SketchySGD accesses second-order information using only minibatch Hessian-vector products to form a sketch, and produces a preconditioner for SGD using the sketch, which is updated only rarely (every epoch or two). This primitive is compatible with standard practices of modern large-scale machine learning, as it can be computed by automatic differentiation. Our major contribution is a tighter theory that enables practical choices of parameters that makes SketchySGD a drop-in replacement for SGD and variants that works out-of-the-box, without tuning, across a wide variety of problem instances.

How? A standard theoretical argument in convex optimization shows that the learning rate in a gradient method should be set as the inverse of the smoothness parameter of the objective to guarantee convergence [24, 123]. This choice generally results in a tiny stepsize and very slow

Algorithm 10 SketchySGD (Practical version)

```
Input: initialization $w_0$, hvp oracle $\mathcal{O}_H$, ranks $\{r_j\}$, regularization $\rho$, preconditioner update frequency $u$, stochastic gradient batch size $b_g$, stochastic Hessian batch sizes $\{b_{h_j}\}$
for $k = 0, 1, \ldots, m-1$ do
  Sample a batch $B_k$
  Compute stochastic gradient $g_{B_k}(w_k)$
  if $k \equiv 0 \pmod{u}$ then {Update preconditioner}
    Set $j = j + 1$
    Sample a batch $S_j$                                  $\{|S_j| = b_{h_j}\}$
    $\Phi = \text{randn}(p, r_j)$                          {Gaussian test matrix}
    $Q = \texttt{qr\_econ}\ (\Phi)$
    Compute sketch $Y = H_{S_j}(w_k)Q$                     $\{r$ calls to $\mathcal{O}_{H_{S_j}}\}$
    $[\hat{V}, \hat{\Lambda}] = \texttt{RandNysApprox}(Y, Q, r_j)$
    $\eta_j = \texttt{get\_learning\_rate}(\mathcal{O}_{H_{S_j}}, \hat{V}, \hat{\Lambda}, \rho)$
  end if
  Compute $v_k = (\hat{H}_{S_j} + \rho I)^{-1} g_{B_k}(w_k)$ via (4.5)
  $w_{k+1} = w_k - \eta_j v_k$                            {Update parameters}
end for
```

convergence. However, in the context of SketchySGD, the preconditioned smoothness constant is generally around 1, and so its inverse provides a reasonable learning rate. Moreover, it is easy to estimate, again using minibatch Hessian-vector products to measure the largest eigenvalue of a preconditioned minibatch Hessian.

Theoretically, we establish SketchySGD converges to a small ball around the minimum on for both smooth convex functions and smooth and strongly convex functions, which suffices for good test error [1, 77, 104]. By appealing to the modern theory of SGD for finite-sum optimization [72] in our analysis, we avoid vacuous or increasing batchsize requirements for the gradient. As a corollary of our theory, we obtain that SketchySGD converges linearly to the optimum whenever the model interpolates the data, recovering the result of [112]. In addition, when the objective is quadratic, we show that the number of iterations SketchySGD requires to reach an ϵ-suboptimal solution improves upon that of SGD.

Numerical experiments verify that SketchySGD yields comparable or superior performance to SGD, SAGA, SVRG, stochastic L-BFGS [115], and loopless Katyusha [93] equipped with tuned hyperparameters that attain their best performance. Experiments also demonstrate that SketchySGD's default hyperparameters, including the rank of the preconditioner and the frequency at which it is updated, work well across a wide range of datasets.

4.1.1 SketchySGD

SketchySGD finds $w \in \mathbb{R}^p$ to minimize the (possibly non-convex) empirical risk

$$\text{minimize} \quad f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w), \tag{4.1}$$

given access to a gradient oracle for each f_i. SketchySGD is formally presented as Algorithm 12. SketchySGD tracks two different sets of indices: k, which counts the number of total iterations, and j, which counts the number of (less frequent) preconditioner updates. SketchySGD updates the preconditioner (every u iterations) by sampling a minibatch S_j and forming a low-rank $\hat{H}_{S_j}$ using Hessian vector products with the minibatch Hessian H_{S_j} evaluated at the current iterate w_k. Given the Hessian approximation, it uses $\hat{H}_{S_j} + \rho I$ as a preconditioner, where $\rho > 0$ is a regularization parameter. Subsequent iterates are then computed as

$$w_{k+1} = w_k - \eta_j (\hat{H}_{S_j} + \rho I)^{-1} g_{B_k}(w_k), \tag{4.2}$$

where $g_{B_k}(w_k)$ is the minibatch stochastic gradient and η_j is the learning rate, which is automatically determined by the algorithm.

The SketchySGD update may be interpreted as a preconditioned stochastic gradient step with Levenberg-Marquardt regularization [100, 107]. Indeed, let $P_j = \hat{H}_{S_j} + \rho I$ and define the preconditioned function $f_{P_j}(z) = f(P_j^{-1/2} z)$, which represents f as a function of the preconditioned variable $z \in \mathbb{R}^p$. Then (4.2) is equivalent to [1]

$$z_{k+1} = z_k - \eta_j \hat{g}_{P_j}(z_k), \quad w_{k+1} = P_j^{-1/2} z_k,$$

where $\hat{g}_{P_j}(z_k)$ is the minibatch stochastic gradient as a function of z. Thus, SketchySGD first takes a step of SGD in preconditioned space and then maps back to the original space. As preconditioning induces more favorable geometry, SketchySGD chooses better search directions and uses a stepsize adapted to the (more isotropic) preconditioned curvature. Hence SketchySGD makes more progress than SGD to ultimately converge faster.

Contributions

1. We develop a new stochastic second-order method that is fast and generalizes well by accessing only a subsampled Hessian and stochastic gradient.

2. We devise an heuristic (but well-motivated) automated learning rate for this algorithm that works well in both ridge and logistic regression. More broadly, we present default settings for all hyperparameters of SketchySGD, which allow it to work out-of-the-box.

[1] See Section 4.8.1 for a proof.

3. We show that SketchySGD with a fixed learning rate converges to a small ball about the minimum for smooth and convex, and smooth and strongly convex objectives. Additionally, we show SketchySGD converges at a faster rate than SGD for ill-conditioned least-squares problems. We verify this improved convergence in numerical experiments.

4. We present experiments showing that SketchySGD with its default hyperparameters can match or outperform popular stochastic first- and second-order methods and randomized least-squares solvers on ridge and logistic regression problems.

5. We present proof-of-concept experiments on tabular deep learning, which show SketchySGD scales well and is competitive with other stochastic second-order optimizers in deep learning, potentially providing an avenue for interesting future research.

4.1.2 Roadmap

Section 4.2 describes the SketchySGD algorithm in detail, explaining how to compute $\hat{H}_{S_k}$ and the update in (4.2) efficiently. Section 4.3 surveys previous work on stochastic second-order methods, particularly in the context of machine learning. Section 4.4 establishes convergence of SketchySGD in convex machine learning problems. Section 4.5 provides numerical experiments showing the superiority of SketchySGD relative to competing optimizers.

4.1.3 Notation

Throughout the paper B_k and S_j denote subsets of $\{1, \ldots, n\}$ that are sampled independently and uniformly without replacement. The corresponding stochastic gradient and minibatch Hessian are given by

$$g_{B_k}(w) = \frac{1}{b_{g_k}} \sum_{i \in B_k} g_i(w), \quad H_{S_j}(w) = \frac{1}{b_{h_j}} \sum_{i \in S_j} \nabla^2 f_i(w),$$

where $b_{g_k} = |B_k|$, $b_{h_j} = |S_j|$. For shorthand, we often omit the dependence upon w and simply write g_{B_k} and H_{S_j}. We also define $H(w)$ as the Hessian of the objective f at w. Given $w \in \mathbb{R}^p$, we define $M(w) = \max_{1 \leq i \leq n} \|\nabla^2 f_i(w)\|$ and $G(w) = \max_{1 \leq i \leq n} \|\nabla f_i(w)\|$. Given any $\beta > 0$, we use the notation $H_{S_j}^{\beta}$ to denote $H_{S_j} + \beta I$. We abbreviate positive-semidefinite as psd, and positive-definite as pd. We say f is L-smooth if $H(w) \preceq LI$ for all $w \in \mathbb{R}^p$. We say f is μ-strongly convex if $\mu I \preceq H(w)$ for all $w \in \mathbb{R}^p$. For L-smooth and μ-strongly convex f, we define the condition number $\kappa := L/\mu$. We denote the Loewner order on the convex cone of psd matrices by $\preceq$, where $A \preceq B$ means $B - A$ is psd. Given a psd matrix $A \in \mathbb{R}^{p \times p}$, we enumerate its eigenvalues in descending order, $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_p(A)$. Throughout the paper $\|\cdot\|$ stands for the usual 2-norm for vectors and operator norm for matrices. For any matrix $A \in \mathbb{R}^{p \times p}$ and $u, v \in \mathbb{R}^p$, $\|v\|_A^2 := v^T A v$, and $\langle u, v\rangle_A = u^T A v$. Finally given a psd matrix A and $\beta > 0$ we define the effective dimension by $d^{\beta}_{\text{eff}}(A) = \operatorname{tr}(A(A+\beta I)^{-1})$, which provides a smoothed measure of the eigenvalues greater than or equal to β.

4.2 SketchySGD: efficient implementation and hyperparameter selection

We now formally describe SketchySGD (Algorithm 12) and its efficient implementation.

Hessian vector product oracle SketchySGD relies on one main computational primitive, a (minibatch) Hessian vector product (hvp) oracle, to compute a low-rank approximation of the (minibatch) Hessian. Access to such an oracle naturally arises in machine learning problems. In the case of generalized linear models (GLMs), the Hessian is given by $H(w) = \frac{1}{n}A^T D(w) A$, where $A \in \mathbb{R}^{n\times p}$ is the data matrix and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix. Accordingly, an hvp between $H_{S_j}(w)$ and $v \in \mathbb{R}^p$ is given by

$$H_{S_j}(w)v = \frac{1}{b_{h_j}} \sum_{i\in S_j} d_i(w) a_i \left(a_i^T v\right).$$

For more complicated losses, an hvp can be computed by automatic differentiation (AD) [129]. The general cost of r hvps with $H_{S_j}(w)$ is $O(b_{h_j} pr)$. In contrast, explicitly instantiating a Hessian entails a heavy $O(p^2)$ storage and $O(np^2)$ computational cost. Further computational gains can be made when the subsampled Hessian enjoys more structure, such as sparsity. If $H_{S_j}(w)$ has s-sparse rows then the complexity of r hvps enjoys a significant reduction from $O(b_{h_j} pr)$ to $O(b_{h_j} sr)$. Hence, computing hvps with $H_{S_j}(w)$ is extremely cheap in the sparse setting.

Randomized low-rank approximation The hvp primitive allows for efficient randomized low-rank approximation to the minibatch Hessian by sketching. Sketching reduces the cost of fundamental numerical linear algebra operations without much loss in accuracy [108,171] by computing the quantity of interest from a *sketch*, or randomized linear image, of a matrix. In particular, sketching enables efficient computation of a near-optimal low-rank approximation to H_{S_j} [35, 76, 164].

SketchySGD computes a sketch of the subsampled Hessian using hvps and returns a randomized low-rank approximation $\hat{H}_{S_j}$ of H_{S_j} in the form of an eigendecomposition $\hat{V}\hat{\Lambda}\hat{V}^T$, where $\hat{V} \in \mathbb{R}^{p\times r}$ and $\hat{\Lambda} \in \mathbb{R}^{r\times r}$. In this paper, we use the randomized Nyström approximation, following the stable implementation in [163]. The resulting algorithm `RandNysApprox` appears in Section 4.7. The cost of forming the Nyström approximation is $O(b_{h_j} pr + pr^2)$, as we need to perform r minibatch hvps to compute the sketch, and we must perform a skinny SVD at a cost of $O(pr^2)$. The procedure is extremely cheap, as we find empirically we can take r to be 10 or less, so constructing the low-rank approximation has negligible cost.

Remark 4.2.1. If the objective for f includes an l_2-regularizer γ, so that the subsampled Hessian has the form $H_{S_j}(w) = \frac{1}{b_{h_j}}\sum_{i\in S_j}\nabla^2 f_i(w) + \gamma I$, we do not include γ in the computation of the sketch in Algorithm 12. The sketch is only computed using minibatch hvps with $\frac{1}{b_{h_j}}\sum_{i\in S_j}\nabla^2 f_i(w)$.

Setting the learning rate SketchySGD automatically selects the learning rate η whenever it updates the preconditioner. The learning rate update rule is inspired by the analysis of gradient descent (GD) on smooth convex functions, which shows GD converges for a fixed learning rate $\eta = 1/L$, where L is the smoothness constant. In the preconditioned setting, L is replaced by L_P, its preconditioned analogue. SketchySGD approximates the ideal learning rate $1/L_P$ by setting the learning rate as

$$\eta_{\text{SketchySGD}} = \frac{\alpha}{\lambda_1\left((\hat{H}_{S_j}+\rho I)^{-1/2} H_{S'}(w_j)(\hat{H}_{S_j}+\rho I)^{-1/2}\right)}, \tag{4.3}$$

where S' is a fresh minibatch that is sampled independently of S_j, and α is scaling factor whose default value is 1/2. The following logic provides intuition for this choice: if f has a Lipschitz Hessian, then

$$\begin{aligned}&\lambda_1\left((\hat{H}_{S_j}+\rho I)^{-1/2} H_{S'}(w_j)(\hat{H}_{S_j}+\rho I)^{-1/2}\right)\\ &\overset{(1)}{\approx} \lambda_1\left((\hat{H}_{S_j}+\rho I)^{-1/2} H(w_j)(\hat{H}_{S_j}+\rho I)\right)^{-1/2} \overset{(2)}{\leq} 1+\zeta \overset{(3)}{\approx} L_P^{\text{loc}}(w_j),\end{aligned}$$

where $\zeta \in (0,1)$, and $L_P^{\text{loc}}(w)$ is the local preconditioned smoothness constant in some appropriately sized ball centered about w_j. Here (1) is due to Lemma 4.4.8, (2) follows from Proposition 4.4.9, and (3) follows f having a Lipschitz Hessian. Hence SketchySGD is expected to work well as long as the local quadratic model at w_j provides an accurate description of the curvature. Moreover, this step size $\eta_{\text{SketchySGD}} = \mathcal{O}(1)$ is much larger (in the preconditioned space) than the standard step size of $1/L$ (in the original space) for an ill-conditioned problem, and should allow faster convergence to the minimum. These predictions are verified experimentally in Section 4.5.

Crucially, our proposed learning rate can be efficiently computed via matvecs with H_{S_j} and $(\hat{H}_{S_j}+\rho I)^{-1/2}$ using techniques from randomized linear algebra, such as randomized powering and the randomized Lanczos method [94, 108]. As an example, we show how to compute the learning rate using randomized powering in Algorithm 11. Note for any vector v, $(\hat{H}_{S_j}+\rho I)^{-1/2}v$ can be computed efficiently via the formula

$$(\hat{H}_{S_j}+\rho I)^{-1/2}v = \hat{V}\left(\hat{\Lambda}+\rho I\right)^{-1/2}\hat{V}^T v + \frac{1}{\sqrt{\rho}}(v - \hat{V}\hat{V}^T v). \tag{4.4}$$

Algorithm 11 `get_learning_rate`

```
Input: hvp oracle $\mathcal{O}_H$, Nyström approximation factors $\hat{V}, \hat{\Lambda}$, regularization $\rho$, maximum number of iterations $q$
$z = \text{randn}(p, 1)$
$y_0 = z/\|z\|$
for $i = 1, \ldots, q$ do
    Compute $v = \left(\hat{H}_{S_j} + \rho I\right)^{-1/2} y_{i-1}$                {Use (4.4)}
    Compute $v' = H_{S'}v$ by calling oracle $\mathcal{O}_{H_{S'}}$          {$S'$ is a fresh minibatch}
    Compute $y_i = \left(\hat{H}_{S_j} + \rho I\right)^{-1/2} v'$                {Use (4.4)}
    $\lambda_i = y_{i-1}^T y_i$
    $y_i = y_i/\|y_i\|$
end for
Set $\eta = \alpha/\lambda_q$                                    {Default $\alpha$ is 1/2}
Output: $\eta$
```

Computing the SketchySGD update (4.2) **fast** Given the rank-r approximation $\hat{H}_{S_j} = \hat{V}\hat{\Lambda}\hat{V}^\top$ to the minibatch Hessian H_{S_j}, the main cost of SketchySGD relative to standard SGD is computing the search direction $v_k = (\hat{H}_{S_j} + \rho I)^{-1} g_{B_k}$. This (parallelizable) computation requires $O(pr)$ flops, by the matrix inversion lemma [79]:

$$v_k = \hat{V}\left(\hat{\Lambda} + \rho I\right)^{-1} \hat{V}^T g_{B_k} + \frac{1}{\rho}(g_{B_k} - \hat{V}\hat{V}^T g_{B_k}). \tag{4.5}$$

The SketchySGD preconditioner is easy to use, fast to compute, and allows SketchySGD to scale to massive problem instances.

Default parameters for Algorithm 12 We recommend setting the ranks $\{r_j\}$ to a constant value of 10, the regularization $\rho = 10^{-3}L$, and the stochastic Hessian batch sizes $\{b_{h_j}\}$ to a constant value of $\lfloor\sqrt{n_{\text{tr}}}\rfloor$, where n_{tr} is the size of the training set. When the Hessian is constant, i.e. as in least-squares/ridge-regression, we recommend using the preconditioner throughout the optimization, which corresponds to $u = \infty$. In settings where the Hessian is not constant, we recommend setting $u = \left\lceil \frac{n_{\text{tr}}}{b_g} \right\rceil$, which corresponds to updating the preconditioner after each pass through the training set.

4.3 Comparison to previous work

Here we review prior work on stochastic second-order methods, with particular emphasis on those developed for convex optimization problems, which is the main focus of this paper.

Stochastic second-order methods for convex optimization Many authors have developed stochastic second-order methods for large-scale machine learning. Broadly, these schemes can be grouped by whether they use stochastic approximations to both the Hessian and gradient or just a stochastic approximation to the Hessian. Stochastic second-order methods that use exact gradients with a stochastic Hessian approximation constructed via sketching or subsampling include [17, 25, 30, 50, 71, 131, 175, 176]. Methods falling into the second group include [20, 21, 112, 115, 145].

Arguably, the most common strategy employed by stochastic second-order methods is to subsample the Hessian as well as the gradient. These methods either directly apply the inverse of the subsampled Hessian to the stochastic gradient [20, 112, 145], or they do an L-BFGS-style update step with the subsampled Hessian [21, 112, 115]. However, the theory underlying these methods requires large or growing gradient batch sizes [20,21,145], periodic full gradient computation [115], or interpolation [112], which are unrealistic assumptions for large-scale convex problems. Further, many of these methods lack practical guidelines for setting hyperparameters such as batch sizes and learning rate, leading to the same tuning issues that plague stochastic first-order methods.

SketchySGD improves on many of these stochastic second-order methods by providing principled guidelines for selecting hyperparameters and requiring only a modest, constant batch size. Table 4.1 compares SketchySGD with a representative subset of stochastic second-order methods on gradient and Hesssian batch sizes, and iteration complexity required to reach a *fixed suboptimality* $\epsilon > 0$. Notice that SketchySGD is the only method that allows computing the gradient with a small constant batch size! Hence SketchySGD empowers the user to select the gradient batch size that meets their computational constraints. Although for fixed-suboptimality ϵ, SketchySGD only converges at rate of $\mathcal{O}\left(\log(1/\epsilon)/\epsilon\right)$, this is often sufficient for machine learning optimization, as it is well-known that high-precision solutions do not improve generalization [1, 22, 104]. Moreover, most of the methods that achieve ϵ-suboptimality in $\mathcal{O}(\log(1/\epsilon))$ iterations require full gradients, which result in costly iterations when n and p are large. In addition to expensive iterations, full gradient methods can also be much slower in yielding a model with good generalization error [5]. Thus, despite converging linearly to the minimum, full gradient methods are expensive and can be slow to reach good generalization error.

SketchySGD also generalizes subsampled Newton methods; by letting the rank parameter $r_j \to \mathrm{rank}(H_{S_j}) \leq p$, SketchySGD reproduces the algorithm of [20,112,145]. This follows as the error in the randomized Nyström approximation is identically zero when $r_j = \mathrm{rank}(H_{S_j})$, so that $\hat{H}_{S_j} = H_{S_j}$ [163]. Using a full batch gradient $b_g = n$ (and a regularized exact low-rank approximation to the subsampled Hessian), SketchySGD reproduces the method of [50]. In particular, these methods are made practical by this work's analysis and practical parameter selections.

Stochastic second-order methods for non-convex optimization In the past decade, there has been a surge of interest in stochastic second-order methods for non-convex optimization, primarily driven by deep learning. Similar to the convex setting, many of these methods are based on

Table 4.1: **Comparison of stochastic 2nd-order methods.** Fix $\epsilon > 0$, and suppose f is of the form (4.1), and is strongly convex. This table compares the required gradient and Hessian batch sizes of various stochastic second-order methods, and the number of iterations required to output a point satisfying $f(w) - f(w_\star) \leq \epsilon$. Here $\zeta_g,\ \zeta \in (0,1)$, while $G(w)$, $M(w)$, and κ are as in Section 4.1.3. $\tau^\rho(H(w))$ denotes the ρ-dissimilarity, which is defined in Definition 4.4.5, and σ^2 represents the variance of the gradient at the optimum (see Proposition 4.4.10). The ρ-dissimilarity offers the tightest characterization of the required Hessian minibatch size required to ensure a non-trivial approximation of H^ρ. In many interesting settings, it is much smaller than n; see Proposition 4.4.7 and the corresponding discussion.

Method	**Gradient batch size**	**Hessian batch size**	**Iteration complexity**
Newton Sketch [97, 131]	Full	Full	$\mathcal{O}\left(\kappa^2 \log(1/\epsilon)\right)$
Subsampled Newton (Full gradients) [145, 175]	Full	$\widetilde{\mathcal{O}}\left(\frac{M(w)/\mu}{\zeta^2}\right)$	$\mathcal{O}\left(\kappa^2 \log(1/\epsilon)\right)$
Subsampled Newton (Stochastic gradients) [145]	$\widetilde{\mathcal{O}}\left(G(w)^2/\zeta_g^2\right)$	$\widetilde{\mathcal{O}}\left(\frac{M(w)/\mu}{\zeta^2}\right)$	$\mathcal{O}(\kappa^2 \log(1/\epsilon))$
Subsampled Newton (Low-rank) [50, 175]	Full	$\widetilde{\mathcal{O}}\left(\frac{M(w)/\lambda_{r+1}(H(w))}{\zeta^2}\right)$	$\mathcal{O}\left(\kappa^2 \log(1/\epsilon)\right)$
SLBFGS [115]	Full evaluation every epoch	b_h	$\mathcal{O}\left(\kappa^2 \log(1/\epsilon)\right)$
SketchySGD (Algorithm 12)	b_g	$\widetilde{\mathcal{O}}\left(\tau^\rho(H(w))/\zeta^2\right)$	$\mathcal{O}\left(\left[\frac{\mathcal{L}_P}{\gamma_\ell}\frac{\rho}{\mu} + \frac{\sigma^2/(\gamma_\ell \mu)}{\epsilon}\right]\log(1/\epsilon)\right)$

subsampling the Hessian, which is then combined with cubic regularization [92, 161, 172] or trust region methods [144, 174]. Taking a more general perspective, [15, 19] have proposed a broad stochastic model-based framework for trust-region methods that only require the gradient and Hessian approximations to satisfy certain error bounds. Thus, even biased approximations to the gradient and Hessian are allowed in their framework. The framework has also been useful in the development of stochastic line search methods [84, 128] and adaptive stochastic second-order methods [149]. Although all these methods come with strong theoretical guarantees, they have not proven popular in deep learning, due to subproblems that are expensive to solve. To maintain computational tractability, most stochastic second-order methods designed for deep learning forgo theoretical guarantees in favor of scalability and good empirical performance. Popular stochastic second-order optimizers in deep learning include K-FAC [73], Shampoo [74], and AdaHessian [173]. Despite recent advances in stochastic second-order methods for deep learning, the advantage of stochastic second-order methods over SGD and its variants is unclear, so stochastic first-order methods have remained the most popular optimization algorithms for deep learning.

