Randomized Numerical Linear Algebra for Optimization

Madeleine Udell and Zachary Frangella, Stanford University

December 21, 2023

1 Introduction

Advances in randomized numerical linear algebra (randNLA) in the new millenium have reduced the complexity of a variety of fundamental linear algebraic operations, including finding the top-k eigenspace or principal components of a matrix or solving a large-scale linear system of equations. These advances have implications throughout optimization that have not yet been fully realized. This article provides a brief overview of some of the most important and useful tools in randomized numerical linear algebra, a few case studies of their use in optimization, and a tour of what we believe possible with these methods. The major technique is to approximate the most computationally challenging step in an optimization algorithm as the solution to a linear system $Ax = b$; and to approximate the matrix A by a matrix that is low rank + diagonal in order to efficiently solve or precondition an indirect solver for the linear system. In particular, we showcase major advances in linear system solvers, smooth optimization, stochastic optimization, composite (smooth + nonsmooth) optimization, and semidefinite optimization that can be achieved using these methods. These advances yield speedups of 3–58x on important machine learning problems like lasso, logistic regression, SVM, and deep learning.

To understand the potential gains, consider Figure 1, which compares the recent NysADMM method from [39] to SAGA [9] on an ℓ^1-regularized logistic regression problem with a $60,000 \times 60,000$ data matrix formed from a random features transformation of the CIFAR-10 data set (CIFAR-10 rf). NysADMM combines ideas from randNLA with the ADMM algorithm to

obtain a scalable large-scale optimization algorithm. SAGA is a stochastic gradient method and is the default solver used by scikit-learn for solving ℓ^1-regularized logistic regression. Figure 1 shows NysADMM runs 8x faster than SAGA using the default stopping criterion in scikit-learn. We see randNLA can dramatically accelerate large scale optimization.

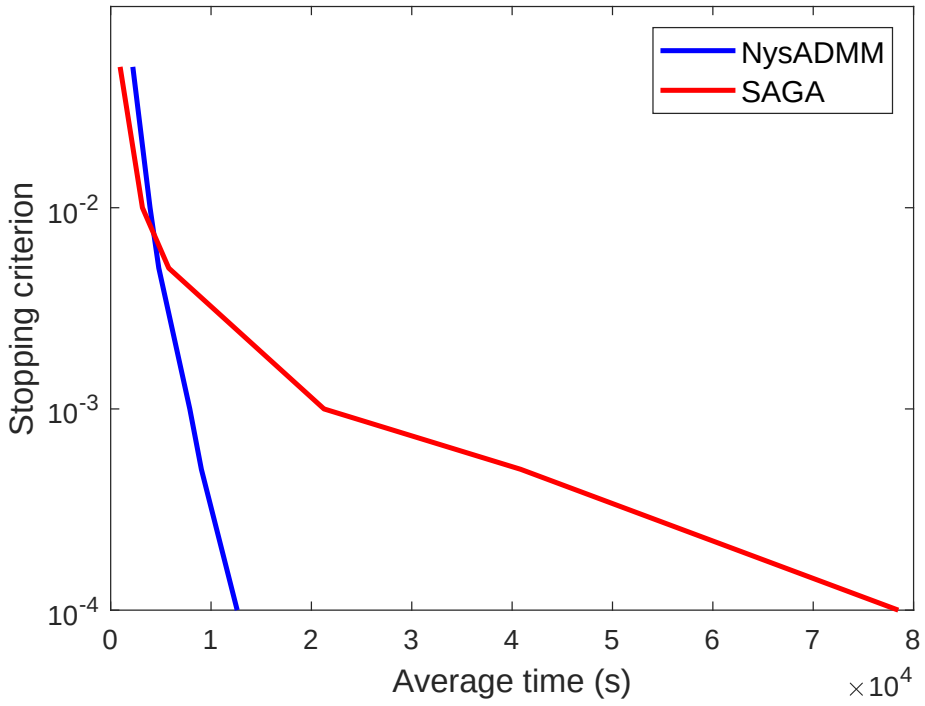

Figure 1: NysADMM vs. SAGA on ℓ^1-logistic regression with CIFAR-10 rf.

2 Background

Randomized rangefinder. The methods we present here build on a fundamental primitive: the randomized rangefinder. Given an input matrix $B \in \mathbf{R}^{m \times n}$ and a target dimension s, the randomized rangefinder produces an orthonormal matrix $Q \in \mathbf{R}^{m \times s}$ whose columns span (as well as possible) the same range as the top s left singular vectors of B. See [22] for a recent review or [16] for an earlier exposition.

One simple method to compute such a Q starts with a random *test matrix* $\Omega \in \mathbf{R}^{n\times s}$, for example, a matrix with iid $\mathcal{N}(0,1)$ entries. We review other choices in Section 4. The randomized rangefinder algorithm computes a *sketch* $Y = B\Omega$ of B, and returns an orthonormal basis Q for Y. The main computational work here consists of s matrix-vectors products (matvecs) with the matrix B, which typically dominates the $O(ns^2)$ work to compute an orthonormal basis. The memory required is $O(ns)$: smaller than the memory required to store B if B is dense.