4.4 Theory

In this section we present our main convergence theorems for SketchySGD. As mentioned in the prequel, we do not directly analyze Algorithm 10, but a slightly modified version, which we present in Algorithm 12. The are two differences between Algorithm 10 and Algorithm 12. First,

Algorithm 12 SketchySGD (Theoretical version)

```
**Input:** initialization $w_0$, learning rate $\eta$, hvp oracle $\mathcal{O}_H$, ranks $\{r_j\}$, regularization $\rho$, preconditioner update frequency $u$, stochastic gradient batch size $b_g$, stochastic Hessian batch sizes $\{b_{h_j}\}$, number of inner iterations $m$
for $s = 0, 1, 2, \dots$ do
  for $k = 0, 1, 2, \dots, m-1$ do
    Sample a batch $B_k^{(s)}$
    Compute stochastic gradient $g_{B_k^{(s)}}(w_k^{(s)})$
    if $ms + k \equiv 0 \pmod u$ then
      Set $j = j + 1$
      Sample a batch $S_j$                                   $\{|S_j| = b_{h_j}\}$
      $\Phi = \mathrm{randn}(p, r_j)$                        {Gaussian test matrix}
      $Q = \texttt{qr\_econ}\ (\Phi)$
      Compute sketch $Y = H_{S_j}(w_k)Q$                     $\{r$ calls to $\mathcal{O}_{H_{S_j}}\}$
      $[\hat{V}, \hat{\Lambda}] = \texttt{RandNysApprox}(Y, Q, r_j)$
    end if
    Compute $v_k^{(s)} = (\hat{H}_{S_j} + \rho I)^{-1} g_{B_k^{(s)}}(w_k^{(s)})$ via (4.5)
    $w_{k+1}^{(s)} = w_k^{(s)} - \eta v_k^{(s)}$             {Update parameters}
  end for
  Set $\hat{w}^{(s+1)} = \frac{1}{m}\sum_{k=0}^{m-1} w_k^{(s)}$.
end for
```

Algorithm 10 uses an adaptive learning rate strategy, while Algorithm 12 uses a fixed learned rate. Second, Algorithm 12 breaks the optimization into *stages* involving periodic averaging. At the end of each stage, Algorithm 12 sets the initial iterate for the next stage to be the average of the iterates from the previous stage. In this sense, Algorithm 12 resembles the SVRG algorithm of [85], except there is no full gradient computation. Just as with SVRG, the addition of averaging is needed purely to facilitate analysis; in practice the periodic averaging in Algorithm 12 yields no benefits. We recommend always running Algorithm 10 in practice, which is the version we use for all of our experiments (Section 4.5).

4.4.1 Assumptions

We show convergence of SketchySGD when f is smooth (Assumption 4.4.1) and strongly convex (Assumption 4.4.2).

Assumption 4.4.1 (Differentiability and smoothness)**.** The function f is twice differentiable and L-smooth. Further, each f_i is L_i-smooth with $L_i \leq L_{\max}$ for every $i = 1, \dots, n$.

Assumption 4.4.2 (Strong convexity)**.** The function f is μ-strongly convex for some $\mu > 0$.

4.4.2 Quadratic regularity

Our analysis rests on the idea of relative upper and lower quadratic regularity. This is a generalization of upper and lower quadratic regularity, which was recently introduced by the authors in [56], and refines the ideas of relative convexity and relative smoothness introduced in [71].

Definition 4.4.3 (Relative quadratic regularity)**.** Let f be a twice differentiable function and $A(w) : \mathbb{R}^p \mapsto \mathbb{S}^p_{++}(\mathbb{R})$. Then f is said to be *relatively upper quadratically regular* with respect to A, if for all $w, w', w'' \in \mathbb{R}^p$ there exists $0 < \gamma_u < \infty$, such that

$$f(w') \leq f(w) + \langle g(w), w' - w\rangle + \frac{\gamma_u}{2}\|w' - w\|^2_{A(w'')}. \tag{4.6}$$

Similarly, f is said is to be *relatively lower quadratically regular*, if for all $w, w', w'' \in \mathbb{R}^p$ there exists $0 < \gamma_\ell < \infty$, such that

$$f(w') \geq f(w) + \langle g(w), w' - w\rangle + \frac{\gamma_\ell}{2}\|w' - w\|^2_{A(w'')}. \tag{4.7}$$

We say f is *relatively quadratically regular* with respect to A if $\gamma_u < \infty$ and $\gamma_\ell > 0$.

When $A(w) = H(w)$, the Hessian of f, relative quadratic regularity reduces to quadratic regularity from [56]. Quadratic regularity extends ideas from [71], by having the Hessian be evaluated at a point $w'' \neq w$ in (4.6)–(4.7). This extension, while simple in nature, is essential for establishing convergence under lazy preconditioner updates.

An important thing to note about relative quadratic regularity is it holds for useful settings of A, under standard hypotheses, as shown by the following lemma.

Lemma 4.4.4 (Smoothness and strong convexity implies quadratic regularity)**.** *Let $h : \mathcal{C} \to \mathbb{R}$, where $\mathcal{C}$ is a closed convex subset of $\mathbb{R}^p$. Then the following items hold*

1. *If h is twice differentiable and L-smooth, then for any $\rho > 0$, f is relatively upper quadratically regular with respect to $\nabla^2 h(w) + \rho I$.*

2. *If h is twice differentiable, L-smooth, and μ-strongly convex, then h is relatively quadratically regular with respect to $\nabla^2 h(w)$.*

A proof of Lemma 4.4.4 may be found in Section 4.8.2. Importantly, (4.6) and (4.7) hold with non-vacuous values of γ_u and γ_ℓ. In the case of least-squares $\gamma_u = \gamma_\ell = 1$. More generally, for strongly convex generalized linear models, it can be shown when $A(\cdot) = H(\cdot)$, that γ_u and γ_ℓ are independent of the condition number [56], similar to the result of [71] for relative smoothness and relative convexity. Thus, for many popular machine learning problems, the ratio γ_u/γ_ℓ is independent of the conditioning of the data.

4.4.3 Quality of SketchySGD preconditioner

To control the batch size used to form the subsampled Hessian, we introduce *ρ-dissimilarity*.

Definition 4.4.5 (ρ-dissimilarity)**.** Let $H(w)$ be the Hessian at w. The *ρ-dissimilarity* is

$$\tau^{\rho}(H(w)) = \max_{1\leq i\leq n} \lambda_1\left((H(w)+\rho I)^{-1/2}(\nabla^2 f_i(w)+\rho I)(H(w)+\rho I)^{-1/2}\right).$$

ρ-dissimilarity may be viewed as an analogue of coherence from compressed sensing and low-rank matrix completion [26, 27]. Similar to how the coherence parameter measures the uniformity of the rows of a matrix, $\tau^{\rho}(H(w))$ measures how uniform the curvature of the sample $\{\nabla^2 f_i(w)\}_{1\leq i\leq n}$ is. Intuitively, the more uniform the curvature, the better the sample average $H(w)$ captures the curvature of each individual Hessian, which corresponds to smaller $\tau^{\rho}(H(w))$. On the other hand, if the curvature is highly non-uniform, curvature information of certain individual Hessians will be in disagreement with that of $H(w)$, leading to a large value of $\tau^{\rho}(H(w))$.

The following lemma provides an upper bound on the ρ-dissimilarity. In particular, it shows that the ρ-dissimilarity never exceeds n. The proof may be found in Section 4.8.3.

Lemma 4.4.6 (ρ-dissimilarity never exceeds n)**.** *For any $\rho \geq 0$ and $w\in\mathbb{R}^p$, the following inequality holds*

$$\tau^{\rho}(H(w)) \leq \min\left\{n, \frac{M(w)+\rho}{\mu+\rho}\right\},$$

where $M(w) = \max_{1\leq i\leq n}\lambda_1(\nabla^2 f_i(w))$.

Lemma 4.4.6 provides a worst-case bound on the ρ-dissimilarity—if the curvature of the sample is highly non-uniform or ρ is very small, then the ρ-dissimilarity can be large as n. However, Lemma 4.4.6 neglects the fact that in machine learning, the f_i's are often similar to one another, so $\tau^{\rho}(H(w))$ ought to be much smaller than n.

Clearly, for arbitrary data distributions, the ρ-dissimilarity can be large. The following proposition shows when f is a GLM, and the data satisfies an appropriate sub-Gaussian condition, the ρ-dissimilarity does not exceed the ρ-effective dimension of the population Hessian.

Proposition 4.4.7 (ρ-dissimilarity is small for GLMs in the machine learning setting)**.** *Let $\ell: \mathbb{R}\mapsto\mathbb{R}$ be a smooth and convex loss, and define $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ where $f_i(w) = \ell(x_i^T w)$. Fix $w\in\mathbb{R}^p$. Assume x_i are drawn i.i.d. from some unknown distribution $\mathbb{P}(x)$ for $i\in\{1,\ldots,n\}$. Let $H_\infty(w) = \mathbb{E}_{x\sim\mathbb{P}}[\ell''(x^T w)xx^T] \succ 0$ be the population Hessian matrix, and set $\bar{d}^{\rho}_{\text{eff}}(H_\infty(w)) = \max\{d^{\rho}_{\text{eff}}(H_\infty(w)), 1\}$. Suppose for some constant ν the following conditions hold:*

i. The random vector $z = H_\infty(w)^{-1/2}\sqrt{\ell''(x^T w)}x$ is ν sub-Gaussian.

ii. $n \geq C\bar{d}^{\rho}_{\text{eff}}(H_\infty(w))\log\left(\frac{n}{\delta}\right)\log\left(\frac{d^{\rho}_{\text{eff}}(H_\infty(w))}{\delta}\right)$.

Then with probability at least $1-\delta$,

$$\tau^\rho(H(w)) = \mathcal{O}\left(\bar{d}^\rho_{\text{eff}}(H_\infty(w))\log\left(\frac{n}{\delta}\right)\right).$$

The proof of Proposition 4.4.7 is given in Section 4.8.4, and is based on showing the ρ-dissimlarity is well-behaved at the population level, and that for large enough n, the empirical Hessian concentrates around the population Hessian.

Proposition 4.4.7 shows if f is a GLM, then for large datasets, $\tau^\rho(H(w)) = \tilde{\mathcal{O}}\left(d^\rho_{\text{eff}}(H_\infty(w))\right)$ with high probability. When the eigenvalues of $H_\infty(w)$ decay rapidly, the effective dimension $d^\rho_{\text{eff}}(H_\infty(w))$ should be smaller than $(M(w)+\rho)/(\rho+\mu)$, so Proposition 4.4.7 yields a stronger bound than Lemma 4.4.6. For example, when the eigenvalues of $H_\infty(w)$ decay at a sufficiently fast polynomial rate, it is easily verified that $d^\rho_{\text{eff}}(H_\infty(w)) = \mathcal{O}(1/\sqrt{\rho})$ [13]. Consequently $\tau^\rho(H(w)) = \tilde{\mathcal{O}}(1/\sqrt{\rho})$, which is a significant improvement over the $\mathcal{O}(1/\rho)$ bound of Lemma 4.4.6 when ρ is small. This is crucial, for it is desirable to set ρ small, as this leads to a smaller preconditioned condition number, see Proposition 4.4.9. As polynomial (or faster) decay of the eigenvalues values is common in machine learning problems [39], $\tau^\rho(H(w))$ will typically be much smaller then $\mathcal{O}(1/\rho)$.

Lemma 4.4.8 (Closeness in Loewner ordering between $H^\rho_S(w)$ and $H^\rho(w)$)**.** *Let* $\zeta \in (0,1)$*,* $w \in \mathbb{R}^p$ *, and* $\rho \geq 0$*. Construct* H_S *with batch size* $b_h = \mathcal{O}\left(\frac{\tau^\rho(H(w))\log\left(\frac{d^\rho_{\text{eff}}(H(w))}{\delta}\right)}{\zeta^2}\right)$*. Then with probability at least* $1-\delta$

$$(1-\zeta)H^\rho_S(w) \preceq H^\rho(w) \preceq (1+\zeta)H^\rho_S(w).$$

The proof of Lemma 4.4.8 is provided in Section 4.8.5. Lemma 4.4.8 refines prior analyses such as [175] (which itself refines the analysis of [145]), where b_h depends upon $(M(w)+\rho)/(\mu+\rho)$, which Lemma 4.4.6 shows is always larger than $\tau^\rho(H(w))$. Hence, the dependence upon $\tau^\rho(H(w))$ in Lemma 4.4.8 leads to a tighter bound on the required Hessian batch size. More importantly, Lemma 4.4.8 and the idealized setting of Proposition 4.4.7 help show why algorithms using minibatch Hessians with small batchsizes are able to succeed, a phenomenon that prior worst-case theory is unable to explain. As a concrete example, adopt the setting of Proposition 4.4.7, assume fast eigenvalue decay of the Hessian, and set $\rho = \mathcal{O}(1/n)$. Then Lemma 4.4.8 gives $b_h = \tilde{\mathcal{O}}(\sqrt{n})$, whereas prior analysis based on $(M(w)+\rho)/(\mu+\rho)$ yields a vacuous batch size of $b_h = \tilde{\mathcal{O}}(n)$. Thus, Lemma 4.4.8 supports taking batch sizes much smaller than n. Motivated by this discussion, we recommend a default batch size of $b_h = \sqrt{n}$, which leads to excellent performance in practice; see Section 4.5 for numerical evidence.

Utilizing our results on subsampling and ideas from randomized low-rank approximation, we can establish the following result, which quantifies how the SketchySGD preconditioner reduces the condition number.

Proposition 4.4.9 (Closeness in Loewner ordering between $H(w)$ and $\hat{H}_S^\rho$)**.** *Let* $\zeta \in (0,1)$ *and* $w \in \mathbb{R}^p$*. Construct* $H_S(w)$ *with batch size* $b_h = \mathcal{O}\left(\frac{\tau^\rho(H(w))\log\left(\frac{d_{\text{eff}}^\rho(H(w))}{\delta}\right)}{\zeta^2}\right)$ *and SketchySGD uses a low-rank approximation* $\hat{H}_S$ *to* $H_S(w)$ *with rank* $r = \mathcal{O}(d_{\text{eff}}^{\zeta\rho}(H_S(w)) + \log(\frac{1}{\delta}))$*. Then with probability at least* $1-\delta$*,*

$$(1-\zeta)\frac{1}{1+\rho/\mu}\hat{H}_S^\rho \preceq H(w) \preceq (1+\zeta)\hat{H}_S^\rho. \tag{4.8}$$

The proof of this proposition is given in Section 4.8.6. Proposition 4.4.9 shows that with high probability, the SketchySGD preconditioner reduces the conditioner number from L/μ to $(1+\rho/\mu)$, which yields an L/ρ improvement over the original value. The proposition reveals a natural trade-off between eliminating dependence upon μ and the size of b_h: as ρ decreases to μ (and the preconditioned condition number becomes smaller), the batch size must increase.

In practice, we have found that a fixed value of $\rho = 10^{-3}L$ yields excellent performance for convex problems. Numerical results showing how the SketchySGD preconditioner improves the conditioning of the Hessian throughout the optimization trajectory are presented in Figure 4.12.

Proposition 4.4.9 requires the rank of $\hat{H}_{S_j}$ to satisfy $r_j = \tilde{\mathcal{O}}\left(d_{\text{eff}}^{\zeta\rho}(H_{S_j}(w_j)\right)$, which ensures $\|\hat{H}_{S_j} - H_{S_j}\| \leq \zeta\rho$ holds with high probability (Lemma 3.7.4) so that the approximate Hessian matches the subsampled Hessian up to the level of the regularization ρ.

4.4.4 Controlling the variance of the preconditioned stochastic gradient

To establish convergence of SketchySGD, we must control the second moment of the preconditioned minibatch stochastic gradient. Recall the usual approach for minibatch SGD. In prior work, [72] showed that when each f_i is smooth and convex, the minibatch stochastic gradient of f satisfies the following *expected smoothness* condition:

$$\mathbb{E}\|g_B(w) - g_B(w_\star)\|^2 \leq 2\mathcal{L}(f(w) - f(w_\star)), \tag{4.9}$$

$$\mathbb{E}\|g_B(w)\|^2 \leq 2\mathcal{L}(f(w) - f(w_\star)) + 2\sigma^2, \tag{4.10}$$

$$\mathcal{L} = \frac{n(b_g-1)}{b_g(n-1)}L + \frac{n-b_g}{b_g(n-1)}L_{\max},\ \sigma^2 = \frac{n-b_g}{b_g(n-1)}\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(w_\star)\|^2. \tag{4.11}$$

Building on the analysis of [72], we prove the following proposition, which directly bounds the second moment of the preconditioned stochastic gradient.

Proposition 4.4.10 (Preconditioned expected smoothness and gradient variance)**.** *Suppose that Assumption 4.4.1 holds,* $P = \hat{H}_S^\rho$ *is constructed at* $w_P \in \mathbb{R}^p$*, and* P *satisfies* $H(w_P) \preceq (1+\zeta)P$*. Then the following inequalities hold:*

$$\mathbb{E}_k\|g_B(w) - g_B(w_\star)\|_{P^{-1}}^2 \leq 2\mathcal{L}_P\left(f(w) - f(w_\star)\right),$$

$$\mathbb{E}_k\|g_B(w)\|^2_{P^{-1}} \leq 4\mathcal{L}_P\left(f(w)-f(w_\star)\right)+\frac{2\sigma^2}{\rho},$$

where $\mathcal{L}_P$ and σ^2 are given by

$$\mathcal{L}_P \coloneqq \left[\frac{n(b_g-1)}{b_g(n-1)}\gamma_u^\rho+\frac{n-b_g}{b_g(n-1)}\tau^\rho(H(w_P))\gamma_u^{\rho,\max}\right](1+\zeta),\ \sigma^2 \coloneqq \frac{n-b_g}{b_g(n-1)}\frac{1}{n}\sum_{i=1}^n\|\nabla f_i(w_\star)\|^2.$$

Here, γ_u^ρ is the relative upper quadratic regularity constant of f with respect to $H(w)+\rho I$, and $\gamma_u^{\rho,\max}=\max_{i\in[n]}\gamma_{i,u}^\rho$, where $\gamma_{i,u}^\rho$ is the relative upper quadratic regularity constant of f_i with respect to $\nabla^2 f_i(w)+\rho I$.

The proof of this proposition may be found in Section 4.8.8. Proposition 4.4.10 generalizes (4.9) from [72]. The bounds differ in that Proposition 4.4.10 depends upon $\mathcal{L}_P$, the preconditioned analogue of $\mathcal{L}$, which we call the preconditioned expected smoothness constant.

In our convergence analysis, $\mathcal{L}_P$ plays the same role as the smoothness constant in gradient descent. Proposition 4.4.10 reveals the role of the gradient batch size in determining the expected smoothness constant. As the gradient batch size b_g increases from 1 to n, $\mathcal{L}_P$ decreases from $\tau^\rho(H(w_P))\gamma_u^{\rho,\max}(1+\zeta)$ to $\gamma_u^\rho(1+\zeta)$. Recall preconditioning helps globally when $\gamma_u^{\rho,\max}=\mathcal{O}(1)$. In this case, Proposition 4.4.10 implies that the batch size $b_g=\mathcal{O}(\tau^\rho(H(w_P))$ is needed to ensure $\mathcal{L}_P=\mathcal{O}(1)$. The dependence upon the ρ-dissimilarity is consistent with [56], which shows a similar dependence when all the f_i's are strongly convex. Hence, the ρ-dissimilarity plays a key role in determining the Hessian and gradient batch sizes.

4.4.5 Convergence of SketchySGD

In this section, we present convergence results for Algorithm 12 when f is convex and strongly convex. We first state hypotheses governing the construction of the preconditioner at each update index j.

Assumption 4.4.11 (Preconditioner hyperparameters)**.** Given update frequency u, *total number of stages* s, number of inner iterations m, and $\zeta\in(0,1)$, Algorithm 12 sets hyperparameters as follows:

1. The Hessian batchsize is set as

$$b_{h_j}=\mathcal{O}\left(\frac{\tau^\rho(H(w_j))\log\left(\frac{d^\rho_{\text{eff}}(H(w_j))}{\delta}\right)}{\zeta^2}\right).$$

2. The randomized Nyström approximation is constructed with rank

$$r_j=\mathcal{O}\left(d_{\text{eff}}^{\zeta\rho}(H_S(w_j))+\log\left(\frac{1}{\delta}\right)\right).$$

Assumption 4.4.11, along with Proposition 4.4.9 and a union bound argument, ensures that the preconditioners constructed by Algorithm 12 faithfully approximate the Hessian with high probability throughout all iterations.

Corollary 4.4.12 (Union bound)**.** *Let* $\mathcal{E}^{(1)}_{\frac{ms}{u}} = \bigcap_{j=1}^{\frac{ms}{u}} \mathcal{E}^{(1)}_j$*, and* $\mathcal{E}^{(2)}_{\frac{ms}{u}} = \bigcap_{j=1}^{\frac{ms}{u}} \mathcal{E}^{(2)}_j$ *where*

$$\mathcal{E}^{(1)}_j = \left\{(1-\zeta)\hat{H}^{\rho}_{S_j} \preceq H^{\rho}(w_j) \preceq (1+\zeta)\hat{H}^{\rho}_{S_j}\right\}$$

$$\mathcal{E}^{(2)}_j = \left\{(1-\zeta)\frac{1}{1+\rho/\mu}\hat{H}^{\rho}_{S_j} \preceq H(w_j) \preceq (1+\zeta)\hat{H}^{\rho}_{S_j}\right\},$$

Then under Assumption 4.4.11,

1. *For convex and smooth* f*,* $\mathbb{P}\left(\mathcal{E}^{(1)}_{\frac{ms}{u}}\right) \geq 1 - \frac{ms}{u}\delta$*.*

2. *For strongly convex and smooth* f*,* $\mathbb{P}\left(\mathcal{E}^{(2)}_{\frac{ms}{u}}\right) \geq 1 - \frac{ms}{u}\delta$*.*

Proof. We only prove item 2, as the proof of item 1 is analogous. To this end, observe Assumption 4.4.11 and Proposition 4.4.9 imply

$$\mathbb{P}\left(\mathcal{E}^{(2),\complement}_j\right) \leq \delta.$$

Thus,

$$\mathbb{P}\left(\mathcal{E}^{(2)}_{\frac{ms}{u}}\right) = 1 - \mathbb{P}\left(\bigcup_{j=1}^{ms/u} \mathcal{E}^{(2),\complement}_j\right) \geq 1 - \sum_{j=1}^{ms/u} \mathbb{P}\left(\mathcal{E}^{(2),\complement}_j\right) \geq 1 - \frac{ms}{u}\delta.$$

□

Convergence for convex f

Our first convergence result shows SketchySGD can obtain sublinear convergence to a ball with only one stage s when f is convex but not strongly convex. We shall see that when f is strongly convex, the convergence rate improves from sublinear to linear, but that multiple stages ($s > 1$) are required to realize this improvement.

Theorem 4.4.13 (SketchySGD convex convergence)**.** *Consider Problem Equation* (4.1) *under Assumption 4.4.1. Run Algorithm 12 for* $s = 1$ *stage with* m *inner iterations, using gradient batch size* b_g*, regularization* $\rho > 0$*, learning rate* $\eta = \min\left\{\frac{1}{4\mathcal{L}_P}, \sqrt{\frac{\rho\|w_0 - w_\star\|^2_{P_0}}{2\sigma^2 m}}\right\}$*, update frequency* u*,* $\zeta \in (0,1)$*, and preconditioner hyperparameters specified in Assumption 4.4.11. Then conditioned on* *the event* $\mathcal{E}^{(1)}_{\frac{sm}{u}}$ *in Corollary 4.4.12,*

$$\mathbb{E}\left[f(\hat{w}) - f(w_\star)\right] \leq \frac{8\mathcal{L}_P\|w_0 - w_\star\|_{P_0}^2}{m} + \frac{\sqrt{2\rho}\sigma\|w_0 - w_\star\|_{P_0}}{\sqrt{m}}.$$

The proof is given in Section 4.4.7.

Discussion Theorem 4.4.13 shows that when f is convex, Algorithm 12 equipped with appropriate fixed learning rate converges in expectation to an ϵ-ball around the minimum after $m = \mathcal{O}\left(\frac{\mathcal{L}_P\|w_0-w_\star\|_{P_0}^2}{\epsilon} + \frac{\rho\sigma^2\|w_0-w_\star\|_{P_0}^2}{\epsilon^2}\right)$ iterations. This convergence rate is consistent with previous results on stochastic approximation using SGD for smooth convex objectives with bounded gradient variance [98]. However, for SketchySGD, the iteration complexity depends on the preconditioned expected smoothness constant and the preconditioned initial distance to the optimum, which may be much smaller than their non-preconditioned counterparts. Thus, SketchySGD provides faster convergence whenever preconditioning favorably transforms the problem. We give a concrete example where SketchySGD yield an explicit advantage over SGD in Section 4.4.6 below.

Convergence for strongly convex f

When f is strongly convex, we can prove a stronger result that relies on the following lemma, a preconditioned analogue of the strong convexity lower bound.

Lemma 4.4.14 (Preconditioned strong convexity bound)**.** *Let* $P = \hat{H}_S + \rho I$*. Assume the conclusion of Proposition 4.4.9 holds:* $\hat{H}_S$ *approximates* H *well. Then*

$$f(w) - f(w_\star) \geq \frac{\hat{\gamma}_\ell}{2}\|w - w_\star\|_P^2,$$

where $\hat{\gamma}_\ell = (1-\zeta)\frac{\mu}{\mu+\rho}\gamma_\ell$*.*

The proof of Lemma 4.4.14 is given in Section 4.8.7. We now state the convergence theorem for SketchySGD when f is strongly convex, which makes use of several stages s.

Theorem 4.4.15 (SketchySGD strongly convex convergence)**.** *Instate Assumption 4.4.1 and Assumption 4.4.2. Run Algorithm 12 for*

$$ms \geq \frac{32}{(1-\zeta)\gamma_\ell}\left(\mathcal{L}_P + \frac{2\sigma^2}{\epsilon\rho}\right)(1+\rho/\mu)\log\left(\frac{2(f(w_0) - f(w_\star))}{\epsilon}\right) \text{ iterations,}$$

with gradient batchsize b_g*, learning rate* $\eta = \min\{1/4\mathcal{L}_P, \varepsilon\rho/(8\sigma^2)\}$*, regularization* $\mu \leq \rho \leq L_{\max}$*, update frequency* u*, and preconditioner hyperparameters specified in Assumption 4.4.11. Then* *conditioned on the event $\mathcal{E}^{(2)}_{\frac{sm}{u}}$ in Corollary 4.4.12, Algorithm 12 outputs a point $\hat{w}^{(s)}$ satisfying*

$$\mathbb{E}\left[f(\hat{w}^{(s)}) - f(w_\star)\right] \leq \epsilon.$$

The proof is given in Section 4.4.7, along with the exact values of m and s. An immediate corollary of Theorem 4.4.15 is that, supposing the optimal model interpolates the data so $\sigma^2 = 0$, SketchySGD converges linearly to the optimum.

Corollary 4.4.16 (Convergence under interpolation)**.** *Suppose $\sigma^2 = 0$, and instate the hypotheses of Theorem 4.4.15. Run Algorithm 12 for*

$$ms \geq \frac{32}{(1-\zeta)} \frac{\mathcal{L}_P}{\gamma_\ell} (1 + \rho/\mu) \log\left(\frac{2(f(w_0) - f(w_\star))}{\epsilon}\right) \textit{ iterations}$$

with learning rate $\eta = \frac{1}{4\mathcal{L}_P}$. Then Algorithm 12 outputs a point $\hat{w}^{(s)}$ satisfying

$$\mathbb{E}\left[f(\hat{w}^{(s)}) - f(w_\star)\right] \leq \epsilon.$$

Discussion Theorem 4.4.15 shows that with an appropriate fixed learning rate, SketchySGD (Algorithm 12) outputs an ϵ-suboptimal point in expectation after $ms = \tilde{\mathcal{O}}\left(\frac{\mathcal{L}_P}{\gamma_\ell}\frac{\rho}{\mu} + \frac{2\sigma^2/\mu}{\epsilon\gamma_\ell}\right)$ iterations. For smooth strongly convex f, minibatch SGD with a fixed learning rate can produce an ϵ-suboptimal point (in expectation) after $\tilde{\mathcal{O}}\left(\frac{\mathcal{L}}{\mu} + \frac{\sigma^2}{\epsilon\mu^3}\right)$ iterations [2]. However, this comparison is flawed as minibatch SGD does not perform periodic averaging steps. By Theorem 4.4.15, minibatch SGD with periodic averaging (a special case of Algorithm 12, when the preconditioner is always the identity) only requires $\mathcal{O}\left(\frac{\mathcal{L}}{\mu} + \frac{\sigma^2}{\epsilon\mu}\right)$ iterations to reach an ϵ-suboptimal point in expectation. Comparing the two rates, we see SketchySGD has lower iteration complexity when $\mathcal{L}_P/\gamma_\ell = \mathcal{O}(1)$ and $\gamma_\ell = \Omega(1)$. These relations hold when the objective is quadratic, which we discuss in detail more below (Corollary 4.4.17).

Under interpolation, Corollary 4.4.16 shows SketchySGD with a fixed learning rate converges linearly to ϵ-suboptimality in at most $\mathcal{O}\left(\frac{\mathcal{L}_P}{\gamma_\ell}\frac{\rho}{\mu}\log\left(\frac{1}{\epsilon}\right)\right)$ iterations. When $\mathcal{L}_P/\gamma_\ell$ satisfies $\mathcal{L}_P/\gamma_\ell = \mathcal{O}(1)$, which corresponds to the setting where Hessian information can help, SketchySGD enjoys a convergence rate of $\mathcal{O}\left(\frac{\rho}{\mu}\log\left(\frac{1}{\epsilon}\right)\right)$, faster than the $\mathcal{O}\left(\kappa\log\left(\frac{1}{\epsilon}\right)\right)$ rate of gradient descent. This improves upon prior analyses of stochastic Newton methods under interpolation, which fail to show the benefit of using Hessian information. ρ-Regularized subsampled-Newton, a special case of SketchySGD, was only shown to converge in at most $\mathcal{O}\left(\frac{\kappa L}{\rho}\log\left(\frac{1}{\epsilon}\right)\right)$ iterations in [112, Theorem 1, p. 3], which is worse than the convergence rate of gradient descent by a factor of $\mathcal{O}(L/\rho)$.

[2] This result follows from [72], using strong convexity.

4.4.6 When does SketchySGD improve over SGD?

We now present a concrete setting illustrating when SketchySGD converges faster than SGD. Specifically, when the objective is quadratic and strongly convex, SketchySGD enjoys an improved iteration complexity relative to SGD. In general, improved global convergence cannot be expected beyond quadratic functions without restricting the function class, as it is well-known in the worst case, that second-order optimization algorithms such as Newton's method do not improve over first-order methods [8, 121]. Thus without imposing further assumptions, improved global convergence for quadratic functions is the best that can be hoped for. We now give our formal result.

Corollary 4.4.17 (SketchySGD converges fast for quadratic functions)**.** *Under the hypotheses of Theorem 4.4.15, and the assumptions that f is quadratic, $u = \infty$, and $b_g = \tau^\rho(H)$, the following holds:*

1. *If $\sigma^2 > 0$, after $ms = \mathcal{O}\left(\left[\frac{\rho}{\mu} + \frac{\sigma^2}{\epsilon\mu}\right]\log(1/\epsilon)\right)$ iterations, Algorithm 12 outputs a point $\hat{w}^{(s)}$ satisfying*
$$\mathbb{E}[f(\hat{w}^{(s)}) - f(w_\star)] \leq \epsilon.$$
2. *If $\sigma^2 = 0$, after $ms = \mathcal{O}\left(\frac{\rho}{\mu}\log\left(\frac{1}{\epsilon}\right)\right)$ iterations, Algorithm 12 outputs a point $\hat{w}^{(s)}$ satisfying,*
$$\mathbb{E}[f(\hat{w}^{(s)}) - f(w_\star)] \leq \epsilon.$$

Recall minibatch SGD has iteration complexity of $\mathcal{O}\left(\left[\frac{\mathcal{L}}{\mu} + \frac{\sigma^2}{\epsilon\mu}\right]\log(1/\epsilon)\right)$ when $\sigma^2 > 0$, and $\mathcal{O}\left(\frac{\mathcal{L}}{\mu}\log(1/\epsilon)\right)$ when $\sigma^2 = 0$. Thus, Corollary 4.4.17 shows SketchySGD (Algorithm 12) roughly enjoys an $\mathcal{O}\left(\mathcal{L}/\rho\right)$ improvement in iteration complexity relative to SGD, provided the gradient batch size satisfies $b_g = \mathcal{O}(\tau^\rho(H))$. We find the prediction that SketchySGD outperforms its non-preconditioned counterpart on ill-conditioned problems, is realized by the practical version (Algorithm 10) in our experiments.

Remark 4.4.18 (When does better iteration complexity imply fast computational complexity?)*.* To understand when SketchySGD has lower computational complexity than SGD, assume interpolation ($\sigma^2 = 0$), and $L \asymp L_{\max}$. Under these hypotheses, the optimal total computational complexity of SGD is $\tilde{\mathcal{O}}(\kappa p)$, which is achieved with $b_g = 1$. By comparison, SketchySGD (with $b_g = \tau^\rho(H)$) has total computational complexity $\tilde{\mathcal{O}}\left(\frac{\rho}{\mu}\tau^\rho(H)p\right)$. Then if $\tau^\rho(H) = \mathcal{O}(1/\sqrt{\rho})$, and $\rho = \theta L_{\max}$ where $\theta \in (0, 1)$, SketchySGD enjoys an improved computational complexity on the order of $\mathcal{O}\left(\sqrt{L_{\max}/\theta}\right)$. Hence when $\tau^\rho(H)$ is not too large, SketchySGD also enjoys better computational complexity.

4.4.7 Proofs of Theorem 4.4.13 and Theorem 4.4.15

We now turn to the proofs of Theorem 4.4.13 and Theorem 4.4.15. To avoid notational clutter in the proofs, we employ the following notation for the preconditioner at iteration k of stage s.

$$P_k^{(s)} := \hat{H}_{S_j} + \rho I,$$

Here j corresponds to the index of the current preconditioner, so that multiple values of k may be mapped to the same index j. For the convex case we omit the the superscript and simply write P_k, as $s = 1$. With this notational preliminaries out the way, we now commence with the proofs.

Proof of Theorem 4.4.13

Proof. Expanding, and taking the expectation conditioned on k, we reach

$$\mathbb{E}_k\|w_{k+1} - w_\star\|_{P_k}^2 = \|w_k - w_\star\|_{P_k}^2 - 2\eta\langle P_k^{-1}g_k, w_k - w_\star\rangle_{P_k} + \eta^2\mathbb{E}_k\|g_{B_k}\|_{P_k^{-1}}^2 \quad (*).$$

Now, by convexity and Proposition 4.4.10, $(*)$ becomes

$$\mathbb{E}_k\|w_{k+1} - w_\star\|_{P_k}^2 \leq \|w_k - w_\star\|_{P_k}^2 + 2\eta\left(2\eta\mathcal{L}_P - 1\right)\left(f(w_k) - f(w_\star)\right) + 2\sigma^2/\rho.$$

Summing the above display from $k = 0, \cdots m - 1$ yields

$$\sum_{k=0}^{m-1}\mathbb{E}_k\|w_{k+1} - w_\star\|_{P_k}^2 \leq \sum_{k=0}^{m-1}\|w_k - w_\star\|_{P_k}^2 + 2\eta m\left(2\eta\mathcal{L}_P - 1\right)\frac{1}{m}\sum_{k=0}^{m-1}[f(w_k) - f(w_\star)] + \frac{2m\eta^2\sigma^2}{\rho}.$$

Rearranging, and using convexity of f in conjunction with $\hat{w} = \frac{1}{m}\sum_{k=0}^{m-1}w_k$, reaches

$$\sum_{k=0}^{m-1}\mathbb{E}_k\|w_{k+1} - w_\star\|_{P_k}^2 + 2\eta m\left(1 - 2\eta\mathcal{L}_P\right)[f(\hat{w}) - f(w_\star)] \leq \sum_{k=0}^{m-1}\|w_k - w_\star\|_{P_k}^2 + \frac{2m\eta^2\sigma^2}{\rho}.$$

Taking the total expectation over all iterations, we find

$$2\eta m\left(1 - 2\eta\mathcal{L}_P\right)\mathbb{E}\left[f(\hat{w}) - f(w_\star)\right] \leq \|w_0 - w_\star\|_{P_0}^2 + \frac{2m\eta^2\sigma^2}{\rho}.$$

Consequently, we have

$$\mathbb{E}\left[f(\hat{w}) - f(w_\star)\right] \leq \frac{1}{2\eta(1 - 2\eta\mathcal{L}_P)m}\|w_0 - w_\star\|_{P_0}^2 + \frac{\eta\sigma^2}{(1 - 2\eta\mathcal{L}_P)\rho}.$$

Setting $\eta = \min\left\{\frac{1}{4\mathcal{L}_P}, \sqrt{\frac{\rho\|w_0-w_\star\|_{P_0}^2}{2\sigma^2 m}}\right\}$, we conclude

$$\mathbb{E}\left[f(\hat{w}) - f(w_\star)\right] \leq \frac{8\mathcal{L}_P\|w_0 - w_\star\|_{P_0}^2}{m} + \frac{\sqrt{2\rho}\sigma\|w_0 - w_\star\|_{P_0}}{\sqrt{m}}.$$

Strongly convex case Suppose we are in stage s of Algorithm 12. Following identical logic to the convex case, we reach

$$\begin{aligned}&\sum_{k=0}^{m-1}\mathbb{E}_k\|w_{k+1}^{(s)} - w_\star\|_{P_k^{(s)}}^2 + 2\eta m\left(2\eta\mathcal{L}_P - 1\right)\left[f(\hat{w}^{(s+1)}) - f(w_\star)\right]\\ &\leq \sum_{k=0}^{m-1}\|w_k^{(s)} - w_\star\|_{P_k^{(s)}}^2 + \frac{2m\eta^2\sigma^2}{\rho}.\end{aligned}$$

Now, taking the total expectation over all inner iterations conditioned on outer iterations 0 through s, yields

$$2\eta m(2\eta\mathcal{L}_P - 1)\left(\mathbb{E}_{0:s}[f(\hat{w}^{(s+1)})] - f(w_\star)\right) \leq \|\hat{w}^{(s)} - w_\star\|_{P_0^{(s)}}^2 + \frac{2m\eta^2\sigma^2}{\rho}.$$

Invoking Lemma 4.4.14, and rearranging, the preceding display becomes

$$\mathbb{E}_{0:s}[f(\hat{w}^{(s+1)})] - f(w_\star) \leq \frac{1}{\hat{\gamma}_\ell\eta(1-2\eta\mathcal{L}_P)m}\left(f(\hat{w}^{(s)}) - f(w_\star)\right) + \frac{\eta\sigma^2}{(1-2\eta\mathcal{L}_P)\rho}.$$

Setting $m = \frac{16\left(\mathcal{L}_P + 2\sigma^2/(\epsilon\rho)\right)}{\hat{\gamma}_\ell}$ and using $\eta = \min\{1/4\mathcal{L}_P, \frac{\varepsilon\rho}{8\sigma^2}\}$, the previous display and a routine computation yield

$$\mathbb{E}_{0:s}[f(\hat{w}^{(s+1)})] - f(w_\star) \leq \frac{1}{2}\left(f(\hat{w}^{(s)}) - f(w_\star)\right) + \frac{\varepsilon}{4}.$$

Taking the total expectation over all outer iterations and recursing, obtains

$$\mathbb{E}[f(\hat{w}^{(s)})] - f(w_\star) \leq \left(\frac{1}{2}\right)^s\left(f(w_0) - f(w_\star)\right) + \frac{\varepsilon}{2}.$$

Setting $s = 2\log\left(\frac{2(f(w_0)-f(w_\star))}{\epsilon}\right)$, we conclude

$$\mathbb{E}[f(\hat{w}^{(s)})] - f(w_\star) \leq \epsilon,$$

as desired. □

4.5 Numerical experiments

In this section, we evaluate the performance of SketchySGD through six sets of experiments. These experiments are presented as follows:

- Comparisons to first-order methods (Section 4.5.1): We compare SketchySGD to SGD, SVRG, minibatch SAGA [63] (henceforth referred to as SAGA) and loopless Katyusha (L-Katyusha) on ridge regression and l_2-regularized logistic regression. SketchySGD outperforms the competitor methods, even after they have been tuned.
- Comparisons to second-order methods (Section 4.5.2): We compare SketchySGD to L-BFGS [103], stochastic LBFGS (SLBFGS) [115], randomized subspace Newton (RSN) [71], and Newton Sketch [131] on ridge regression and l_2-regularized logistic regression. SketchySGD either outperforms or performs comparably to these methods.
- Comparisons to preconditioned CG (PCG) (Section 4.5.3): We compare SketchySGD to Jacobi PCG [82, 160], sketch-and-precondition PCG (with Gaussian and sparse embeddings) [10, 34, 108, 113], and Nyström PCG on ridge regression from Chapter 2. Again, SketchySGD either outperforms or performs comparably to these methods.
- Large-scale logistic regression (Section 4.5.4): We compare SketchySGD to SGD and SAGA on a random features transformation of the HIGGS dataset, where computing full gradients of the objective is computationally prohibitive. SketchySGD vastly outperforms the competition in this setting.
- Tabular deep learning with multi-layer perceptrons (Section 4.5.5): We compare SketchySGD to popular first-order (SGD, Adam, Yogi [89, 142, 178]) and second-order methods in deep learning (AdaHessian, Shampoo [74, 152, 173]). SketchySGD outperforms the second-order methods and performs comparably to the first-order methods.
- In the supplement, we provide ablation studies for SketchySGD's key hyperparmeters: update frequency (Section 4.11.2), rank parameter (Section 4.11.2), and learning rate (Section 4.11.4). We also demonstrate how SketchySGD improves problem conditioning in Section 4.11.5.

For convex problems, we run SketchySGD with two different preconditioners: (1) Nyström, which takes a randomized-low rank approximation to the subsampled Hessian and (2) Subsampled Newton (SSN), which uses the subsampled Hessian without approximation. Recall, this is a special case of the Nyström preconditioner for which the rank r_j is equal to the Hessian batch size b_{h_j}. SketchySGD is ran according to the defaults presented in Section 4.2, except for in the deep learning experiments (Section 4.5.5).

The datasets used in Sections 4.5.1 to 4.5.3 are presented in Table 4.2. Datasets with "-rf" after their names have been transformed using random features [110, 135]. The condition number reported in Table 4.2 is a lower bound on the condition number of the corresponding ridge regression/logistic regression problem.[3]

Table 4.2: **Datasets and summary statistics.**

Dataset	n_{tr}	n_{test}	p	nonzeros %	Condition number	Task
E2006-tfidf [90]	16087	3308	150360	0.8256	1.051×10^6	Ridge
YearPredictionMSD-rf [46]	463715	51630	4367	50.58	1.512×10^4	Ridge
yolanda-rf [75]	320000	80000	1000	100	1.224×10^4	Ridge
ijcnn1-rf [133]	49990	91701	2500	100	3.521×10^5	Logistic
real-sim [29]	57847	14462	20958	0.2465	1.785×10^1	Logistic
susy-rf [14]	4500000	500000	1000	100	4.626×10^8	Logistic

Each method is run for 40 full gradient evaluations (except Sections 4.5.4 and 4.5.5, where we use 10 and 105, respectively). Note for SVRG, L-Katyusha, and SLBFGS, this corresponds to 20 epochs, as at the end of each epoch SVRG, they compute full a gradients to perform variance reduction. All methods use a gradient batch size of $b_g = 256$.

We plot the distribution of results for each dataset and optimizer combination over several random seeds to reduce variability in the outcomes; the solid/dashed lines show the median and shaded regions represent the 10–90th quantile. The figures we show are plotted with respect to both wall-clock time and full gradient evaluations. We truncate plots with respect to wall-clock time at the time when the second-fastest optimizer terminates. We place markers at every 10 full data passes for curves corresponding to SketchySGD, allowing us to compare the time efficiency of using the Nyström and SSN preconditioners in our method.

Additional details appear in Section 4.10 and code to reproduce our experiments may be found at the git repo `https://github.com/udellgroup/SketchySGD`.

4.5.1 SketchySGD outperforms first-order methods

In this section, we compare SketchySGD to the first-order methods SGD, SVRG, SAGA, and L-Katyusha. In Section 4.5.1, we use the default values for the learning rate/smoothness hyperparameters, based on recommendations made in [37, 85, 93] and scikit-learn [130][4] (see Section 4.10 for more details). In Section 4.5.1, we tune the learning rate/smoothness hyperparameter via grid search. Across both settings, SketchySGD outperforms the competition.

First-order methods — defaults

Figures 4.2 and 4.3 compare SketchySGD to first-order methods run with their defaults. SketchySGD (Nyström) and SketchySGD (SSN) uniformly outperform their first-order counterparts, sometimes

[3] Details of how this lower bound is computed are given in Section 4.9.

[4] SGD does not have a default learning rate. Therefore, we exclude this method from this comparison.

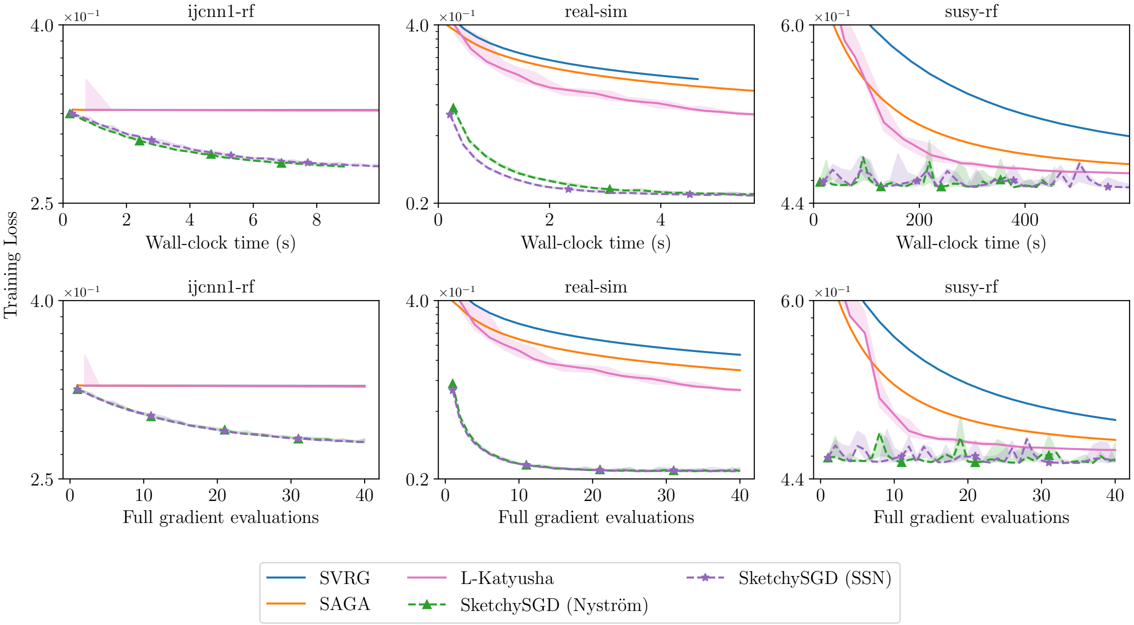


Figure 4.2: Comparisons to first-order methods with default learning rates (SVRG, SAGA) and smoothness parameters (L-Katyusha) on l_2-regularized logistic regression.

dramatically. In the case of the E2006-tfidf and ijcnn1-rf datasets, SVRG, SAGA, and L-Katyusha make no progress at all. Even for datasets where SVRG, SAGA, and L-Katyusha do make progress, their performance lags significantly behind SketchySGD (Nyström) and SketchySGD (SSN). Second-order information speeds up SketchySGD without significant computational costs: the plots show both variants of SketchySGD converge faster than their first-order counterparts.

The plots also show that SketchySGD (Nyström) and SketchySGD (SSN) exhibit similar performance, despite SketchySGD (Nyström) using much less information than SketchySGD (SSN). For the YearPredictionMSD-rf and yolanda-rf datasets, SketchySGD (Nyström) performs better than SketchySGD (SSN). We expect this gap to become more pronounced as n grows, for SketchySGD (SSN) requires $O(\sqrt{n}p)$ flops to apply the preconditioner, while SketchySGD (Nyström) needs only $O(rp)$ flops. This hypothesis is validated in Section 4.5.4, where we perform experiments on a large-scale version of the HIGGS dataset.

First-order methods — tuned

Figures 4.4 and 4.5 show that SketchySGD (Nyström) and SketchySGD (SSN) generally match or outperform tuned first-order methods. For the tuned first-order methods, we only show the curve corresponding to the lowest attained training loss.