How well does the rangefinder work? Suppose $m \geq n$ and matrix $B \in \mathbf{R}^{m\times n}$ has singular values $\sigma_1 \geq \cdots \geq \sigma_n$. Then for any $k < s-1$, the expected spectral-norm error of the randomized rangefinder, $\mathbf{E}\,\|(I-P_Q)B\|$, with standard normal test matrix $\Omega \in \mathbf{R}^{n\times s}$ is bounded by

$$\left(1 + \sqrt{\frac{k}{s-k-1}}\right)\sigma_{k+1} + \gamma\left(\sum_{j>k}\sigma_j^2\right)^{1/2},$$

where $\gamma = \mathbf{E}\,\|\Gamma^\dagger\|$ for standard normal $\Gamma \in \mathbf{R}^{k\times s}$ [22]. We see the randomized rangefinder works extremely well when the singular values of B decay rapidly. In particular, if B has rank $r \leq k$ so that $\sigma_{k+1} = 0$, the randomized rangefinder exactly recovers the matrix B as $\mathbf{E}\,\|(I-P_Q)B\| = 0$.

Randomized SVD. The randomized rangefinder can be used to compute a low rank approximation of a matrix. Given a basis Q that approximately spans the top s-dimensional left singular subspace of a matrix B, compute the SVD of $Q^TB = \hat{U}\Sigma V^T$. Then $B \approx (Q\hat{U})\Sigma V^T$ approximates the top-s SVD of B. If Q exactly spans the top s-dimensional left singular subspace of B, then this approximation is exact.

Randomized Nyström approximation. It is even simpler to compute an approximate eigenvalue decomposition of a matrix $A \in \mathbf{R}^{n\times n}$. Given test matrix $\Omega \in \mathbf{R}^{n\times s}$, the Nyström approximation of A is

$$A \approx A\Omega(\Omega^TA\Omega)^\dagger(A\Omega)^T.$$

Notice that given the sketch $Y = A\Omega$, no further access to A is needed. We may form an approximate eigenvalue decomposition using ideas similar to the randomized SVD. A stable implementation requires a bit more care; see [30].

Distributed and parallel. One delightful aspect of randNLA is how well its primitives adapt to modern computational paradigms, such as computation on a GPU or in a distributed system; indeed, the increasing importance of parallel and distributed computation makes the techniques of randNLA an essential part of any computational toolbox. The fundamental workhorse of randNLA is the computation of a sketch $Y = A\Omega$ of a matrix A. It is easy to distribute this computation over the rows of A (concatenate the sketch of each row $Y_{i:} = A_{i:}\Omega$) or over the columns of A (sum the sketch of each column $Y = \sum_j (A_{:j}\Omega)$).

Kinds of guarantees. We state many of the bounds in this newsletter as bounds in expectation. High probability bounds are also available: the simplest are easy to prove from an expectation bound by applying Markov's inequality: if $\mathbf{E}\,\|A - \hat{A}\| \leq \epsilon$, then

$$\mathbf{Prob}\left[\|A - \hat{A}\| \geq \epsilon t\right] \leq \frac{1}{t}.$$

For most results in this newsletter, exponential concentration bounds are also available.

More importantly, these algorithms yield methods that work well and reliably in practice. To prove convergence of optimization algorithms that rely on these methods, we often use union bounds to guarantee good performance at every iteration of an outer optimization algorithm. In our experiments, we simply never see any large deviations from expected performance that would confound the optimizer.

Low rank + diagonal. To state the obvious: low rank approximation works well for matrices that are low rank. However, in optimization, many important matrices are the sum of a low rank part and a diagonal part: for example, the Hessian of a regularized linear system, or the covariance matrix corresponding to a factor model in finance. Direct low rank approximation of these matrices works poorly. Instead, to approximate these matrices, it is best to subtract off the diagonal first and sketch the rest to form a low rank approximation. Interestingly, we are not aware of a linear algebraic method to find a good low rank + diagonal approximation to a matrix. Iterative methods like matrix completion are generally required; these work well but are generally too expensive to be useful in the context of randNLA. Luckily, the diagonal part of the matrix is often known in advance.

Statistical perspectives. For many large-scale machine learning problems, solving the associated optimization problem to high-accuracy often yields little benefit for predictions. In this case, we can replace a deterministic solver with a randomized solver with impunity. We can formalize our understanding of which problems are unharmed by randomized methods by considering the irreducible statistical error due to uncertain problem data. So long as optimization error is bounded by statistical error, there is no statistical benefit to solving the optimization problem to higher precision [1, 19].

Many theoretical and practical algorithms exploit the fact that solving a problem beyond statistical error is unimportant for practical performance. For example, randomized PCA works as well as PCA on large scale statistical datasets: [34] show that randomized PCA via the sketch-and-solve SVD [10, 20] works nearly as well as PCA to recover the top principal components when data is generated by the *spike covariance model* which models the data matrix as low rank + iid Gaussian. As another example, Falkon [28] is a large-scale method for approximate kernel ridge regression that uses column sampling to solve a reduced problem. [28] show that Falkon obtains minimax optimal statistical performance, even though it does not solve the original problem.