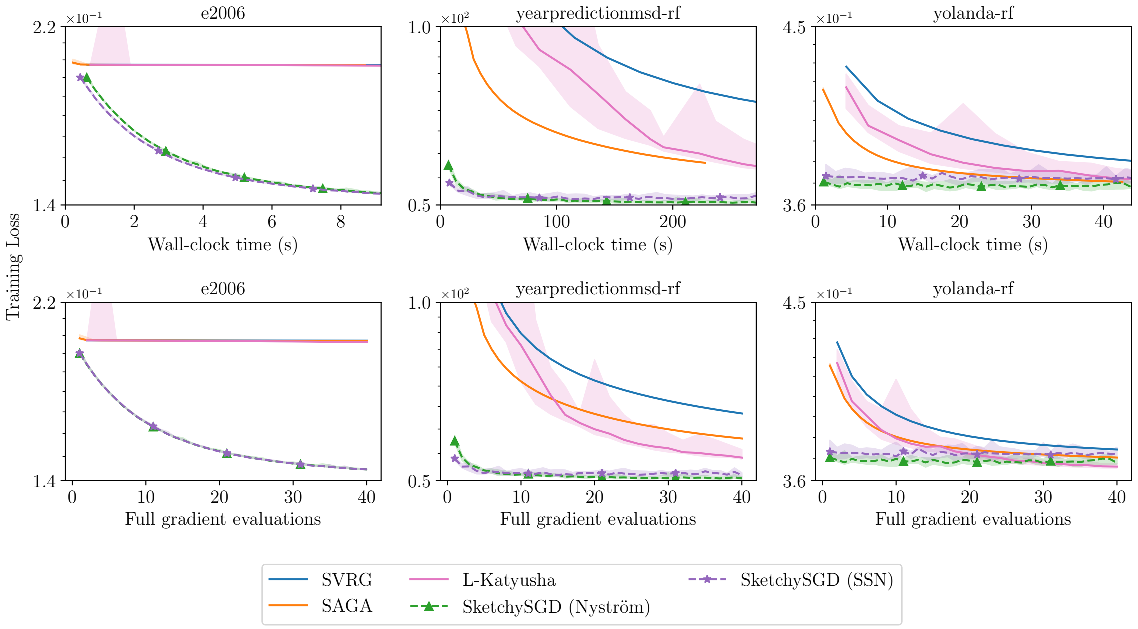


Figure 4.3: Comparisons to first-order methods with default learning rates (SVRG, SAGA) and smoothness parameters (L-Katyusha) on ridge regression.

SketchySGD (Nyström) and SketchySGD (SSN) outperform the competitor methods on E2006-tfidf and YearPredictionMSD-rf, while performing comparably on both yolanda-rf and susy-rf. On real-sim, we find SGD and SAGA perform better than SketchySGD (Nyström) and SketchySGD (SSN) on wall-clock time, but perform similarly on gradient evaluations.

4.5.2 SketchySGD (usually) outperforms second-order methods

We compare SketchySGD to the second-order methods L-BFGS (using the implementation in SciPy), SLBFGS, RSN, and Newton Sketch. For SLBFGS, we tune the learning rate; we do not tune learning rates for L-BFGS, RSN, or Newton Sketch since these methods use line search. The results for logistic and ridge regression are presented in Figures 4.6 and 4.7, respectively. In several plots, SLBFGS cuts off early because it tends to diverge at the best learning rate obtained by tuning. L-BFGS also terminates early on ijcnn1-rf and real-sim because it reaches a high-accuracy solution in under 40 iterations. We are generous to methods that use line search — we do not account for the number of function evaluations performed by L-BFGS, RSN, and Newton Sketch, nor do we account for the number of additional full gradient evaluations performed by L-BFGS to satisfy the strong Wolfe conditions.

Out of all the methods, SketchySGD provides the most consistent performance. When considering wall-clock time performance, SketchySGD is only outperformed by SLBFGS (although it eventually

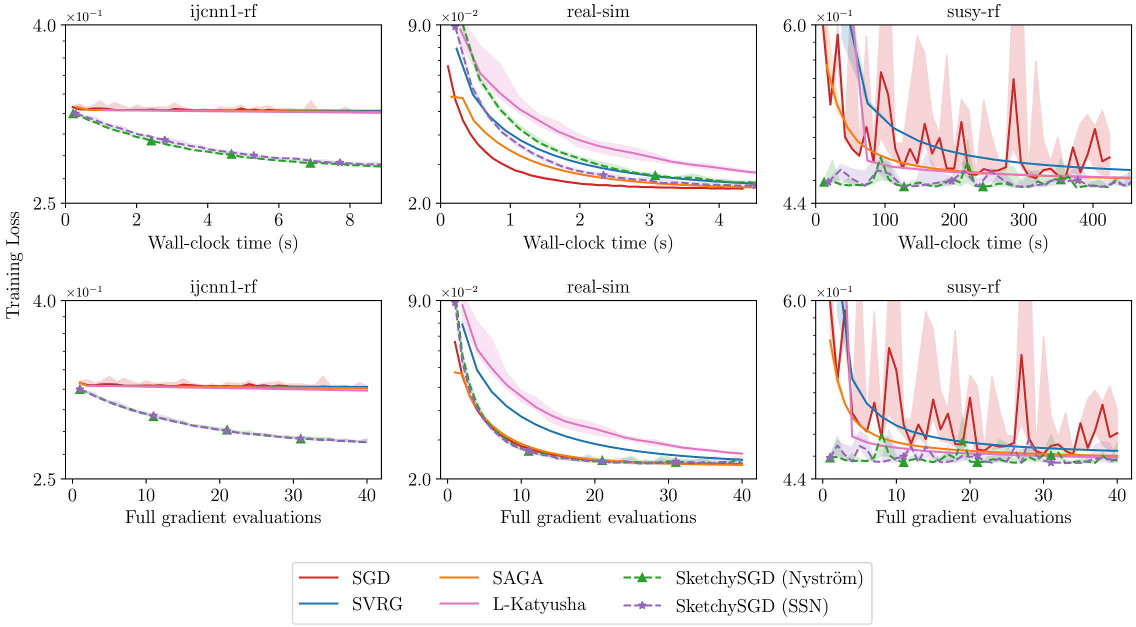


Figure 4.4: Comparisons to first-order methods with tuned learning rates (SGD, SVRG, SAGA) and smoothness parameters (L-Katyusha) on l_2-regularized logistic regression.

diverges) and Newton Sketch on ijcnn1-rf, L-BFGS on real-sim, and RSN on yolanda-rf. On larger, dense datasets, such as YearPredictionMSD-rf and susy-rf, SketchySGD is the clear winner. We expect the performance gap between SketchySGD and the second-order methods to grow as the datasets become larger, and we show this is the case in Section 4.12.1.

4.5.3 SketchySGD (usually) outperforms PCG

We compare SketchySGD to PCG with Jacobi, Nyström, and sketch-and-precondition (Gaussian and sparse embeddings) preconditioners. The results for ridge regression are presented in Figure 4.8. For PCG, full gradient evaluations refer to the total number of iterations.

Similar to the results in Section 4.5.2, SketchySGD provides the most consistent performance. On wall-clock time, SketchySGD is eventually outperformed by JacobiPCG on E2006-tfidf and NyströmPCG on yolanda-rf. However, SketchySGD (Nyström) already reaches a reasonably low training loss within 10 seconds and 2 seconds on YearPredictionMSD-rf and yolanda-rf, respectively, while the PCG methods take much longer to reach this level of accuracy. Furthermore, SketchySGD outperforms the sketch-and-precondition methods on all datasets. We expect the performance gap between SketchySGD and PCG to grow as the datasets become larger, and we show this is the case in Section 4.12.2.

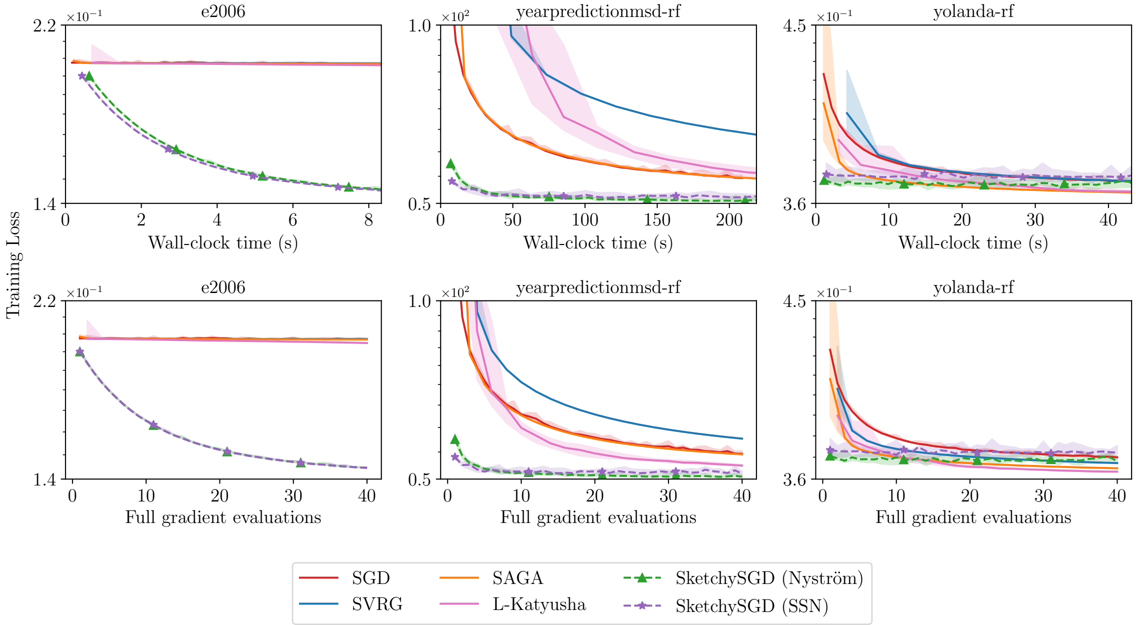


Figure 4.5: Comparisons to first-order methods with tuned learning rates (SGD, SVRG, SAGA) and smoothness parameters (L-Katyusha) on ridge regression.

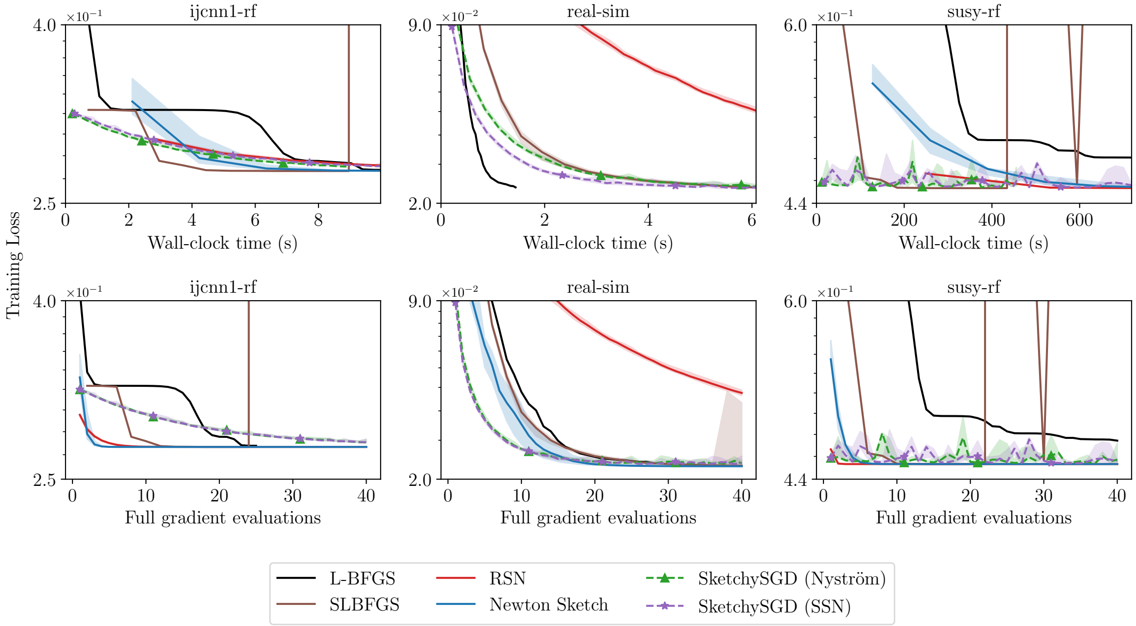


Figure 4.6: Comparisons to second-order methods (L-BFGS, SLBFGS, RSN, Newton Sketch) on l_2-regularized logistic regression.

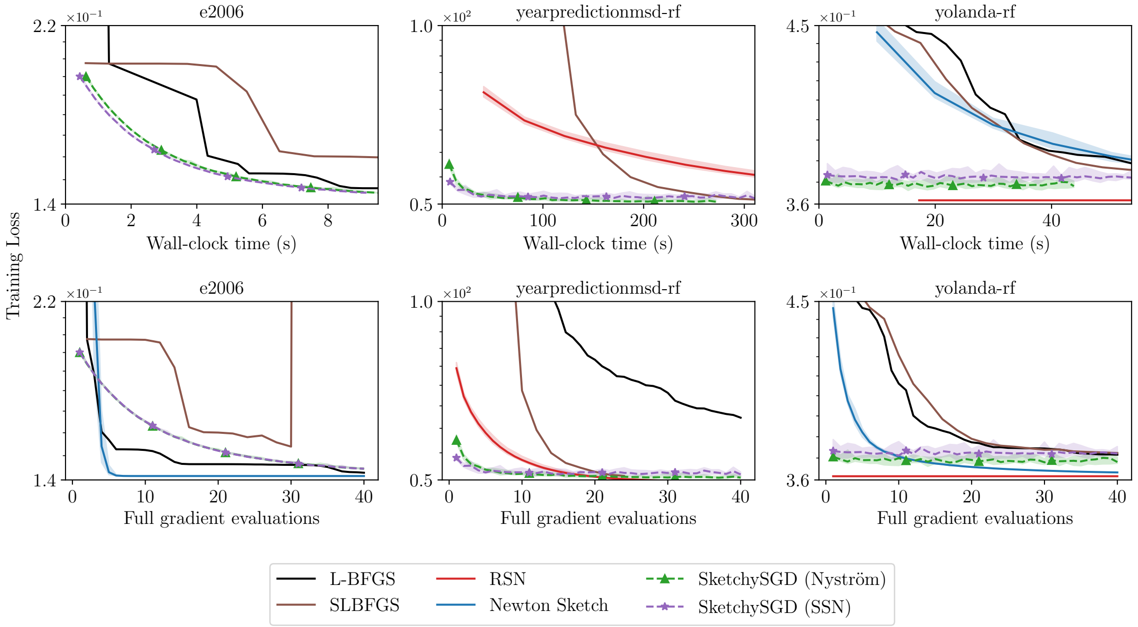


Figure 4.7: Comparisons to second-order methods (L-BFGS, SLBFGS, RSN, Newton Sketch) on ridge regression.

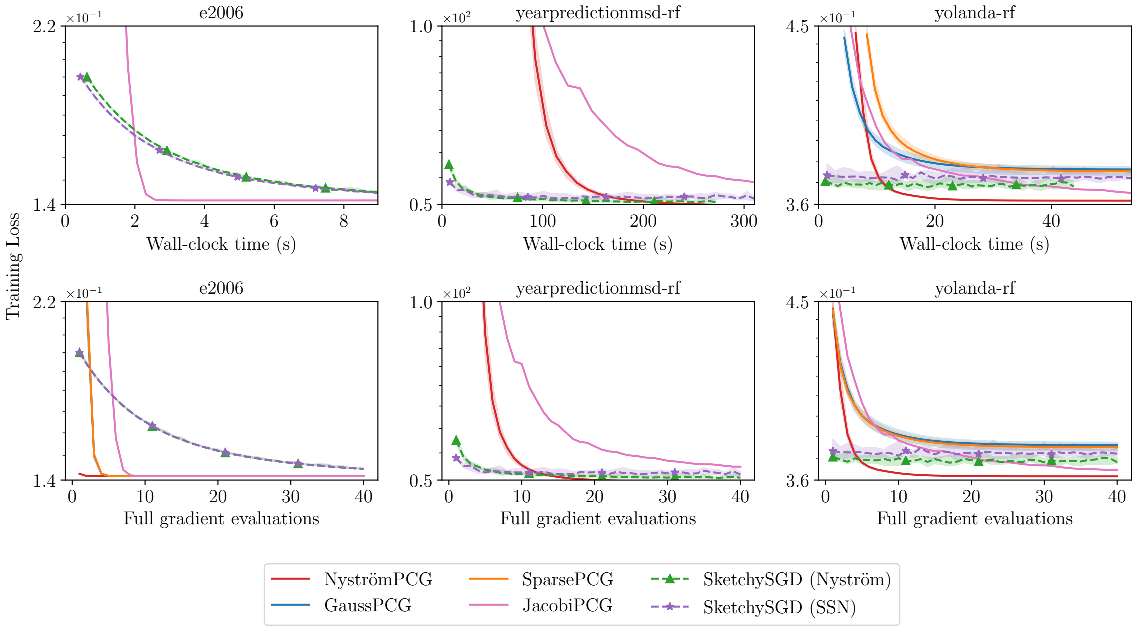


Figure 4.8: Comparisons to PCG (Jacobi, Nyström, sketch-and-precondition w/ Gaussian and sparse embeddings) on ridge regression.

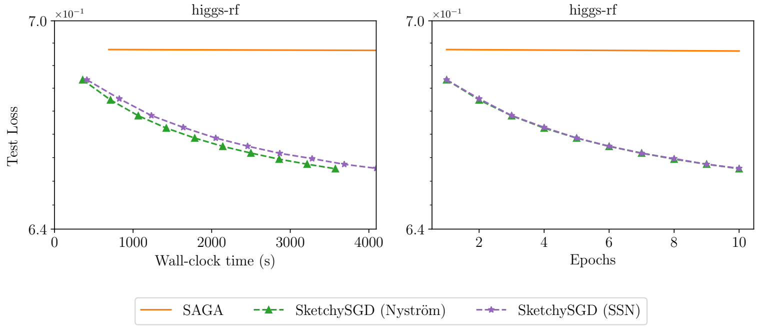


Figure 4.9: Comparison between SketchySGD and SAGA with default learning rate.

4.5.4 SketchySGD outperforms competitor methods on large-scale data

We apply random Fourier features to the HIGGS dataset, for which $(n_{\mathrm{tr}}, p) = (1.05 \cdot 10^7, 28)$, to obtain a transformed dataset with size $(n_{\mathrm{tr}}, p) = (1.05 \cdot 10^7, 10^4)$. This transformed dataset is 840 GB, larger than the hard drive and RAM capacity of most computers. To optimize, we load the original HIGGS dataset in memory and at each iteration, form a minibatch of the transformed dataset by applying the random features transformation to a minibatch of HIGGS. In this setting, computing a full gradient of the objective is computationally prohibitive. We exclude SVRG, L-Katyusha, and SLBFGS since they require full gradients.

We compare SketchySGD to SGD and SAGA with both default learning rates (SAGA only) and tuned learning rates (SGD and SAGA) via grid search.

Figures 4.9 and 4.10 show these two sets of comparisons[5]. We only plot test loss, as computing the training loss is as expensive as computing a full gradient. The wall-clock time plots only show the time taken in optimization; they do not include the time taken in repeatedly applying the random features transformation. We find SGD and SAGA make little to no progress in decreasing the test loss, even after tuning. However, both SketchySGD (Nyström) and SketchySGD (SSN) are able to decrease the test loss significantly. Furthermore, SketchySGD (Nyström) is able to achieve a similar test loss to SketchySGD (SSN) while taking less time, confirming that SketchySGD (Nyström) can be more efficient than SketchySGD (SSN) in solving large problems.

4.5.5 Tabular deep learning with multilayer perceptrons

Our experiments closely follow the setting in [86]. We compare SketchySGD (Nyström) to SGD, Adam, AdaHessian, Yogi, and Shampoo, which are popular optimizers for deep learning. For experiments in this setting, we modify SketchySGD (Nyström) to use momentum and gradient debiasing, as in Adam (Section 4.7.1); we also set $\rho = 10^{-1}$ since it provides better performance. We

[5] The wall-clock time in Figure 4.10 cuts off earlier than in Figure 4.9 due to the addition of SGD, which completes 10 epochs faster than the other methods.

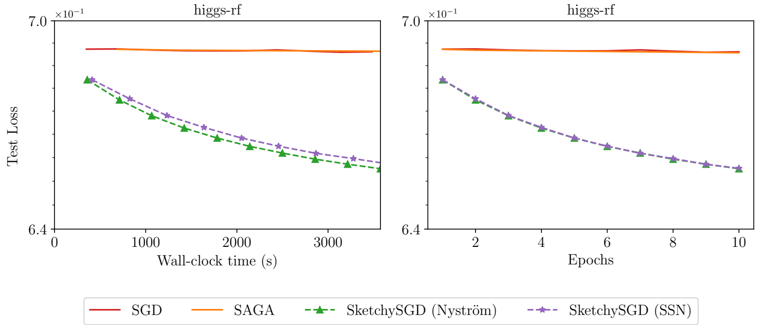


Figure 4.10: Comparison between SketchySGD, SGD and SAGA with tuned learning rates.

use a 9-layer MLP with 512 units in each layer, and cosine annealing for learning rate scheduling. We run the methods on the Fashion-MNIST, Devnagari-Script, and volkert datasets from OpenML [166]. Throughout, we use the weighted cross-entropy loss and balanced accuracy as evaluation metrics. We only tune the initial learning rate for the methods and do so via random search. We form a 60/20/20 training/validation/test split of the data, and select the learning rate with the highest balanced validation accuracy to generate the results reported in this section.

Test accuracy curves for each optimizer are presented in Figure 4.11, and final test accuracies with quantiles are given in Table 4.3. We see SketchySGD consistently outperforms Shampoo, which is an optimizer designed to approximate full-matrix AdaGrad [47]. Furthermore, SketchySGD tends to be more stable than SGD after tuning. However, it is unclear whether SketchySGD performs better than SGD, Adam, or Yogi, which are all first-order optimizers. The reasons for this lack of improvement are unclear, providing an interesting direction for future work.

Table 4.3: **10th and 90th quantiles for final test accuracies.**

Dataset\ Optimizer	SGD	Adam	Yogi	AdaHessian	Shampoo	SketchySGD
Fashion-MNIST	(82.19, 90.47)	(90.10, 90.51)	(90.26, 90.58)	(82.25, 90.53)	(88.59, 89.10)	(90.19, 90.50)
Devnagari-Script	(2.17, 96.23)	(95.57, 95.98)	(95.67, 95.91)	(95.90, 96.17)	(92.18, 93.34)	(96.11, 96.30)
volkert	(64.03, 64.58)	(64.61, 65.36)	(64.08, 64.84)	(63.66, 64.52)	(57.50, 60.22)	(63.90, 65.01)

4.6 Conclusion

In this paper, we have presented SketchySGD, a fast stochastic second-order method for convex machine learning problems. SketchySGD uses subsampling and randomized low-rank approximation to improve conditioning by approximating the curvature of the loss. Furthermore, SketchySGD uses

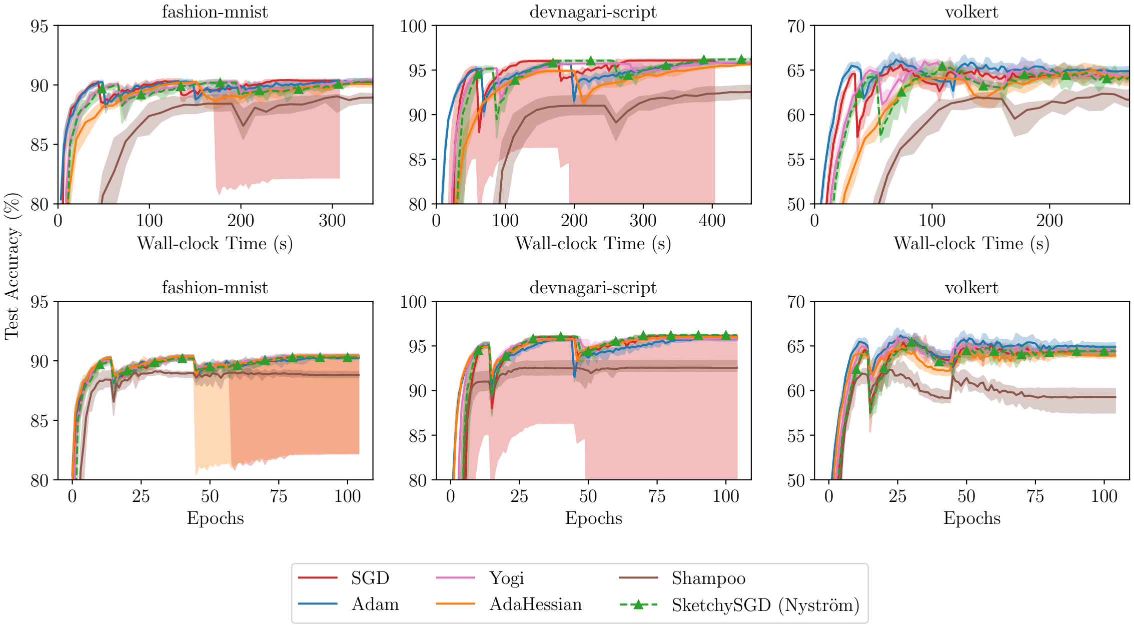


Figure 4.11: Test accuracies for SketchySGD and competitor methods on tabular deep learning tasks.

a novel automated learning rate and comes with default hyperparameters that enable it to work out of the box without tuning.

SketchySGD has strong benefits both in theory and in practice. For quadratic objectives, our theory shows SketchySGD converges to ϵ-suboptimality at a faster rate than SGD, and our experiments validate this improvement in practice. SketchySGD with its default hyperparameters outperforms or matches the performance of SGD, SAGA, SVRG, SLBFGS, and L-Katyusha (the last four of which use variance reduction), even when optimizing the learning rate for the competing methods using grid search.

4.7 Additional algorithms

In this section we provide the pseudocode for the `RandNysApprox` algorithm mentioned in Section 4.2, which SketchySGD uses to construct the low-rank approximation $\hat{H}_{S_j}$.

Algorithm 13 follows Algorithm 3 from [163]. eps(x) is defined as the positive distance between x and the next largest floating point number of the same precision as x. The test matrix Q is the same test matrix used to generate the sketch Y of H_{S_k}. The resulting Nyström approximation $\hat{H}_{S_k}$ is given by $\hat{V}\hat{\Lambda}\hat{V}^T$. The resulting Nyström approximation is psd but may have eigenvalues that are equal to 0. In our algorithms, this approximation is always used in conjunction with a regularizer to ensure positive definiteness.

Algorithm 13 `RandNysApprox`

```
Input: sketch $Y \in \mathbb{R}^{p \times r_j}$ of $H_{S_j}$, orthogonalized test matrix $Q \in \mathbb{R}^{p \times r_j}$,
$\nu = \sqrt{p}\text{eps}(\text{norm}(Y, 2))$    {Compute shift}
$Y_\nu = Y + \nu Q$    {Add shift for stability}
$C = \text{chol}(Q^T Y_\nu)$    {Cholesky decomposition: $C^T C = Q^T Y_\nu$}
$B = Y C^{-1}$    {Triangular solve}
$[\hat{V}, \Sigma, \sim] = \text{svd}(B, 0)$    {Thin SVD}
$\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$    {Compute eigs, and remove shift with element-wise max}
Output: $\hat{V}, \hat{\Lambda}$
```

4.7.1 Modifications for deep learning

We make modifications to the randomized Nyström approximation and SketchySGD for deep learning. Algorithm 14 adapts Algorithm 13 to the non-convex (e.g., deep learning) setting, and ensures that the Randomized Nyström approximation remains positive definite. The main difference between Algorithm 14 and Algorithm 13 is that Algorithm 14 comes with a fail-safe step (colored red in the algorithm block) in case the subsampled Hessian is indefinite and the resulting Cholesky decomposition fails. When this failure occurs, the fail-safe step shifts the spectrum of C by its smallest eigenvalue to ensure it is positive definite. When there is no failure, it is easy to see that Algorithm 14 gives the exact same output as Algorithm 13.

Algorithm 14 `RandNysApproxMod`

```
Input: sketch $Y \in \mathbb{R}^{p \times r_j}$ of $H_{S_j}$, orthogonalized test matrix $Q \in \mathbb{R}^{p \times r_j}$
$\nu = \sqrt{p}\ \text{eps}(\text{norm}(Y, 2))$    {Compute shift}
$Y_\nu = Y + \nu Q$    {Add shift for stability}
$\lambda = 0$    {Additional shift may be required for positive definiteness}
$C = \text{chol}(Q^T Y_\nu)$    {Cholesky decomposition: $C^T C = Q^T Y_\nu$}
if chol fails then
  Compute $[W, \Gamma] = \text{eig}(Q^T Y_\nu)$    {$Q^T Y_\nu$ is small and square}
  Set $\lambda = \lambda_{\min}(Q^T Y_\nu)$
  $R = W(\Gamma + |\lambda| I)^{-1/2} W^T$
  $B = YR$    {$R$ is psd}
else
  $B = Y C^{-1}$    {Triangular solve}
end if
$[\hat{V}, \Sigma, \sim] = \text{svd}(B, 0)$    {Thin SVD}
$\hat{\Lambda} = \max\{0, \Sigma^2 - (\nu + |\lambda|) I\}$    {Compute eigs, and remove shift with element-wise max}
Output: $\hat{V}, \hat{\Lambda}$
```

Algorithm 15 adds momentum (which is a hyperparameter β) and gradient debiasing (similar to Adam) to the practical version of SketchySGD Algorithm 10. These changes to the algorithm are shown in red. In addition, Algorithm 15 does not use the automated learning rate; the learning rate is now an input to the algorithm.

Algorithm 15 SketchySGD (Deep learning version)

```
**Input:** initialization $w_0$, learning rate $\eta$, momentum parameter $\beta$, hvp oracle $\mathcal{O}_H$, ranks $\{r_j\}$, regularization $\rho$, preconditioner update frequency $u$, stochastic gradient batch size $b_g$, stochastic Hessian batch sizes $\{b_{h_j}\}$
Initialize $z_{-1} = 0$                                   $\{$Initialize momentum vector$\}$
**for** $k = 0, 1, \ldots, m-1$ **do**
  Sample a batch $B_k$
  Compute stochastic gradient $g_{B_k}(w_k)$
  **if** $k \equiv 0 \pmod u$ **then**
    Set $j = j + 1$
    Sample a batch $S_j$                                  $\{|S_j| = b_{h_j}\}$
    $\Phi = \mathrm{randn}(p, r_j)$                       $\{$Gaussian test matrix$\}$
    $Q = \texttt{qr\_econ}\ (\Phi)$
    Compute sketch $Y = H_{S_j}(w_k)Q$                    $\{r$ calls to $\mathcal{O}_{H_{S_j}}\}$
    $[\hat{V}, \hat{\Lambda}] = \texttt{RandNysApproxMod}(Y, Q, r_j)$
  **end if**
  $z_k = \beta z_{k-1} + (1-\beta) g_{B_k}(w_k)$          $\{$Update biased first moment estimate$\}$
  $\hat{z}_k = z_k/(1-\beta^{k+1})$                       $\{$Compute bias-corrected first moment estimate.$\}$
  Compute $v_k = (\hat{H}_{S_j} + \rho I)^{-1}\hat{z}_k$ via (4.5)
  $w_{k+1} = w_k - \eta v_k$                              $\{$Update parameters$\}$
**end for**
```

4.8 Proofs not appearing in the main chapter

In this section, we give the proofs of claims that are not present in the main chapter.

4.8.1 Proof that SketchySGD is SGD in preconditioned space

Here we give the proof of (4.2) from Section 4.1. As in the prequel, we set $P_j = \hat{H}_{S_j}^{\rho}$ in order to avoid notational clutter. Recall the SketchySGD update is given by

$$w_{k+1} = w_k - \eta_j P_j^{-1} g_{B_k},$$

where $\mathbb{E}_{B_k}[g_{B_k}] = g_k$. We start by making the following observation about the SketchySGD update.

Lemma 4.8.1 (SketchySGD is SGD in preconditioned space)**.** *At outer iteration j define $f_{P_j}(z) = f(P_j^{-1/2}z)$, that is define the change of variable $w = P_j^{-1/2}z$. Then,*

$$\begin{aligned} g_{P_j}(z) &= P_j^{-1/2} g(P_j^{-1/2}z) = P_j^{-1/2} g(w) \\ H_{P_j}(z) &= P_j^{-1/2} H(P_j^{-1/2}z) P_j^{-1/2} = P_j^{-1/2} H(w) P_j^{-1/2}. \end{aligned}$$

Hence the SketchySGD update may be realized as

$$
\begin{aligned}
z_{k+1} &= z_k - \eta_j \hat{g}_{P_j}(z_k) \\
w_{k+1} &= P_j^{-1/2} z_{k+1},
\end{aligned}
$$

where $\hat{g}_{P_j}(z_k) = P_j^{-1/2} g_{B_k}(P_j^{-1/2} z_k)$ *is the stochastic gradient in preconditioned space.*

Proof. The first display of equations follow from the definition of the change of variable and the chain rule, while the last display follows from definition of the SketchySGD update and the first display. □

4.8.2 Proof of Lemma 4.4.4

Proof. Let $w, w', w'' \in \mathcal{C}$ and set $v = w' - w$. Then by Taylor's theorem and smoothness and convexity of h, it holds that

$$
\begin{aligned}
h(w') &= h(w) + \langle \nabla f(w), v\rangle + \left(\int_0^1 (1-t)\frac{\|v\|^2_{\nabla^2 h(w+tv)}}{\|v\|^2_{A(w'')}} dt\right) \|v\|^2_{A(w'')} \\
&= h(w) + \langle \nabla h(w), v\rangle + \frac{\mathcal{I}}{2}\|v\|^2_{A(w'')}, \\
&\text{where } \mathcal{I} = \int_0^1 (1-t)\frac{\|v\|^2_{\nabla^2 h(w+tv)}}{\|v\|^2_{A(w'')}}.
\end{aligned}
$$

Now, if h is just convex and $A(w'') = \nabla^2 h(w'') + \rho I$, a routine calculation shows that $\mathcal{I} \leq L/\rho$, which by definition implies $\gamma_u(\mathcal{C}) \leq L/\rho$. This gives the first statement. For the second statement, note that when h is μ-strongly convex and $A(w'') = \nabla^2 h(w'')$, we have $\mu/L \leq \mathcal{I} \leq L/\mu$, which implies the second claim with $\mu/L \leq \gamma_\ell(\mathcal{C}) \leq \gamma_u(\mathcal{C}) \leq L/\mu$. □

4.8.3 Proof of Lemma 4.4.6

Below we provide the proof of Lemma 4.4.6.

Proof. By convexity of the f_i's and the finite sum structure of f, it is easy to see that

$$\nabla^2 f_i(w) + \rho I \preceq n\left(H(w) + \rho I\right) \ \forall i \in [n].$$

Conjugating both sides by $(H^\rho)^{-1/2}$, we reach

$$(H(w) + \rho I)^{-1/2}\left(\nabla^2 f_i(w) + \rho I\right)(H(w) + \rho I)^{-1/2} \preceq nI.$$

It now immediately follows that $\tau^\rho(H(w)) \leq n$. On the other hand, for all $i \in \{1, \dots n\}$ we have

$$\nabla^2 f_i(w) + \rho I \preceq \left[\lambda_1\left(\nabla^2 f_i(w)\right) + \rho\right] I \preceq \left(M(w) + \rho\right) I,$$

where the last relation follows by definition of $M(w)$. So, conjugating by $\left(H^\rho(w)\right)^{-1/2}$ we reach

$$\left(H^\rho(w)\right)^{-1/2}\left(\nabla^2 f_i(w) + \rho I\right)\left(H^\rho(w)\right)^{-1/2} \preceq \left(M(w) + \rho\right)\left(H(w) + \rho I\right)^{-1} \preceq \frac{M(w)+\rho}{\mu+\rho} I,$$

which immediately yields $\tau^\rho(H(w)) \leq \frac{M(w)+\rho}{\mu+\rho}$. Combining both bounds, we conclude

$$\tau^\rho\left(H(w)\right) \leq \min\left\{n, \frac{M(w)+\rho}{\mu+\rho}\right\}.$$

□

4.8.4 Proof of Proposition 4.4.7

The proof of Proposition 4.4.7 is the culmination of several lemmas. We begin with a *truncated* intrinsic dimension Matrix Bernstein Inequality discussed, only requires bounds on the first and second moments that hold with some specified probability. It refines [80], who established a similar result for the vanilla Matrix Bernstein Inequality.

Lemma 4.8.2 (Truncated Matrix Bernstein with intrinsic dimension)**.** *Let $\{X_i\}_{i\in[n]}$ be a sequence of independent mean zero random matrices of the same size. Let $\beta \geq 0$ and $\{V_{1,i}\}_{i\in[n]}, \{V_{2,i}\}_{i\in[n]}$ be sequences of matrices with $V_{1,i}, V_{2,i} \succeq 0$ for all i. Consider the event $\mathcal{E}_i = \left\{\|X_i\| \leq \beta, X_iX_i^T \preceq V_{1,i}, X_i^TX_i \preceq V_{2,i}\right\}$. Define $Y_i = X_i 1_{\mathcal{E}_i}$, $Y = \sum_{i=1}^n Y_i$. Suppose that the following conditions hold*

$$\mathbb{P}\left(\mathcal{E}_i\right) \geq 1 - \delta \quad \textit{for all } i \in [n],$$

$$\|\mathbb{E}[Y_i]\| \leq q.$$

Set $V_1 = \sum_{i=1}^n V_{1,i}, V_2 = \sum_{i=1}^n V_{2,i}$, and define

$$\mathcal{V} = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}, \ \varsigma^2 = \max\{\|V_1\|, \|V_2\|.\}$$

Then for all $t \geq q + \varsigma + \frac{\beta}{3}$, $X = \sum_{i=1}^n X_i$ satisfies

$$\mathbb{P}\left(\|X\| \geq t\right) \leq n\delta + 4\frac{\operatorname{trace}(\mathcal{V})}{\|\mathcal{V}\|}\exp\left(\frac{-(t-q)^2/2}{\varsigma^2 + \beta(t-q)/3}\right).$$

Proof. The argument consists of relating $\mathbb{P}\left(\|X\| \geq t\right)$ to $\mathbb{P}\left(\|Y\| \geq t\right)$, the latter of which is easily bounded. Indeed, from the definition of the $\mathcal{E}_i$'s, it is easily seen that

$$\|Y_i\| \leq \beta,\ \mathbb{E}\left[(Y-\mathbb{E}[Y])(Y-\mathbb{E}[Y])^T\right] \preceq V_1,\ \mathbb{E}\left[(Y-\mathbb{E}[Y])^T(Y-\mathbb{E}[Y])\right] \preceq V_2.$$

Consequently, the intrinsic dimension Matrix Bernstein inequality [162, Theorem 7.3.1] implies for any $s \geq \varsigma + \beta/3$, that

$$\mathbb{P}\left(\|Y-\mathbb{E}[Y]\| \geq s\right) \leq 4\frac{\text{trace}(\mathcal{V})}{\|\mathcal{V}\|}\exp\left(\frac{-s^2/2}{\varsigma^2+\beta s/3}\right). \tag{4.12}$$

We now relate the tail probability of $\|X\|$ to the tail probability of $\|Y\|$. To this end, the law of total probability implies

$$\begin{aligned}&\mathbb{P}\left(\|X\| \geq t\right) = \mathbb{P}\left(\|Y\| \geq t|X=Y\right)\mathbb{P}(X=Y)\\&+\mathbb{P}\left(\|X\| \geq t|X\neq Y\right)\mathbb{P}(X\neq Y)\\&\leq \mathbb{P}\left(\|Y\| \geq t|X=Y\right)+\mathbb{P}\left(\bigcup_{i=1}^n \mathcal{E}_i^{\complement}\right)\\&\leq \mathbb{P}\left(\|X\| \geq t|X=Y\right)+n\delta,\end{aligned}$$

where the third inequality follows from $\{X\neq Y\} \subset \bigcup_{i=1}^n \mathcal{E}_i^{\complement}$, and the last inequality uses $\mathbb{P}\left(\bigcup_{i=1}^n \mathcal{E}_i^{\complement}\right) \leq \sum_{i=1}^n (1-\mathbb{P}(\mathcal{E}_i)) \leq n\delta$. To bound $\mathbb{P}\left(\|X\| \geq t|X=Y\right)$, observe that

$$\|X\| \leq \|X-\mathbb{E}[Y]\| + \|\mathbb{E}[Y]\| \leq \|X-\mathbb{E}[Y]\| + q,$$

which implies

$$\begin{aligned}&\mathbb{P}\left(\|X\| \geq t|X=Y\right) \leq \mathbb{P}\left(\|X-\mathbb{E}[Y]\|+q \geq t|X=Y\right)\\&=\mathbb{P}\left(\|Y-\mathbb{E}[Y]\| \geq t-q|X=Y\right).\end{aligned}$$

Inserting this last display into our bound for $\mathbb{P}\left(\|X\| \geq t\right)$, we find

$$\mathbb{P}\left(\|X\| \geq t\right) \leq n\delta + \mathbb{P}\left(\|Y-\mathbb{E}[Y]\| \geq t-q|X=Y\right).$$

To conclude, we apply (4.12) with $s = t-q$ to obtain

$$\mathbb{P}\left(\|X\| \geq t\right) \leq n\delta + 4\frac{\text{trace}(\mathcal{V})}{\|\mathcal{V}\|}\exp\left(\frac{-(t-q)^2/2}{\varsigma^2+\beta(t-q)/3}\right),$$

for all $t \geq q+\varsigma+\beta/3$. □

Lemma 4.8.3 (Bounded statistical leverage)**.** *Let* $D_\infty^\rho = H_\infty(w)^{1/2}H_\infty^\rho(w)^{-1}H_\infty(w)^{1/2}$*, and set* $\bar{d}_{\text{eff}}^\rho(H_\infty^\rho(w)) = \max\{d_{\text{eff}}^\rho(H_\infty^\rho(w)), 1\}$*. Then for some absolute constant* $C > 0$*,*

$$\mathbb{P}\left\{\left\|\sqrt{\ell''(x^Tw)}x\right\|_{H_\infty^\rho(w)^{-1}}^2 > C\bar{d}_{\text{eff}}^\rho(H_\infty(w))\log\left(\frac{1}{\delta}\right)\right\} \leq \delta.$$

Proof. Recall that $\sqrt{\ell''(x^Tw)}x = H_\infty^{1/2}(w)z$, so that $\|\sqrt{\ell''(x^Tw)}x\|_{H_\infty^\rho(w)^{-1}}^2 = \|z\|_{D_\infty^\rho}^2$. As z is ν-sub-Gaussian and $\text{trace}(D_\infty^\rho) = d_{\text{eff}}^\rho(H_\infty(w))$, Theorem 2.1 of [81] with $\Sigma = D_\infty^\rho$ implies that

$$\mathbb{P}\left\{\|z\|_{D_\infty^\rho}^2 > \nu^2\left(d_{\text{eff}}^\rho(H_\infty(w)) + 2\sqrt{d_{\text{eff}}^\rho(H(w))t} + 2t\right)\right\} \leq \exp(-t).$$

Setting $t = \bar{d}_{\text{eff}}^\rho(H_\infty^\rho(w))\log(1/\delta)$, we obtain the desired claim with $C = 5\nu^2$. □

Lemma 4.8.4 (Empirical Hessian concentration)**.** *Suppose* $n = \tilde{\Omega}\left[\bar{d}_{\text{eff}}^\rho(H_\infty(w))\log(n/\delta)\right]$*, then*

$$\left\|H_\infty^\rho(w)^{-1/2}\left[H(w) - H_\infty(w)\right]H_\infty^\rho(w)^{-1/2}\right\| \leq 1/2,$$

with probability at least $1 - \delta/n$*.*

Proof. We begin by writing

$$H_\infty^\rho(w)^{-1/2}\left[H(w) - H_\infty(w)\right]H_\infty^\rho(w)^{-1/2} = \frac{1}{n}\sum_{i=1}^n\left(Z_iZ_i^T - D_\infty^\rho\right).$$

where $Z_i = H_\infty^\rho(w)^{-1/2}\sqrt{\ell''(x^Tw)}x_i$ and $D_\infty^\rho = H_\infty^\rho(w)^{-1/2}H_\infty(w)H_\infty^\rho(w)^{-1/2}$. Set $X_i = \frac{1}{n}\left(Z_iZ_i^T - D_\infty^\rho\right)$, and observe that $\mathbb{E}[X_i] = 0$. We seek to apply Lemma 4.8.2, to this end observe that Lemma 4.8.3 implies

$$\max_{i\in[n]}\mathbb{P}\left(\left\|\sqrt{\ell''(x_i^Tw)}x_i\right\|_{H_\infty^\rho(w)^{-1}}^2 > C\bar{d}_{\text{eff}}^\rho(H_\infty(w))\log\left(\frac{2n}{\delta}\right)\right) \leq \frac{\delta}{2n^2}.$$

Consequently, we obtain the following bounds on $\|X_i\|$ and $\mathbb{E}[X_i^2]$:

$$\begin{aligned}
&\|X_i\| = \frac{1}{n}\max\left\{\lambda_{\max}\left(Z_iZ_i^T - D_\infty^\rho\right), -\lambda_{\min}\left(Z_iZ_i^T - D_\infty^\rho\right)\right\} \\
&\leq \frac{1}{n}\max\left\{\|Z_i\|^2, \lambda_{\max}(D_\infty^\rho)\right\} \leq \frac{1}{n}\max\{\|Z_i\|^2, 1\} \\
&= \frac{1}{n}\max\left\{\left\|\sqrt{\ell''(x^Tw)}x_i\right\|_{H_\infty^\rho(w)^{-1}}^2, 1\right\} \leq \frac{C\bar{d}_{\text{eff}}^\rho(H_\infty(w))\log\left(\frac{2n}{\delta}\right)}{n},
\end{aligned}$$

and

$$\mathbb{E}[X_i^2] = \frac{1}{n^2}\mathbb{E}[\|Z_i\|^2Z_iZ_i^T] \preceq \frac{C\bar{d}_{\text{eff}}^\rho(H_\infty(w))\log\left(\frac{2n}{\delta}\right)}{n^2}D_\infty^\rho,$$

Hence setting $\beta = C\bar{d}_{\text{eff}}^{\rho}(H_\infty(w))\log\left(\frac{2n}{\delta}\right)$ and $V_i = \beta D_{\rho}^{\infty}$, it follows immediately from the preceding considerations that

$$\max_{i\in[n]} \mathbb{P}\left(\|X_i\| \leq \beta/n, \mathbb{E}[X_i^2] \preceq \frac{1}{n^2}V_i\right) \geq 1-\delta/(2n^2).$$

As $V_1 = V_2 = V = \beta D_\infty^\rho/n$, it follows that $\|\mathcal{V}\| \leq \beta/n$. Moreover,

$$\begin{aligned}\text{trace}(\mathcal{V})/\|\mathcal{V}\| &= \text{trace}(V)/\|V\| = \text{trace}(D_\infty^\rho)/\|D_\infty^\rho\| \\ &= d_{\text{eff}}^{\rho}(H_\infty(w))\left[1+\rho/\lambda_1(H_\infty(w))\right] \leq 2d_{\text{eff}}^{\rho}(H_\infty(w)),\end{aligned}$$

where the last inequality follows as $\rho \leq \lambda_1(H_\infty(w))$. Thus, we can invoke Lemma 4.8.2 with

$$t = C\left(\sqrt{\frac{\beta\log\left(\frac{d_{\text{eff}}^{\rho}(H_\infty(w))}{\delta}\right)}{n}}+\frac{\beta\log\left(\frac{d_{\text{eff}}^{\rho}(H_\infty(w))}{\delta}\right)}{n}\right),$$

to reach with probability at least $1-\delta/(2n)$ that

$$\begin{aligned}&\left\|H_\infty^\rho(w)^{-1/2}\left[H(w)-H_\infty(w)\right]H_\infty^\rho(w)^{-1/2}\right\| \\ &\leq C\left(\sqrt{\frac{\beta\log\left(\frac{d_{\text{eff}}^{\rho}(H_\infty(w))}{\delta}\right)}{n}}+\frac{\beta\log\left(\frac{d_{\text{eff}}^{\rho}(H_\infty(w))}{\delta}\right)}{n}\right).\end{aligned}$$

Recalling $\beta = \mathcal{O}\left(\bar{d}_{\text{eff}}^{\rho}(H_\infty(w))\log\left(\frac{n}{\delta}\right)\right)$ and $n = \Omega\left(\bar{d}_{\text{eff}}^{\rho}(H_\infty(w))\log\left(\frac{d_{\text{eff}}^{\rho}(H_\infty(w))}{\delta}\right)\log\left(\frac{n}{\delta}\right)\right)$, we conclude from the last display that

$$\mathbb{P}\left(\left\|H_\infty^\rho(w)^{-1/2}\left[H(w)-H_\infty(w)\right]H_\infty^\rho(w)^{-1/2}\right\| \leq 1/2\right) \geq 1-\delta/(2n).$$

□

Proof of Proposition 4.4.7

Proof. Let $Z_i = H_\infty^\rho(w)^{-1/2}\sqrt{\ell''(x_i^Tw)}x_i$ and observe the hypotheses on n, combined with Lemma 4.8.3 and Lemma 4.8.4 imply that

$$\begin{aligned}&\mathbb{P}\left(\max_{i\in[n]}\|Z_i\|^2 \leq C\bar{d}_{\text{eff}}^{\rho}(H_\infty(w))\log\left(\frac{n}{\delta}\right), \quad \frac{1}{2}H_\infty^\rho(w) \preceq H^\rho(w) \preceq \frac{3}{2}H_\infty^\rho(w)\right) \\ &\geq 1-\delta/n.\end{aligned}$$

Combining the previous relation with matrix similarity, we find

$$\begin{aligned}
&\lambda_1\left(H^\rho(w)^{-1/2}\nabla^2 f_i^\rho(w)H^\rho(w)^{-1/2}\right) = \left(\nabla^2 f_i^\rho(w)^{1/2}H^\rho(w)^{-1}\nabla^2 f_i^\rho(w)^{1/2}\right)\\
&\leq 2\left(\nabla^2 f_i^\rho(w)^{1/2}H^\rho_\infty(w)^{-1}\nabla^2 f_i^\rho(w)^{1/2}\right) = 2\lambda_1\left(H^\rho_\infty(w)^{-1/2}\nabla^2 f_i^\rho(w)H^\rho_\infty(w)^{-1/2}\right)\\
&\leq 2+2\lambda_1\left(H^\rho_\infty(w)^{-1/2}\nabla^2 f_i(w)H^\rho_\infty(w)^{-1/2}\right) = 2+2\lambda_1(Z_iZ_i^T) = 2+2\|Z_i\|^2\\
&\leq C\bar{d}^\rho_{\text{eff}}(H_\infty(w))\log\left(\frac{n}{\delta}\right).
\end{aligned}$$

Recalling that $\tau^\rho(H(w)) = \max_{i\in[n]}\lambda_1\left(H^\rho(w)^{-1/2}\nabla^2 f_i^\rho(w)H^\rho(w)^{-1/2}\right)$, the last display and a union bound yield

$$\mathbb{P}\left(\tau^\rho(H(w)) \leq C\bar{d}^\rho_{\text{eff}}(H_\infty(w))\log\left(\frac{n}{\delta}\right)\right) \geq 1-\delta,$$

as desired. □

4.8.5 Proof of Lemma 4.4.8

Proof. The result is a consequence of a standard application of the intrinsic dimension Matrix Bernstein inequality. Indeed, let $D^\rho = (H(w)+\rho I)^{-1/2}H(w)(H(w)+\rho I)^{-1/2}$ and $X_i = \frac{1}{b_h}\left(Z_iZ_i^T - D^\rho\right)$, where $Z_i = (H(w)+\rho I)^{-1/2}\nabla^2 f_i(w)^{1/2}$. Observe that $\mathbb{E}[X_i] = 0$, and set $X = \sum_i X_i$. To see that the conditions of the intrinsic dimension Matrix Bernstein inequality are met, note that X_i and $\mathbb{E}[X^2]$ satisfy

$$\begin{aligned}
\|X_i\| &= \frac{1}{b_h}\|Z_i\|^2 \leq \frac{\tau^\rho(H(w))}{b_h},\\
\mathbb{E}[X^2] &\preceq \frac{1}{b_h^2}\sum_i \mathbb{E}[\|z_i\|^2 z_iz_i^T] \preceq \frac{\tau^\rho(H(w))}{b_h}D^\rho := \mathcal{V}.
\end{aligned}$$

Moreover as $\rho \leq \lambda_1(H(w))$,

$$\text{tr}(\mathcal{V})/\|\mathcal{V}\| \leq 2d^\rho_{\text{eff}}(H(w)).$$

Thus, the intrinsic dimension Matrix Bernstein inequality [162, Theorem 7.3.1] implies

$$\mathbb{P}\left\{\|X\| \geq t\right\} \leq 8d^\rho_{\text{eff}}(H(w))\exp\left(-\frac{b_ht^2/2}{\tau^\rho(H(w))\,(1+t/3)}\right),$$

for all $t \geq \sqrt{\tau^\rho(H(w))/b_h} + \tau^\rho(H(w))/(3b_h)$. So, setting

$$t = \sqrt{\frac{4\tau^\rho(H(w))\log\left(\frac{8d^\rho_{\text{eff}}(H(w))}{\delta}\right)}{b_h}} + \frac{4}{3b_h}\tau^\rho(H(w))\log\left(\frac{8d^\rho_{\text{eff}}(H(w))}{\delta}\right),$$

and $b_h = \mathcal{O}\left(\frac{\tau^\rho(H(w))\log\left(\frac{d^\rho_{\text{eff}}(H(w))}{\delta}\right)}{\zeta^2}\right)$, it holds that

$$\mathbb{P}\left(\|X\| \leq \frac{\zeta}{1+\zeta}\right) \geq 1-\delta.$$

This last display immediately implies that

$$\left(1-\frac{\zeta}{1+\zeta}\right)H^\rho(w) \preceq H^\rho_S(w) \preceq \left(1+\frac{\zeta}{1+\zeta}\right)H^\rho(w),$$

which is equivalent to

$$\left(1+\frac{\zeta}{1+\zeta}\right)^{-1}H^\rho_S \preceq H^\rho \preceq \left(1-\frac{\zeta}{1+\zeta}\right)^{-1}H^\rho_S.$$

The desired claim now follows from the last display, upon observing that

$$1-\zeta \leq \left(1+\frac{\zeta}{1+\zeta}\right)^{-1} \leq \left(1-\frac{\zeta}{1+\zeta}\right)^{-1} = 1+\zeta.$$

□

4.8.6 Proof of Proposition 4.4.9 and Corollary 4.4.12

Proof of Proposition 4.4.9

Proof. Let $E = H_S - \hat{H}_S$, and note that $E \succeq 0$ by Lemma 2.2.1. Now by our hypothesis on r, it follows from Lemma 3.7.4, that

$$\mathbb{P}\left(\|E\| \leq \zeta\rho/4\right) \geq 1-\delta/2.$$

Using the decomposition $H^\rho_S = P + E$, we apply Weyl's inequalities to find

$$\begin{aligned}&\lambda_1(P^{-1/2}H^\rho_S P^{-1/2}) \leq \lambda_1\left(P^{-1/2}\hat{H}^\rho_S P^{-1/2}\right) + \lambda_1\left(P^{-1/2}EP^{-1/2}\right) = \\ &1+\|P^{-1/2}EP^{-1/2}\| \leq 1+\|P^{-1}\|\|E\| \leq 1+\frac{\|E\|}{\rho} \leq 1+\zeta/4.\end{aligned}$$

Moreover as $E \succeq 0$ we have $H^\rho_S \succeq P$, so conjugation yields $P^{-1/2}H^\rho_S P^{-1/2} \succeq I_p$. The preceding inequality immediately yields $\lambda_p(P^{-1/2}H^\rho_S P^{-1/2}) \geq 1$. Hence

$$1 \leq \lambda_p(P^{-1/2}H^\rho_S P^{-1/2}) \leq \lambda_1(P^{-1/2}H^\rho_S P^{-1/2}) \leq 1+\zeta/4.$$

As an immediate consequence of this last display, we obtain the Loewner ordering relation

$$P \preceq H_S^\rho \preceq (1+\zeta/4)P.$$

Now, Lemma 4.4.8 and a union bound implies that

$$\mathbb{P}\left(P \preceq H_S^\rho \preceq (1+\zeta/4)P,\ (1-\zeta/4)H_S^\rho \preceq H^\rho \preceq (1+\zeta/4)H_S^\rho\right) \geq 1-\delta.$$

Combining these relations and using that $(1+\zeta/4)^2 \leq 1+\zeta$ for $\zeta \in (0,1)$, we find

$$(1-\zeta)P \preceq H^\rho \preceq (1+\zeta)P.$$

To conclude, note that $(1+\rho/\mu)H^\rho \preceq H \preceq H^\rho$, which combined with the last display implies

$$(1-\zeta)\frac{1}{1+\rho/\mu}P \preceq H \preceq (1+\zeta)P.$$

□

4.8.7 Proof of Lemma 4.4.14

Proof. The function f is smooth and strongly convex, so it is quadratically regular. Consequently,

$$f(w) \geq f(w_\star) + \frac{\hat{\gamma}_\ell}{2}\|w-w_\star\|_{H(w_P)}^2.$$

Hence we have

$$f(w) - f(w_\star) \geq \frac{\gamma_\ell}{2}(1-\zeta)\frac{1}{1+\rho/\mu}\|w-w_\star\|_P^2,$$

where in the last inequality we have used the hypothesis that the conclusion of Proposition 4.4.9 holds. The claim now follows by recalling that $\hat{\gamma}_\ell = \frac{(1-\zeta)\mu}{\mu+\rho}\gamma_\ell$. □

4.8.8 Proof of Proposition 4.4.10

In this subsection we prove Proposition 4.4.10, which controls the variance of the preconditioned minibatch gradient. We start by proving the following more general result, from which Proposition 4.4.10 follows immediately.

Proposition 4.8.5 (Expected smoothness in the dual-norm)**.** *Suppose that each f_i is convex and satisfies*

$$f_i(w+h) \leq f_i(w) + \langle g_i(w), h\rangle + \frac{L_i}{2}\|h\|_{M_i}^2, \quad \forall w, h \in \mathbb{R}^p,$$

for some symmetric positive definite matrix M_i. Moreover, let f satisfy

$$f(w+h) \leq f(w) + \langle g(w), h\rangle + \frac{L}{2}\|h\|_M^2, \quad \forall w, h \in \mathbb{R}^p,$$

where $M = \frac{1}{n}\sum_{i=1}^n M_i$. Define $\tau(M) := \max_{1\leq i\leq n} \lambda_1\left(M^{-1/2}M_iM^{-1/2}\right)$. Further, suppose we construct the gradient sample $g_B(w)$ with batch-size b_g. Then for every $w \in \mathbb{R}^p$

$$\mathbb{E}_B\|g_B(w) - g_B(w')\|_{M^{-1}}^2 \leq 2\mathcal{L}_M(f(w) - f(w') - \langle g(w'), w - w'\rangle),$$

$$\mathbb{E}_B\|g_B(w)\|_{M^{-1}}^2 \leq 4\mathcal{L}_M(f(w) - f(w_\star)) + 2\sigma_M^2,$$

where

$$\mathcal{L}_M = \frac{n(b_g-1)}{b_g(n-1)}L + \tau(M)\frac{n-b_g}{b_g(n-1)}\text{max}_{1\leq i\leq n}L_i,$$

and

$$\sigma_M^2 = \frac{n-b_g}{b_g(n-1)}\frac{1}{n}\sum_{i=1}^n\|\nabla f_i(w_\star)\|_{M^{-1}}^2.$$

Proof. Introduce the change of variable $w = M^{-1/2}z$, so that $f_i(w) = f_i(M^{-1/2}z) = f_{i,M}(z)$ and $f(w) = f(M^{-1/2}z) = f_M(z)$. Then by our hypotheses on the f_i's, f, and the definition of $\tau(M)$, we have that for all $z, h \in \mathbb{R}^p$

$$f_{i,M}(z+h) \leq f_{i,M}(z) + \langle g_{i,M}(z), h\rangle + \frac{\tau(M)L_i}{2}\|h\|^2,$$

$$f_M(z+h) \leq f_M(z) + \langle g_M(z), h\rangle + \frac{L}{2}\|h\|^2.$$

Consequently, Proposition 3.8 of [72] implies

$$\mathbb{E}\|g_{B,M}(z) - g_{B,M}(z')\|^2 \leq 2\mathcal{L}_M\left(f_M(z) - f_M(z) - \langle g_M(z'), z - z'\rangle\right),$$

$$\mathbb{E}\|g_{B,M}(z_\star)\|^2 = \frac{n-b_g}{b_g(n-1)}\frac{1}{n}\sum_{i=1}^n\|g_{M,i}(z_\star)\|^2,$$

where $\mathcal{L}_M = \frac{n(b_g-1)}{b_g(n-1)}L + \tau(M)\frac{n-b_g}{b_g(n-1)}\text{max}_{1\leq i\leq n}L_i$. Invoking Lemma 4.8.1, the above displays become

$$\mathbb{E}\|g_B(w) - g_B(w')\|_{M^{-1}}^2 \leq 2\mathcal{L}_M\left(f(w) - f(w') - \langle g(w'), w - w'\rangle\right),$$

$$\mathbb{E}\|g_B(w_\star)\|_{M^{-1}}^2 = \frac{n-b_g}{b_g(n-1)}\frac{1}{n}\sum_{i=1}^n\|g_i(w_\star)\|_{M^{-1}}^2 = \sigma^2.$$

The last portion of the desired claim now follows by combining the preceding displays with $w' = w_\star$, and the identity $\mathbb{E}\|a\|_{M^{-1}}^2 \leq 2\mathbb{E}\|a-b\|_{M^{-1}}^2 + 2\mathbb{E}\|b\|_{M^{-1}}^2$. □

Proof of Proposition 4.4.10

Proof. Set $M_i = P$ so that $M = H^\rho(w_P)$. Now, by the assumption that $H(w_P) \preceq (1+\zeta)P$ and item 1 of Lemma 4.4.4 it holds that each f_i is smooth with respect to M_i with $L_i = (1+\zeta)\tau^\rho H(w_P)\gamma_{i,u}^\rho$, while f is smooth with respect to M with $L = (1+\zeta)\gamma_u^\rho$. Noting that $\tau(M) = 1$, the claim follows from Proposition 4.8.5. □

4.9 Lower bound on condition number in Table 4.2

Recall the condition number κ is given by

$$\kappa = \frac{\sup_{w\in\mathbb{R}^p} \lambda_1(H(w))}{\inf_{w\in\mathbb{R}^p} \lambda_p(H(w))}.$$

Consequently when f is a the least-squares or logistic loss with data matrix A, and l^2-regularization $\mu \geq 0$, it holds for any $r \leq p$ that

$$\kappa \geq \frac{\lambda_1(H(0))}{\lambda_p(H(0))} \geq \frac{\lambda_1(H(0))}{\lambda_r(H(0))} = \frac{\sigma_1^2(A)/n + \mu}{\sigma_r^2(A)/n + \mu}.$$

Hence κ is lower bounded by $(\sigma_1^2(A)/n + \mu)/(\sigma_r^2(A)/n + \mu)$. Table 4.2 gives the numerical value for this lower bound for $r = 100$.

4.10 Experimental details

Here we provide more details regarding the experiments in Section 4.5.

Regularization For convex problems (Sections 4.5.1 to 4.5.4 and 4.11.4), we set the l_2-regularization to $10^{-2}/n_{\text{tr}}$, where n_{tr} is the size of the training set. For deep learning (Section 4.5.5), we use no regularization or weight decay, as the experiments are proof-of-concept.

Ridge regression datasets The ridge regression experiments are run on the datasets described in the main text. E2006-tfidf and YearPredictionMSD's rows are normalized to have unit-norm, while we standardize the features of yolanda. For YearPredictionMSD we use a ReLU random features transformation that gives us 4367 features in total. For yolanda we use a random features transformation with bandwidth 1 that gives us 1000 features in total, and perform a random 80-20 split to form a training and test set. In Table 4.4, we provide the dimensions of the datasets, where n_{tr} is the number of training samples, n_{test} is the number of testing samples, and p is the number of features.

Table 4.4: **Dimensions of ridge regression datasets.**

Dataset	n_{tr}	n_{test}	p
E2006-tfidf	16087	3308	150360
YearPredictionMSD	463715	51630	4367
yolanda	320000	80000	1000

Logistic regression datasets The logistic regression experiments are run on the datasets described in the main text. All datasets' rows are normalized so that they have unit norm. For ijcnn1 and susy we use a random features transformation with bandwidth 1 that gives us 2500 and 1000 features, respectively. For real-sim, we use a random 80-20 split to form a training and test set. For HIGGS, we repeatedly apply a random features transformation with bandwidth 1 to obtain 10000 features, as described in Section 4.5.4. In Table 4.5, we provide the dimensions of the datasets, where n_{tr} is the number of training samples, n_{test} is the number of testing samples, and p is the number of features.

Table 4.5: **Dimensions of logistic regression datasets.**

Dataset	n_{tr}	n_{test}	p
ijcnn1	49990	91701	2500
susy	4500000	500000	1000
real-sim	57847	14462	20958
HIGGS	10500000	500000	10000

Deep learning datasets The deep experiments are run on the datasets described in the main text. We download the datasets using the OpenML-Python connector [54]. Each dataset is standardized to have zero mean and unit variance, and the statistics for standardization are calculated using only the training split. In Table 4.6, we provide the dimensions of the datasets, where n is the number of samples (before splitting into training, validation, and test sets), p is the number of features, and ID is the unique identifier of the dataset on OpenML.

Dataset augmentation for scaling (Sections 4.5.2 and 4.5.3) We perform data augmentation before any additional preprocessing steps (e.g., normalization, standardization, random features). To increase the samples by a factor of k, we duplicate the dataset a total of $k - 1$ times. For each duplicate, we generate a random Gaussian matrix, where each element has variance 0.02. For sparse datasets, this Gaussian matrix is generated to have the same number of nonzeros as the original dataset. Each duplicate and Gaussian matrix is summed; the resulting sums are stacked to form the augmented dataset.

Table 4.6: **Dimensions of deep learning datasets.**

Dataset	n	p	ID
Fashion-MNIST	70000	784	40996
Devnagari-Script	92000	1024	40923
volkert	58310	180	41166

Random seeds In Sections 4.5.1, 4.11.1 and 4.11.4 we run all experiments with 10 random seeds, with the exception of susy, for which we use 3 random seeds.

We use the same number of random seeds in Section 4.5.2, except for the scaling experiments. For the scaling experiments, we only use 3 random seeds.

In Section 4.5.4 we use only 1 random seed due to the sheer size of the problem.

In Section 4.5.5 we use only 1 random seed for each learning rate given by random search. However, we use 10 random seeds to generate the results with the tuned learning rate.

Additional hyperparameters (Sections 4.5.1 and 4.5.4) For SVRG we perform a full gradient computation at every epoch.

For L-Katyusha, we initialize the update probability $p_{\mathrm{upd}} = b_g/n_{\mathrm{tr}}$ to ensure the average number of iterations between full gradient computations is equal to one epoch. We follow [93] and set μ equal to the l_2-regularization parameter, $\sigma = \frac{\mu}{L}, \theta_1 = \min\{\sqrt{2\sigma n_{\mathrm{tr}}/3}, \frac{1}{2}\}$, and $\theta_2 = \frac{1}{2}$.

All algorithms use a batch size of 256 for computing stochastic gradients, except on the HIGGS dataset. For the HIGGS dataset, SGD, SAGA, and SketchySGD are all run with a batch size of 4096.

Additional hyperparameters (Section 4.5.2) For SLBFGS we perform a full gradient computation at every epoch. Furthermore, we update the inverse Hessian approximation every epoch and set the Hessian batch size to $\sqrt{n_{\mathrm{tr}}}$, which matches the Hessian batch size hyperparameter in SketchySGD. In addition, we follow [115] and set the memory size of SLFBGS to 10. We use a batch size of 256 for computing stochastic gradients.

We use the defaults for L-BFGS provided in the SciPy implementation, only tuning the "factr" parameter when necessary to avoid early termination.

For RSN, we grid search the sketch size in $\{250, 500, 750, 1000\}$.

For Newton Sketch, we grid search the sketch size in $\{2 \cdot 10^{-3} \cdot \min(n_{\mathrm{tr}}, p) \cdot 50^{k/9} : k = 0, 1, \ldots, 9\}$.

For RSN and Newton Sketch, we follow the original publications' suggestions [71, 131] for setting the line search parameters.

Additional hyperparameters (Section 4.5.3) We grid search the sketch size in $\{2 \cdot 10^{-3} \cdot \min(n_{\mathrm{tr}}, p) \cdot 50^{k/9} : k = 0, 1, \ldots, 9\}$ for all three of NyströmPCG, GaussPCG, and SparsePCG.

Additional hyperparameters (Section 4.5.5) For all of the competitor methods (except Shampoo), we use the default hyperparameters. For Shampoo, we modify the preconditioner update frequency to occur every epoch, similar to SketchySGD. For SketchySGD, we set the momentum parameter β to 0.9, just as in Adam. We compute stochastic gradients using a batch size of 128. For learning rate scheduling, we use cosine annealing with restarts. For the restarts, we use an initial budget of 15 epochs, with a budget multiplier of 2.

Default hyperparameters for SAGA/SVRG/L-Katyusha The theoretical analysis of SVRG, SAGA, and L-Katyusha all yield recommended learning rates that lead to linear convergence. In practical implementations such as scikit-learn [130], these recommendations are often taken as the default learning rate. For SAGA, we follow the scikit-learn implementation, which uses the following learning rate:

$$\eta = \max\left\{\frac{1}{3L}, \frac{1}{2\left(L + n_{\mathrm{tr}}\mu\right)}\right\},$$

where L is the smoothness constant of f and μ is the strong convexity constant. The theoretical analysis of SVRG suggests a step-size of $\eta = \frac{1}{10\mathcal{L}}$, where $\mathcal{L}$ is the expected-smoothness constant. We have found this setting to pessimistic relative to the SAGA default, so we use the same default for SVRG as we do for SAGA. For L-Katyusha the hyperparameters θ_1 and θ_2 are controlled by how we specify L^{-1}, the reciprocal of the smoothness constant. Thus, the default hyperparameters for all methods are controlled by how L is specified.

Now, standard computations show that the smoothness constants for least-squares and logistic regression satisfy

$$L_{\text{least-squares}} \leq \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \|a_i\|^2,$$

$$L_{\text{logistic}} \leq \frac{1}{4n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \|a_i\|^2.$$

The scikit-learn software package uses the preceding upper-bounds in place of L to set η in their implementation of SAGA. We adopt this convention for setting the hyperparameters of SAGA, SVRG and L-Katyusha. We display the defaults for the three methods in Table 4.7.

Table 4.7: **Default hyperparameters for SVRG/SAGA/L-Katyusha.**

Method\Dataset	E2006-tfidf	YearPredictionMSD	yolanda	ijcnn1	real-sim	susy	HIGGS
SVRG/SAGA	$4.95 \cdot 10^{-1}$	$1.01 \cdot 10^{0}$	$4.96 \cdot 10^{-1}$	$1.91 \cdot 10^{0}$	$1.93 \cdot 10^{0}$	$1.87 \cdot 10^{0}$	$1.93 \cdot 10^{0}$
L-Katyusha	$1.00 \cdot 10^{0}$	$4.85 \cdot 10^{-1}$	$9.98 \cdot 10^{-1}$	$2.52 \cdot 10^{-1}$	$2.50 \cdot 10^{-1}$	$2.57 \cdot 10^{-1}$	N/A

Grid search parameters (Sections 4.5.1 and 4.5.2) We choose the grid search ranges for SVRG, SAGA, and L-Katyusha to (approximately) include the default hyperparameters across the tested datasets (Table 4.7). For ridge regression, we set $[10^{-3}, 10^{2}]$ as the search range for the learning rate in SVRG and SAGA, and $[10^{-2}, 10^{0}]$ as the search range for the smoothness parameter L in L-Katyusha. Similar to SVRG and SAGA, we set $[10^{-3}, 10^{2}]$ as the search range for SGD. For SLBFGS, we set the search range to be $[10^{-5}, 10^{0}]$ in order to have the same log-width as the search range for SGD, SVRG, and SAGA. In logistic regression, the search ranges for SGD/SVRG/SAGA, L-Katyusha, and SLBFGS become $[4 \cdot 10^{-3}, 4 \cdot 10^{2}], [2.5 \cdot 10^{-3}, 2.5 \cdot 10^{-1}]$, and $[4 \cdot 10^{-5}, 4 \cdot 10^{0}]$, respectively. The grid corresponding to each range samples 10 equally spaced values in log space. The tuned hyperparmeters for all methods across each dataset are presented in Table 4.8.

Table 4.8: **Tuned hyperparameters for competitor methods.**

Method\Dataset	E2006-tfidf	YearPredictionMSD-rf	yolanda-rf	ijcnn1-rf	real-sim	susy-rf
SGD	$5.99 \cdot 10^{-1}$	$2.15 \cdot 10^{0}$	$5.99 \cdot 10^{-1}$	$8.62 \cdot 10^{0}$	$4 \cdot 10^{2}$	$8.62 \cdot 10^{0}$
SVRG	$5.99 \cdot 10^{-1}$	$2.15 \cdot 10^{0}$	$2.15 \cdot 10^{0}$	$8.62 \cdot 10^{0}$	$4 \cdot 10^{2}$	$8.62 \cdot 10^{0}$
SAGA	$5.99 \cdot 10^{-1}$	$2.15 \cdot 10^{0}$	$2.15 \cdot 10^{0}$	$8.62 \cdot 10^{0}$	$4 \cdot 10^{2}$	$8.62 \cdot 10^{0}$
L-Katyusha	$2.15 \cdot 10^{-1}$	$1.29 \cdot 10^{-1}$	$2.15 \cdot 10^{-1}$	$1.94 \cdot 10^{-2}$	$2.5 \cdot 10^{-3}$	$3.32 \cdot 10^{-2}$
SLBFGS	$7.74 \cdot 10^{-2}$	$2.15 \cdot 10^{-2}$	$1.67 \cdot 10^{-3}$	$1.11 \cdot 10^{0}$	$8.62 \cdot 10^{-2}$	$8.62 \cdot 10^{-2}$

Grid search parameters (Section 4.5.4) Instead of using a search range of $[4 \cdot 10^{-3}, 4 \cdot 10^{2}]$ for SGD/SAGA, we narrow the range to $[4 \cdot 10^{-2}, 4 \cdot 10^{1}]$ and sample 4 equally spaced values in log space. The reason for reducing the search range and grid size is to reduce the total computational cost of running the experiments on the HIGGS dataset. Furthermore, we find that $4 \cdot 10^{0}$ is the best learning rate for HIGGS, while $4 \cdot 10^{1}$ leads to non-convergent behavior, meaning these search ranges are appropriate.

Random search parameters (Section 4.5.5) We tune the learning rate for each optimizer using 30 random search trials with log-uniform sampling in the range $[10^{-3}, 10^{-1}]$. The tuning is performed with Optuna [2].

4.11 Additional experimental results and figures

4.11.1 Sensitivity experiments

In this section, we investigate the sensitivity of SketchySGD to the rank hyperparameter r (Section 4.11.2) and update frequency hyperparameter u (Section 4.11.3). In the first set of sensitivity experiments, we select ranks $r \in \{1, 2, 5, 10, 20, 50\}$ while holding the update frequency fixed at $u = \left\lceil \frac{n_{\mathrm{tr}}}{b_g} \right\rceil$ (1 epoch)[6]. In the second set of sensitivity experiments, we select update frequencies $u \in \left\{0.5 \left\lceil \frac{n_{\mathrm{tr}}}{b_g} \right\rceil, \left\lceil \frac{n_{\mathrm{tr}}}{b_g} \right\rceil, 2 \left\lceil \frac{n_{\mathrm{tr}}}{b_g} \right\rceil, 5 \left\lceil \frac{n_{\mathrm{tr}}}{b_g} \right\rceil, \infty \right\}$, while holding the rank fixed at $r = 10$. We use the

[6] If we set $u = \infty$ in ridge regression, which fixes the preconditioner throughout the run of SketchySGD, the potential gain from a larger rank r may not be realized due to a poor initial Hessian approximation.

datasets from Table 4.2. Each curve is the median performance of a given (r, u) pair across 10 random seeds (except for susy, which uses 3 seeds), run for 40 epochs.

4.11.2 Effects of changing the rank

Looking at Figure 4.12, we see two distinct patterns: either increasing the rank has no noticeable impact on performance (E2006-tfidf, real-sim), or increasing the rank leads to faster convergence to a ball of noise around the optimum (YearPredictionMSD-rf). We empirically observe that these patterns are related to the spectrum of each dataset, as shown in Figure 4.13. For example, the spectrum of E2006-tfidf is highly concentrated in the first singular value, and decays rapidly, increased rank does not improve convergence. On the other hand, the spectrum of YearPredictionMSD-rf is not as concentrated in the first singular value, but still decays rapidly, so convergence improves as we increase the rank from $r = 1$ to $r = 10$, after which performance no longer improves, in fact it slightly degrades. The spectrum of real-sim decays quite slowly in comparison to E2006-tfidf or YearPredictionMSD-rf, so increasing the rank up to 50 does not capture enough of the spectrum to improve convergence. One downside in increasing the rank is that the quantity $\eta_{\text{SketchySGD}}$ (4.3) can become large, leading to SketchySGD taking a larger step size. As a result, SketchySGD oscillates more about the optimum, as seen in YearPredictionMSD-rf (Figure 4.12). Last, Figure 4.12 shows $r = 10$ delivers great performance across all datasets, supporting its position as the recommended default rank. Rank sensitivity plots for yolanda-rf, ijcnn1-rf, and susy-rf appear in Section 4.11.1.

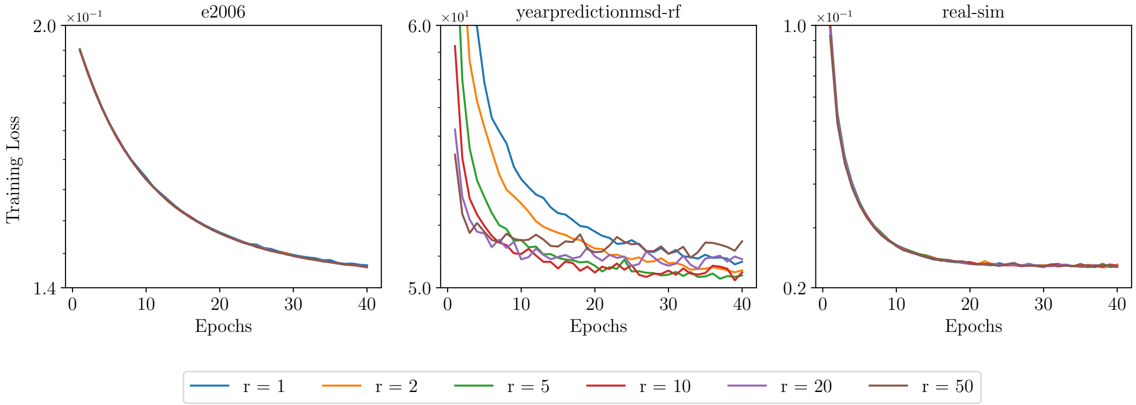


Figure 4.12: Sensitivity of SketchySGD to rank r.

4.11.3 Effects of changing the update frequency

In this section, we display results only for logistic regression (Figure 4.14), since there is no benefit to updating the preconditioner for a quadratic problem such as ridge regression (Section 4.11.1): the Hessian in ridge regression is constant for all $w \in \mathbb{R}^p$. The impact of the update frequency depends

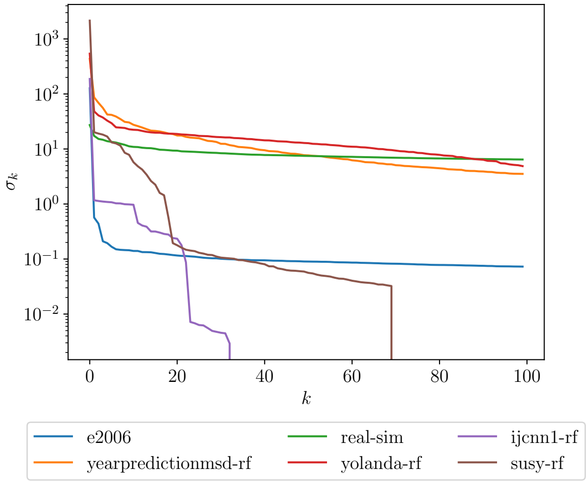


Figure 4.13: Top 100 singular values of datasets after preprocessing.

on the spectrum of each dataset. The spectra of ijcnn1-rf and susy-rf are highly concentrated in the top $r = 10$ singular values and decay rapidly (Figure 4.13), so even the initial preconditioner approximates the curvature of the loss well throughout optimization. On the other hand, the spectrum of real-sim decays quite slowly, and the initial preconditioner does not capture most of the curvature information in the Hessian. Hence for real-sim it is beneficial to update the preconditioner, however only infrequent updating is required, as an update frequency of 5 epochs yields almost identical performance to updating every half epoch. So, increasing the update frequency of the preconditioner past a certain threshold does not improve performance, it just increases the computational cost of the algorithm. Last, $u = \lceil \frac{n_{\rm tr}}{b_g} \rceil$ exhibits consistent excellent performance across all datasets, supporting the recommendation that it be the default update frequency.

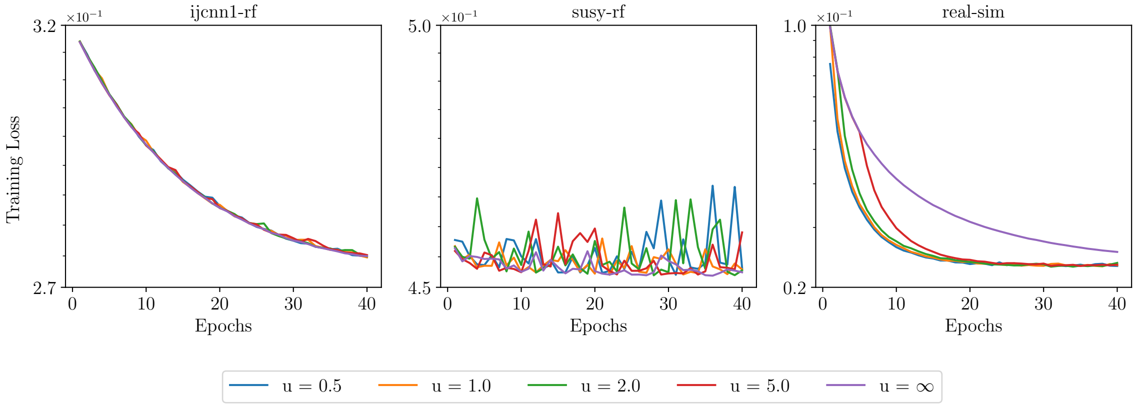


Figure 4.14: Sensitivity of SketchySGD to update frequency u.

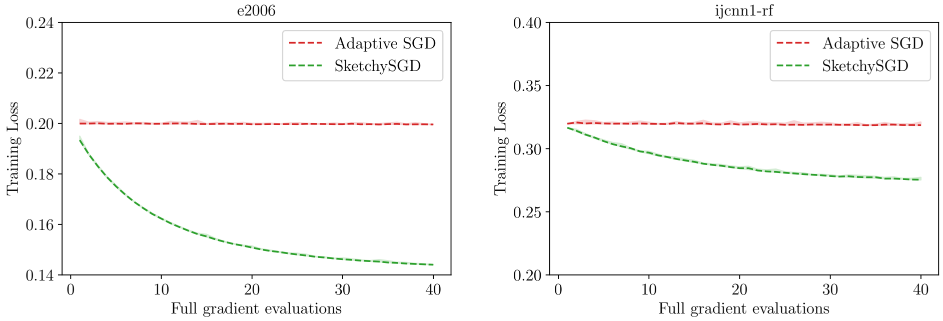


Figure 4.15: Adaptive SGD vs. SketchySGD. Adaptive SGD performs much worse then SketchySGD on these two problems, despite employing the same learning rate strategy as SketchySGD. Thus, SketchySGD's improved performance over SGD comes from incorporating preconditioning, and not from how it sets the learning rate.

4.11.4 SketchySGD default learning rate ablation

It is natural to ask how much of SketchySGD's improved performance relative to SGD stems from preconditioning. Indeed, it may be the case that SketchySGD's gains arise from how it sets the learning rate, and not from using approximate second-order information. To test this, we employ SGD with the same learning rate selection strategy as SketchySGD, but with the preconditioned minibatch Hessian replaced by the minibatch Hessian. We refer to this algorithm as *Adaptive SGD* (AdaSGD).

Figure 4.15 shows the results of AdaSGD and SketchySGD on the E2006-tfidf and ijcnn1-rf datasets. Adaptive SGD performs significantly worse then SketchySGD on these two problems, which shows that SketchySGD's superior performance over SGD is due to employing preconditioning. This result is not too surprising. To see why, let us consider the case of the least-squares loss. In this setting, if the subsampled Hessian is representative of the true Hessian, then $\eta_{\text{AdaSGD}} \approx \mathcal{O}(1/L)$. Hence when the problem is ill-conditioned, the resulting stepsize will result in poor progress, which is precisely what is observed in Figure 4.15.

4.11.5 SketchySGD improves the conditioning of the Hessian

In Section 4.5.1, we showed that SketchySGD generally converges faster than other first-order stochastic optimization methods. In this section, we examine the conditioning of the Hessian before and after preconditioning to understand why SketchySGD displays these improvements.

Recall from Section 4.1.1 that SketchySGD is equivalent to performing SGD in a preconditioned space induced by $P_j = \hat{H}_{S_j} + \rho I$. Within this preconditioned space, the Hessian is given by $P_j^{-1/2} H P_j^{-1/2}$, where H is the Hessian in the original space. Thus, if $\kappa(P_j^{-1/2} H P_j^{-1/2}) \ll \kappa(H)$, we know that SketchySGD is improving the conditioning of the Hessian, which allows SketchySGD to converge faster.

Figures 4.16 and 4.17 display the top 500 eigenvalues (normalized by the largest eigenvalue) of the Hessian H and the preconditioned Hessian $P_j^{-1/2} H P_j^{-1/2}$ at the initialization of SketchySGD (Nyström) for both logistic and ridge regression. With the exception of real-sim, SketchySGD (Nyström) improves the conditioning of the Hessian by several orders of magnitude. This improved conditioning aligns with the improved convergence that is observed on the ijcnn1-rf, susy-rf, E2006-tfidf, YearPredictionMSD-rf, and yolanda-rf datasets in Section 4.5.1.

4.12 Scaling experiments

4.12.1 Second-order

For larger datasets, we expect the performance gap between SketchySGD and the selected second-order methods to grow even larger, since these methods require full-gradient computations. We increase the number of samples for each dataset in Table 4.2 (with the exception of susy-rf) by a factor of 3 using data augmentation with Gaussian random noise, i.e., a dataset of size $n_{\mathrm{tr}} \times p$ now has size $3n_{\mathrm{tr}} \times p$. The results are shown in Figures 4.18 and 4.19. When looking at performance with respect to wall-clock time, SketchySGD is outperformed less often by the second-order methods; it is only outperformed by SLBFGS (before it diverges) on ijcnn1-rf and RSN on yolanda-rf. Again, SketchySGD performs much better than the competition on YearPredictionMSD-rf, which is a larger, dense dataset.

4.12.2 PCG

The main costs of PCG are generally in (i) computing the preconditioner and (ii) performing matrix-vector products with the data matrix. For larger datasets, we expect both of these costs to increase, which should close the performance gap between SketchySGD and PCG. We increase the number of samples for each ridge regression dataset by a factor of 3 as in Section 4.5.2; the results on these augmented datasets are presented in Figure 4.20. We see that the performance gap closes slightly — on E2006-tfidf, SketchySGD now performs comparably to JacobiPCG. In addition, the PCG methods now take significantly more wall-clock time to reach the training loss attained by SketchySGD.

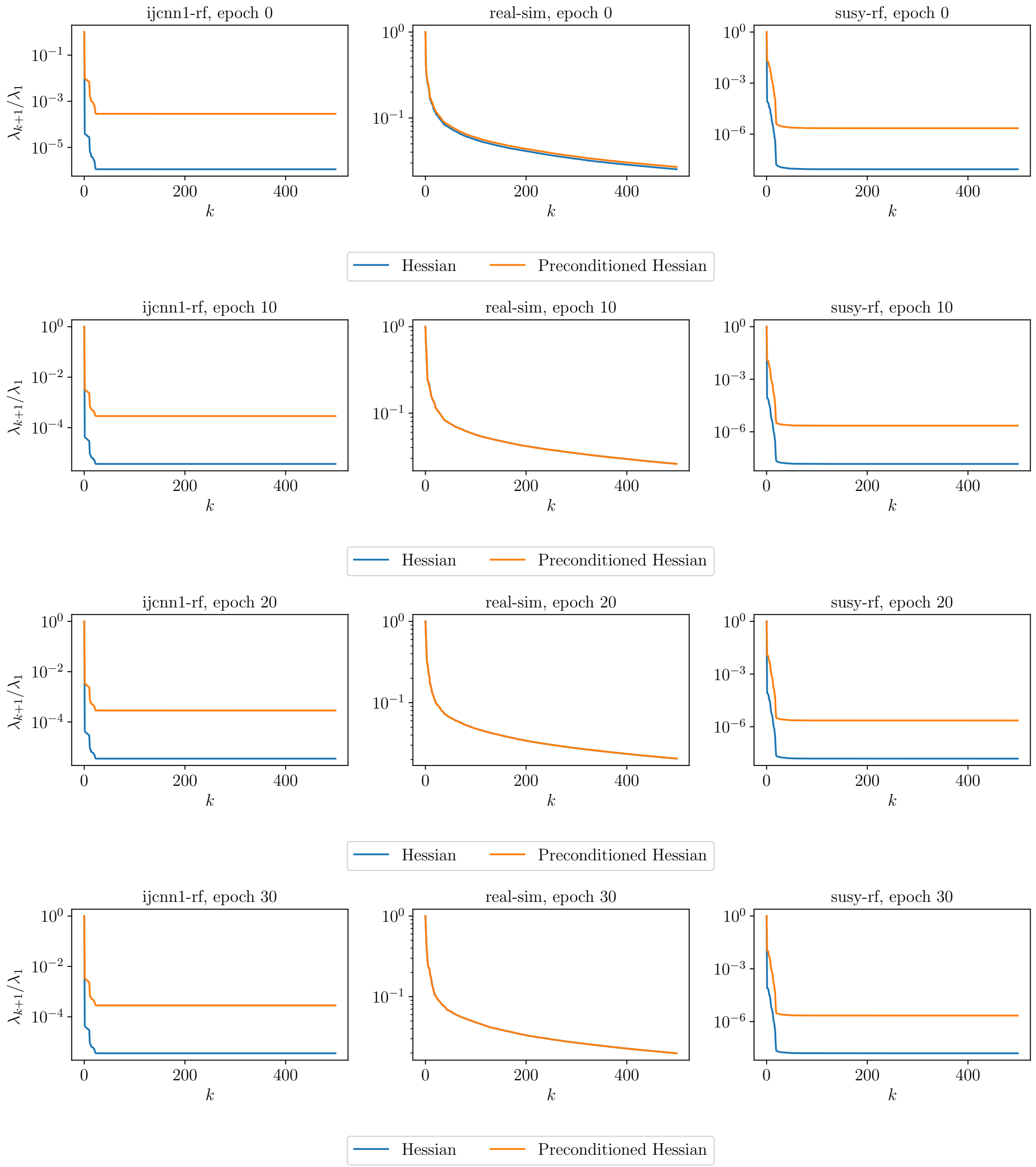


Figure 4.16: Spectrum of the Hessian at epochs $0, 10, 20, 30$ before and after preconditioning in l_2-regularized logistic regression.

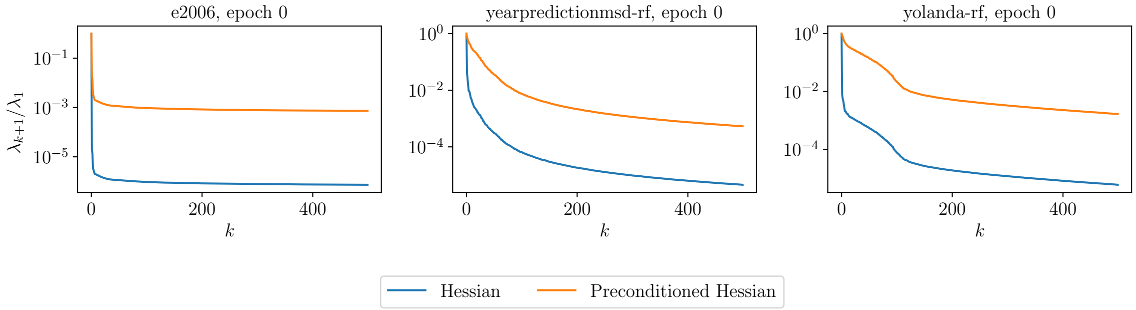


Figure 4.17: Normalized spectrum of the Hessian before and after preconditioning in ridge regression.

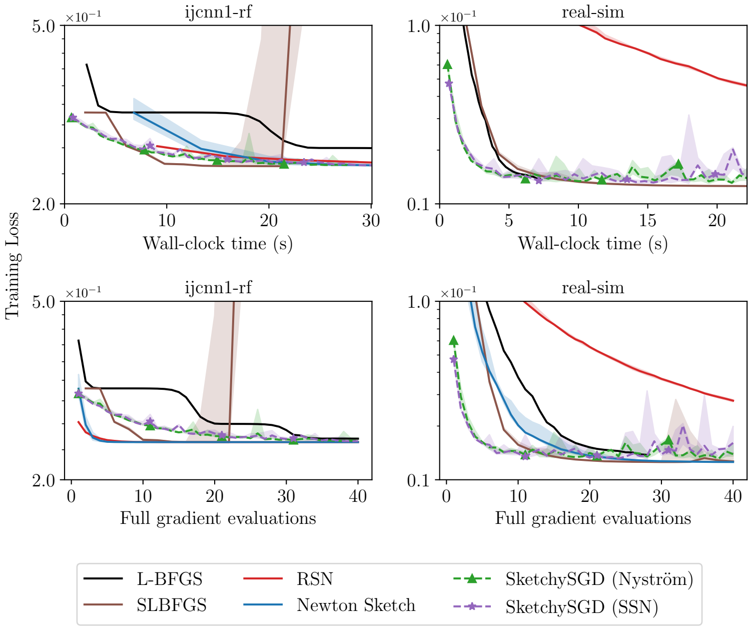


Figure 4.18: Comparisons to second-order methods (L-BFGS, SLBFGS, RSN, Newton Sketch) on l_2-regularized logistic regression with augmented datasets.

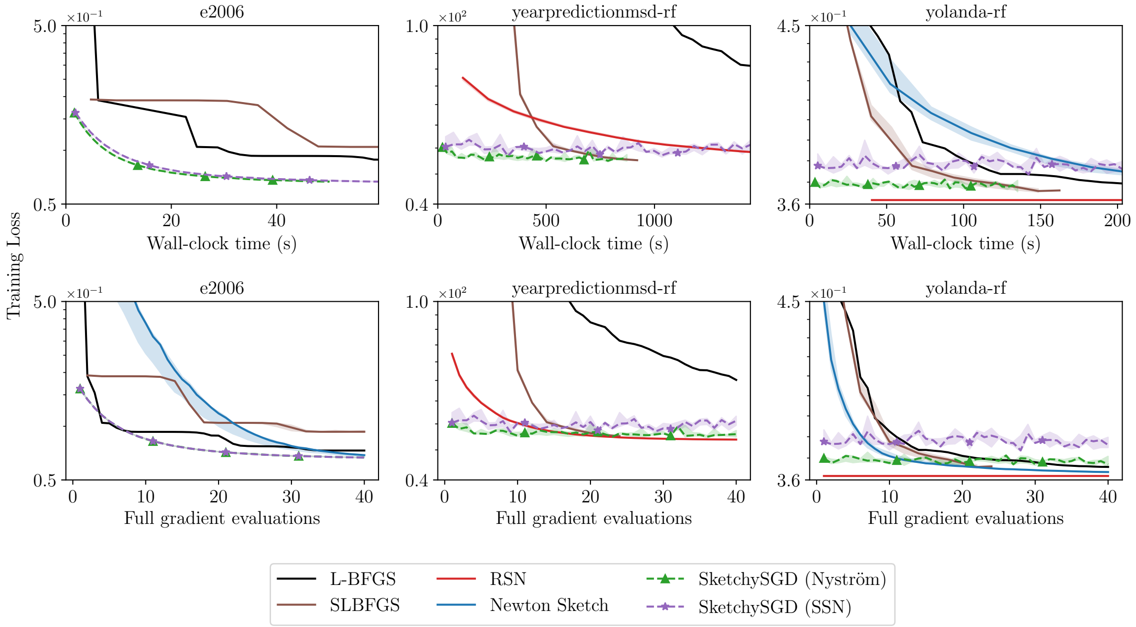


Figure 4.19: Comparisons to second-order methods (L-BFGS, SLBFGS, RSN, Newton Sketch) on ridge regression with augmented datasets.

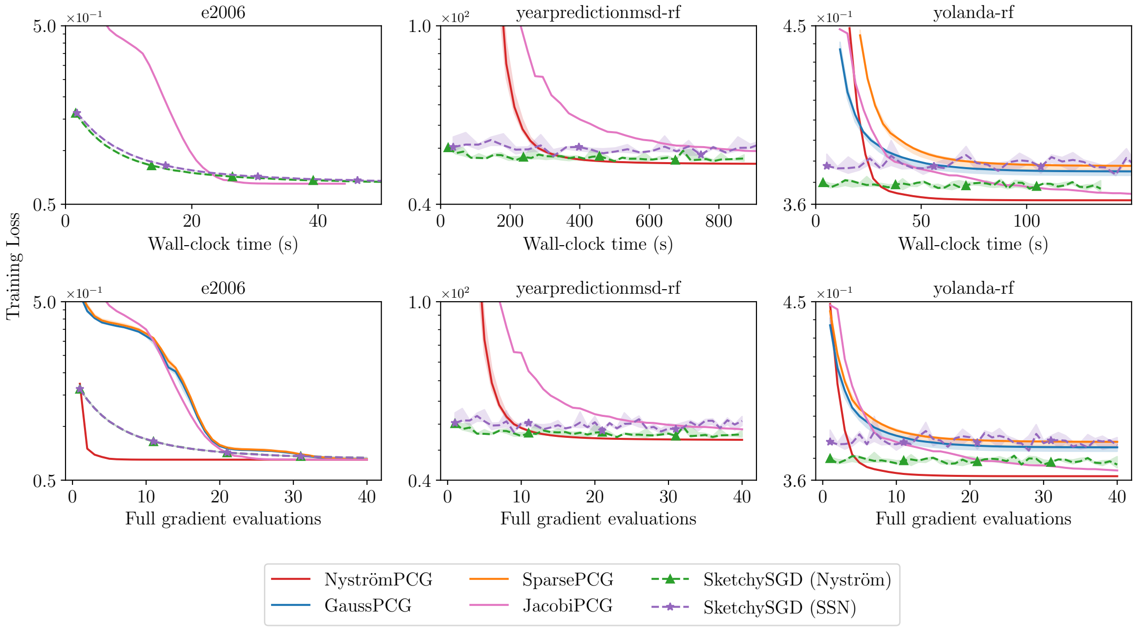


Figure 4.20: Comparisons to PCG (Jacobi, Nyström, sketch-and-precondition w/ Gaussian and sparse embeddings) on ridge regression with augmented datasets.

Chapter 5

Conclusions

5.1 Summary

This thesis introduces three novel algorithms for large-scale optimization: (i) Nyström PCG, (ii) NysADMM, and (iii) SketchySGD. The unifying principle underlying these methods is the observation that data and Hessian matrices in large-scale optimization often exhibit approximate low-rank structure, enabling efficient approximation through low-rank matrices computed via Randomized Numerical Linear Algebra (RandNLA). From these low-rank approximations, we construct preconditioners that bring the dominant eigenvalues closer to the level of the well-conditioned tail eigenvalues. By leveraging this technique, Nyström PCG, NysADMM, and SketchySGD achieve significant improvements over existing methods, in some cases yielding speed-ups of up to 58 times faster than baseline approaches.

We anticipate these algorithms will prove valuable to practitioners seeking more efficient solutions to large-scale optimization problems, while also providing a pathway to better address ill-conditioning in this domain. Notably, Nyström PCG has been incorporated into the recent RandLAPACK library [117], a highly optimized set of routines poised to serve as a fundamental backbone for randomized linear algebra, analogous to LAPACK's role in deterministic numerical linear algebra [7].

Looking ahead, we briefly discuss extensions of the ideas developed in this work and potential directions for future research, underscoring the broader impact and ongoing relevance of this contribution to the field of large-scale optimization.

5.2 Extensions

The ideas developed in this thesis have several extensions, which have led to further improvements in large-scale optimization, that we now discuss.

For Nyström PCG in large-scale kernel kernel ridge regression, [42] employs a randomized Nyström preconditioner constructed via column sampling, utilizing the Randomized Pivoted Cholesky algorithm introduced in [31]. This approach reduces the preconditioner computation cost to $\mathcal{O}(n\ell^2)$ while maintaining similar theoretical guarantees.

The NysADMM paper leaves open the question of convergence for non-quadratic objectives, as well as what the explict rate of convergence of the algorithm is. The follow-up paper [59] resolves these questions, and establishes explicit convergence rates under standard regularity assumptions for a more general version of the model problem in (3.1), which now includes linear equality constraints. Notably, these results demonstrate that the approximations in NysADMM do not compromise the overall global convergence rate. Building on this, [41] extends NysADMM to develop GeNIOS, a highly optimized solver for general convex composite optimization problems with efficient proximal and projection oracles. Extensive numerical experiments demonstrate GeNIOS outperforms state-of-the-art open-source ADMM solvers like COSMO [62] and OSQP [154], as well as the commercial interior-point based solver MOSEK [6].

SketchySGD is limited to modest accuracy solutions due to its use of stochastic gradients, [56] addresses this limitation in smooth and strongly convex settings. By combining SketchySGD principles with variance-reduced stochastic gradient algorithms like SVRG [85], SAGA [37], and Katyusha [4], they achieve global linear convergence. Moreover, they demonstrate local linear convergence independent of the condition number, highlighting the benefits of preconditioning.

For massive-scale kernel ridge regression where $n > 10^6$, PCG becomes prohibitive due to the $\mathcal{O}(n^2)$ cost of computing a matvec and the $\mathcal{O}(n\ell)$ storage required for the preconditioner. Inspired by SketchySGD, the works [139, 140] develop approximate versions of sketch-and-project based on Nyström sketch-and-solve. These methods offer strong theoretical guarantees, outperform state-of-the-art competitors, and can scale to datasets with over 10^8 training points.

5.3 Directions for future research

We conclude with a brief discussion of promising avenues for future work.

This thesis has developed scalable preconditioned algorithms that empirically improve convergence and provably enhance it in certain special cases, such as when the objective is quadratic. An intriguing direction for future research would be to develop similar algorithms that also improve the global rate of convergence.

One possible approach for improving the global convergence of SketchySGD and its variance-reduced variants [56], would be to change the base algorithm from Newton's method to alternatives such the Hybrid Proximal Extragradient Algorithm (HPE). To obtain the best possible rate, it is critical to incorporate acceleration into a stochastic HPE framework. A key challenge would be determining how to adapt/generalize techniques such as Monteiro-Svaiter acceleration [28, 114] to the stochastic setting. Recent work by [83] has shown an improved global convergence rate relative to Newton's method with linesearch, when using full gradients with a stochastic Hessian approximation in the HPE algorithm, suggesting a promising direction for exploration.

An alternative route to establishing better global convergence rates, is to focus on more structured function classes, as in [43, 82, 87], which develop fast second-order algorithms for quasi self-concordant functions [12]. Applying the techniques developed in this thesis to obtain scalable stochastic analogues of these algorithms that achieve improved global rates presents another interesting research direction.

A direction completely different from developing preconditioned algorithms that enjoy improved global rates, is to develop more memory efficient preconditioned algorithms for deep learning. At present the most popular deep learning optimizer is Adam, which uses momentum and a diagonal preconditioner based on gradient whitening. However, recent work [88] has shown when properly tuned, Shampoo can outperform Adam, demonstrating the benefit of using more sophisticated preconditioners than diagonal ones. Unfortunately, Shampoo's high memory requirements make it prohibitively expensive for the large models used in production today [51]. An exciting challenge would be to extend the techniques developed in this thesis to create a more memory-efficient version of Shampoo while maintaining its performance advantages.

Bibliography

[1] Alekh Agarwal, Sahand Negahban, and Martin J Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40(5):2452–2482, 2012.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[3] Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, 2015.

[4] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018.

[5] Idan Amir, Yair Carmon, Tomer Koren, and Roi Livni. Never go full batch (in stochastic convex optimization). *Advances in Neural Information Processing Systems*, 34:25033–25043, 2021.

[6] Erling D Andersen and Knud D Andersen. The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, pages 197–232. Springer.

[7] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, and Danny Sorensen. *LAPACK users' guide*. SIAM, 1999.

[8] Yossi Arjevani, Ohad Shamir, and Ron Shiff. Oracle complexity of second-order methods for smooth convex optimization. *Mathematical Programming*, 178:327–360, 2019.

[9] Haim Avron, Kenneth L Clarkson, and David P Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1116–1138, 2017.

[10] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

[11] Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48:499–523, 1986.

[12] Francis Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4:384 – 414, 2010.

[13] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, 2013.

[14] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1), 2014.

[15] A. S. Bandeira, K. Scheinberg, and L. N. Vicente. Convergence of trust-region methods based on probabilistic models. *SIAM Journal on Optimization*, 24(3):1238–1264, 2014.

[16] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[17] Albert S Berahas, Jorge Nocedal, and Martin Takác. A multi-batch l-bfgs method for machine learning. *Advances in Neural Information Processing Systems*, 29, 2016.

[18] Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

[19] Jose Blanchet, Coralia Cartis, Matt Menickelly, and Katya Scheinberg. Convergence rate analysis of a stochastic trust-region method via supermartingales. *INFORMS Journal on Optimization*, 1(2):92–119, 2019.

[20] Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2019.

[21] Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching l-bfgs method for machine learning. In *International Conference on Machine Learning*, pages 620–629. PMLR, 2018.

[22] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems*, 20, 2007.

[23] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

[24] Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[25] Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic Hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

[26] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[27] Emmanuel Candes and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse problems*, 23(3):969, 2007.

[28] Yair Carmon, Danielle Hausler, Arun Jambulapati, Yujia Jin, and Aaron Sidford. Optimal and adaptive monteiro-svaiter acceleration. *Advances in Neural Information Processing Systems*, 35:20338–20350, 2022.

[29] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[30] Jiabin Chen, Rui Yuan, Guillaume Garrigos, and Robert M Gower. SAN: stochastic average Newton algorithm for minimizing finite sums. In *International Conference on Artificial Intelligence and Statistics*, pages 279–318. PMLR, 2022.

[31] Yifan Chen, Ethan N Epperly, Joel A Tropp, and Robert J Webber. Randomly pivoted cholesky: Practical approximation of a kernel matrix with few entry evaluations. *Communications on Pure and Applied Mathematics*, 78(5):995–1041, 2025.

[32] Agniva Chowdhuri, Palma London, Haim Avron, and Petros Drineas. Speeding up linear programming using randomized linear algebra. In *Advances in Neural Information Processing Systems*, 2020.

[33] Agniva Chowdhury, Jiasen Yang, and Petros Drineas. Randomized iterative algorithms for Fisher discriminant analysis. In *Uncertainty in Artificial Intelligence*, pages 239–249. PMLR, 2020.

[34] Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

[35] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *ACM Symposium on Theory of Computing*, pages 163–172, 2015.

[36] Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming*, 2016.

[37] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, 27, 2014.

[38] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

[39] Michal Derezinski, Feynman T Liang, Zhenyu Liao, and Michael W Mahoney. Precise expressions for random projections: Low-rank approximation and randomized Newton. In *Advances in Neural Information Processing Systems*, volume 33, pages 18272–18283, 2020.

[40] Michał Dereziński, Christopher Musco, and Jiaming Yang. Faster linear systems and matrix norm approximation via multi-level sketched preconditioning. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1972–2004. SIAM, 2025.

[41] Theo Diamandis, Zachary Frangella, Shipu Zhao, Bartolomeo Stellato, and Madeleine Udell. Genios: an (almost) second-order operator-splitting solver for large-scale convex optimization. *arXiv preprint arXiv:2310.08333*, 2023.

[42] Mateo Díaz, Ethan N Epperly, Zachary Frangella, Joel A Tropp, and Robert J Webber. Robust, randomized preconditioning for kernel ridge regression. *arXiv preprint arXiv:2304.12465*, 2023.

[43] Nikita Doikov. Minimizing quasi-self-concordant functions by gradient regularization of newton method. *arXiv preprint arXiv:2308.14742*, 2023.

[44] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1):3475–3506, 2012.

[45] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[46] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[47] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[48] Jonathan Eckstein and Dimitri P Bertsekas. On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992.

[49] Jonathan Eckstein and Wang Yao. Approximate versions of the alternating direction method of multipliers. *Optimization Online*, 2016.

[50] Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. *Advances in Neural Information Processing Systems*, 28, 2015.

[51] Vladimir Feinberg, Xinyi Chen, Y Jennifer Sun, Rohan Anil, and Elad Hazan. Sketchy: Memory-efficient adaptive regularization with frequent directions. *Advances in Neural Information Processing Systems*, 36:75911–75924, 2023.

[52] Miria Feng, Zachary Frangella, and Mert Pilanci. Cronos: Enhancing deep learning with scalable gpu accelerated convex neural networks. *arXiv preprint arXiv:2411.01088*, 2024.

[53] YT Feng, DRJ Owen, and D Perić. A block conjugate gradient method applied to linear systems with multiple right-hand sides. *Computer methods in applied mechanics and engineering*, 127(1-4):203–215, 1995.

[54] Matthias Feurer, Jan N. van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Mueller, Joaquin Vanschoren, and Frank Hutter. Openml-python: an extensible python api for openml. *arXiv*, 1911.02490, 2019.

[55] Kimon Fountoulakis, Jacek Gondzio, and Pavel Zhlobich. Matrix-free interior point method for compressed sensing problems. *Mathematical Programming Computation*, 6(1):1–31, 2014.

[56] Zachary Frangella, Pratik Rathore, Shipu Zhao, and Madeleine Udell. Promise: Preconditioned stochastic optimization methods by incorporating scalable curvature estimates. *arXiv preprint arXiv:2309.02014*, 2023.

[57] Zachary Frangella, Pratik Rathore, Shipu Zhao, and Madeleine Udell. Sketchysgd: reliable stochastic optimization via randomized curvature estimates. *SIAM Journal on Mathematics of Data Science*, 6(4):1173–1204, 2024.

[58] Zachary Frangella, Joel A Tropp, and Madeleine Udell. Randomized nyström preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, 2023.

[59] Zachary Frangella, Shipu Zhao, Theo Diamandis, Bartolomeo Stellato, and Madeleine Udell. On the (linear) convergence of generalized newton inexact admm. *arXiv preprint arXiv:2302.03863*, 2023.

[60] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[61] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPytorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration. *NeurIPS*, 31, 2018.

[62] Michael Garstka, Mark Cannon, and Paul Goulart. COSMO: A conic operator splitting method for convex conic problems. *Journal of Optimization Theory and Applications*, 190(3):779–810, 2021.

[63] Nidham Gazagnadou, Robert Gower, and Joseph Salmon. Optimal mini-batch and step sizes for SAGA. In *International Conference on Machine Learning*, pages 2142–2150. PMLR, 2019.

[64] Ryan Giordano, William T Stephenson, Runjing Liu, Michael I Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *AISTATS*, pages 1139–1147. PMLR, 2019.

[65] Luc Giraud and Serge Gratton. On the sensitivity of some spectral preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1089–1105, 2006.

[66] Alex Gittens. The spectral norm error of the naive Nyström extension. *arXiv preprint arXiv:1110.5305*, 2011.

[67] Alex Gittens and Michael W Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 17(1):3977–4041, 2016.

[68] Gene Golub and Charles Van Loan. *Matrix computations*. Johns Hopkins University Press, 2013.

[69] Alon Gonen, Francesco Orabona, and Shai Shalev-Shwartz. Solving ridge regression using sketched preconditioned SVRG. In *ICML*, pages 1397–1405. PMLR, 2016.

[70] Yehoram Gordon. Some inequalities for Gaussian processes and applications. *Israel Journal of Mathematics*, 50(4):265–289, 1985.

[71] Robert M Gower, Dmitry Kovalev, Felix Lieder, and Peter Richtárik. RSN: randomized subspace Newton. In *Advances in Neural Information Processing Systems*, 2019.

[72] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

[73] Roger Grosse and James Martens. A Kronecker-factored approximate Fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

[74] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[75] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander Statnikov, Wei-Wei Tu, and Evelyne Viegas. *Analysis of the AutoML Challenge Series 2015–2018*, pages 177–219. Springer International Publishing, Cham, 2019.

[76] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[77] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference On Machine Learning*, pages 1225–1234. PMLR, 2016.

[78] Bingsheng He and Xiaoming Yuan. On the o(1/n) convergence rate of the Douglas–Rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[79] Nicholas J Higham. *Accuracy and stability of numerical algorithms.* SIAM, 2002.

[80] Samuel B Hopkins, Tselil Schramm, Jonathan Shi, and David Steurer. Fast spectral algorithms from sum-of-squares proofs: tensor decomposition and planted sparse vectors. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 178–191, 2016.

[81] Daniel Hsu, Sham M Kakade, and Tong Zhang. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17:1, 2012.

[82] Arun Jambulapati, Jerry Li, Christopher Musco, Aaron Sidford, and Kevin Tian. Fast and near-optimal diagonal preconditioning. *arXiv preprint arXiv:2008.01722*, 2020.

[83] Ruichen Jiang, Michal Derezinski, and Aryan Mokhtari. Stochastic newton proximal extragradient method. *Advances in Neural Information Processing Systems*, 37:90818–90852, 2024.

[84] Billy Jin, Katya Scheinberg, and Miaolan Xie. High probability complexity bounds for line search based on stochastic oracles. *Advances in Neural Information Processing Systems*, 34:9193–9203, 2021.

[85] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26, 2013.

[86] Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23928–23941. Curran Associates, Inc., 2021.

[87] Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Global linear convergence of newton's method without strong-convexity or lipschitz gradients. *arXiv preprint arXiv:1806.00413*, 2018.

[88] Priya Kasimbeg, Frank Schneider, Runa Eschenhagen, Juhan Bae, Chandramouli Shama Sastry, Mark Saroufim, Boyuan Feng, Less Wright, Edward Z Yang, Zachary Nado, et al. Accelerating neural network training: An analysis of the algoperf competition. *arXiv preprint arXiv:2502.15015*, 2025.

[89] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[90] Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. Predicting risk from financial reports with regression. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, page 272–280, USA, 2009. Association for Computational Linguistics.

[91] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, pages 1885–1894. PMLR, 2017.

[92] Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning*, pages 1895–1904. PMLR, 2017.

[93] Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *International Conference on Algorithmic Learning Theory*, volume 117, pages 451–467. PMLR, 2020.

[94] Jacek Kuczyński and Henryk Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

[95] Jonathan Lacotte and Mert Pilanci. Effective dimension adaptive sketching methods for faster regularized least-squares optimization. In *Advances in Neural Information Processing Systems*, 2020.

[96] Jonathan Lacotte and Mert Pilanci. Fast convex quadratic optimization solvers with adaptive sketching-based preconditioners. *arXiv preprint arXiv:2104.14101*, 2021.

[97] Jonathan Lacotte, Yifei Wang, and Mert Pilanci. Adaptive Newton sketch: linear-time optimization with quadratic convergence and effective Hessian dimensionality. In *International Conference on Machine Learning*, pages 5926–5936. PMLR, 2021.

[98] Guanghui Lan. *First-order and stochastic optimization methods for machine learning*, volume 1. Springer, 2020.

[99] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

[100] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[101] Huamin Li, George C Linderman, Arthur Szlam, Kelly P Stanton, Yuval Kluger, and Mark Tygert. Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Transactions on Mathematical Software (TOMS)*, 43(3):1–14, 2017.

[102] Xudong Li, Defeng Sun, and Kim-Chuan Toh. A highly efficient semismooth newton augmented lagrangian method for solving lasso problems. *SIAM Journal on Optimization*, 28(1):433–458, 2018.

[103] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[104] Po-Ling Loh. On lower bounds for statistical learning theory. *Entropy*, 19(11):617, 2017.

[105] Po-Ling Loh and Martin J Wainwright. Regularized M-estimators with nonconvexity: Statistical and algorithmic theory for local optima. *Journal of Machine Learning Research*, 16(19):559–616, 2015.

[106] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, pages 1540–1552. PMLR, 2020.

[107] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[108] Per-Gunnar Martinsson and Joel A Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

[109] Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi. Kernel methods through the roof: handling billions of points efficiently. *NeurIPS*, 33:14410–14422, 2020.

[110] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

[111] Maike Meier and Yuji Nakatsukasa. Fast randomized numerical rank estimation for numerically low-rank matrices. *Linear Algebra and its Applications*, 686:1–32, 2024.

[112] Si Yi Meng, Sharan Vaswani, Issam Hadj Laradji, Mark Schmidt, and Simon Lacoste-Julien. Fast and furious convergence: Stochastic second order methods under interpolation. In *International Conference on Artificial Intelligence and Statistics*, pages 1375–1386. PMLR, 2020.

[113] Xiangrui Meng, Michael A Saunders, and Michael W Mahoney. LSRN: A parallel iterative solver for strongly over-or underdetermined systems. *SIAM Journal on Scientific Computing*, 36(2):C95–C118, 2014.

[114] Renato DC Monteiro. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

[115] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

[116] Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in Neural Information Processing Systems*, 24, 2011.

[117] Riley Murray, James Demmel, Michael W Mahoney, N Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E Lopes, et al. Randomized numerical linear algebra: A perspective on the field with an eye to software. *arXiv preprint arXiv:2302.11474*, 2023.

[118] Cameron Musco and Christopher Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. *NIPS*, 28, 2015.

[119] Yuji Nakatsukasa. Fast and stable randomized low-rank matrix approximation. *arXiv preprint arXiv:2009.11392*, 2020.

[120] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[121] Arkadi S Nemirovski and David B Yudin. *Problem complexity and method efficiency in optimization.* Wiley-Interscience, 1983.

[122] Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical programming*, 140(1):125–161, 2013.

[123] Yurii Nesterov. *Lectures on Convex Optimization*, volume 137. Springer, 2018.

[124] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 1999.

[125] Dianne P O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 1980.

[126] Brendan O'Donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

[127] Christopher C Paige and Michael A Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[128] Courtney Paquette and Katya Scheinberg. A stochastic line search method with expected complexity analysis. *SIAM Journal on Optimization*, 30(1):349–376, 2020.

[129] Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

[130] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[131] Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

[132] John Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, 1998.

[133] Danil Prokhorov. IJCNN 2001 neural network competition, 2001.

[134] Kamiar Rahnama Rad, Arian Maleki, et al. A scalable estimate of the out-of-sample prediction error via approximate leave-one-out cross-validation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(4):965–996, 2020.

[135] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20, 2007.

[136] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2008.

[137] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008.

[138] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008.

[139] Pratik Rathore, Zachary Frangella, Sachin Garg, Shaghayegh Fazliani, Michał Dereziński, and Madeleine Udell. Turbocharging gaussian process inference with approximate sketch-and-project. *arXiv preprint arXiv:2505.13723*, 2025.

[140] Pratik Rathore, Zachary Frangella, Jiaming Yang, Michał Dereziński, and Madeleine Udell. Have askotch: A neat solution for large-scale kernel ridge regression, 2025.

[141] Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: A loss landscape perspective. In *Forty-first International Conference on Machine Learning*, 2024.

[142] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[143] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.

[144] Fred Roosta, Yang Liu, Peng Xu, and Michael W Mahoney. Newton-mr: Inexact Newton method with minimum residual sub-problem solver. *EURO Journal on Computational Optimization*, 10:100035, 2022.

[145] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled Newton methods. *Mathematical Programming*, 174(1):293–326, 2019.

[146] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. *NIPS*, 30, 2017.

[147] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[148] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152. IEEE, 2006.

[149] Katya Scheinberg and Miaolan Xie. Stochastic adaptive regularization method with cubics: A high probability complexity bound. In *2023 Winter Simulation Conference (WSC)*, pages 3520–3531. IEEE, 2023.

[150] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[151] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

[152] Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale, 2023.

[153] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[154] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[155] Will Stephenson, Zachary Frangella, Madeleine Udell, and Tamara Broderick. Can we globally optimize cross-validation loss? quasiconvexity in ridge regression. *Advances in Neural Information Processing Systems*, 34:24352–24364, 2021.

[156] William T Stephenson and Tamara Broderick. Approximate cross-validation in high dimensions with guarantees. In *AISTATS*, pages 2424–2434. PMLR, 2020.

[157] William T Stephenson, Madeleine Udell, and Tamara Broderick. Approximate cross-validation with low-rank data in high dimensions. In *NeurIPS*, volume 33, 2020.

[158] Jingruo Sun, Zachary Frangella, and Madeleine Udell. Sapphire: Preconditioned stochastic variance reduction for faster large-scale statistical learning. *arXiv preprint arXiv:2501.15941*, 2025.

[159] Tian Tong, Cong Ma, and Yuejie Chi. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. *Journal of Machine Learning Research*, 22(1):6639–6701, 2021.

[160] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997.

[161] Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I Jordan. Stochastic cubic regularization for fast nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[162] Joel A Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

[163] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. *Advances in Neural Information Processing Systems*, 30, 2017.

[164] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1454–1485, 2017.

[165] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[166] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[167] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[168] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact Gaussian processes on a million data points. *NeurIPS*, 32, 2019.

[169] Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, volume 13, pages 682–688, 2001.

[170] Ashia Wilson, Maximilian Kasy, and Lester Mackey. Approximate cross-validation: Guarantees for model assessment and selection. In *AISTATS*, pages 4530–4540. PMLR, 2020.

[171] David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[172] Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact Hessian information. *Mathematical Programming*, 184(1-2):35–70, 2020.

[173] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. AdaHessian: An adaptive second order optimizer for machine learning. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 10665–10673, 2021.

[174] Zhewei Yao, Peng Xu, Fred Roosta, and Michael W Mahoney. Inexact nonconvex Newton-type methods. *INFORMS Journal on Optimization*, 3(2):154–182, 2021.

[175] Haishan Ye, Luo Luo, and Zhihua Zhang. Approximate Newton methods. *Journal of Machine Learning Research*, 22(66):1–41, 2021.

[176] Rui Yuan, Alessandro Lazaric, and Robert M Gower. Sketched Newton–Raphson. *SIAM Journal on Optimization*, 32(3):1555–1583, 2022.

[177] Hangrui Yue, Qingzhi Yang, Xiangfeng Wang, and Xiaoming Yuan. Implementing the alternating direction method of multipliers for big datasets: A case study of least absolute shrinkage and selection operator. *SIAM Journal on Scientific Computing*, 40(5):A3121–A3156, 2018.

[178] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[179] Shipu Zhao, Zachary Frangella, and Madeleine Udell. NysADMM: faster composite convex optimization via low-rank approximation. In *International Conference on Machine Learning*, pages 26824–26840. PMLR, 2022.