3 Optimization problems

This section surveys some paradigmatic applications of randNLA to speed up optimization. We organize our discussion by application area, and consider linear systems, statistical learning problems, smooth optimization, and conic optimization, each in its own subsection. The solution of a linear system is at the computational core of a variety of optimization algorithms, including first-order solvers for a variety of statistical learning problems, Newton's method, and interior point methods, so the first subsection on linear system solvers is fundamental for understanding ideas in the subsequent subsections.

3.1 Linear systems

Many algorithms rely on a fast solver for the regularized linear system

$$(A + \mu I)x = b,$$

where $A \in \mathbf{S}_+^n$ is symmetric psd and $\mu \geq 0$. In the context of the regularized least squares problem

$$\text{minimize} \qquad \frac{1}{2}\|Ux - y\|^2 + \frac{\mu}{2}\|x\|^2,$$

$A = U^T U$ is the Gram matrix and $b = U^T y$ is the righthand side. This problem appears in iteratively reweighted least squares, (kernel) ridge regression, Gaussian processes, approximate cross validation [29], influence functions [17], and hyperparameter optimization [21]. For small systems ($n \leq 50,000$), direct methods are most efficient: these factor the matrix A and then solve the factored system. For larger systems, indirect methods are preferred: these iteratively solve the system by applying the matrix A to the iterate x to form the residual $r = Ax - b$ at each iteration, which is used to update x. Classic Krylov methods like Conjugate Gradients (CG) are guaranteed to return, at the kth iteration, the best solution x in the kth Krylov subspace $\mathcal{K}_k = \text{Span}\{b, Ab, \ldots, A^{k-1}b\}$. CG requires $O(\sqrt{\kappa(A)}\log(\frac{1}{\epsilon}))$ matvecs to reach ϵ accuracy. So the condition number $\kappa(A) = \lambda_1(A)/\lambda_n(A)$ matters tremendously!

Sketch-and-solve. A natural first idea is to solve an easier problem instead. Given a rank-s (say, Nyström) approximation $A \approx \hat{A} = V\hat{\Lambda}V^T$ to $A \in \mathbf{S}_+^n$, it is easy to solve

$$(\hat{A} + \mu I)\hat{x} = b \qquad \text{instead of} \qquad (A + \mu I)x^\star = b.$$

In fact, we can apply the inverse of $\hat{A} + \mu I$ in $O(ns)$ time, since

$$(\hat{A} + \mu I)^{-1} = V(\hat{\Lambda} + \mu I)^{-1}V^T + \frac{1}{\mu}(I - VV^T).$$

This solution paradigm, which returns the solution $\hat{x}$ to the sketched problem, is called *sketch-and-solve*. Variants of this idea approximate $A = U^T U$ using the sketch of a tall-skinny factor $U \in \mathbf{R}^{m\times n}$ [33, 22].

Sketch-and-solve works well if $b \in \text{span}(V)$, but in general, high accuracy solutions require large sketch sizes $s \to n$: a guaranteed ϵ-accurate solution requires a sketch size s for which $\lambda_s \leq \epsilon\mu$ [12], where $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of A.

Hence the method is only useful for low accuracy solutions (large ϵ) to strongly regularized problems (large μ), or when A is low rank. For example,

Nyström sketch-and-solve works well for kernel ridge regression. Indeed, the Nyström approximation was first introduced to machine learning in the context of kernel ridge regression [32], and bears a strong resemblance to the newer Falkon method [28]. In this setting, the sketch-and-solve solution $\hat{x}$ achieves good prediction error even though it may not be close to the true solution $x^\star$ [4, 2].

Sketch-and-precondition. An alternative approach to solving a linear system seeks to improves the condition number by solving a related system. For any *preconditioner* $P \succ 0$,

$$\begin{aligned} Ax = b \quad \Longleftrightarrow \quad & P^{-1/2}Ax = P^{-1/2}b \\ & P^{-1/2}AP^{-1/2}z = P^{-1/2}b \end{aligned}$$

where $x = P^{-1/2}z$. Preconditioning works well when the preconditioner P is easy to invert and results in a system with much smaller condition number $\kappa(P^{-1/2}AP^{-1/2}) \ll \kappa(A)$. Common preconditioners include Jacobi preconditioning $P = \mathbf{diag}(A)$; incomplete Cholesky, which works best for structured sparsity; and randomized preconditioners, which approximate the matrix on its top-k eigenspace and work well for ill-conditioned matrices with fast spectral decay.

Sketch-and-precondition solvers form a randomized preconditioner from a sketch of the matrix A. These solvers enable fast and accurate solutions and can solve both overdetermined and underdetermined least-squares problems. An early sketch-and-precondition method [27] proposed an algorithm with runtime $O(mn\log(n/\epsilon) + n^4)$. Progress in the field has improved the idea substantially: [3] improved the runtime to $O(mn\log(n/\epsilon) + n^3\log(n))$ and showed that randomized least-squares solvers significantly outperform LAPACK on large-scale overdetermined least-squares problems. A sketch-and-precondition variant using sparse sketching matrices [8] enables solution in input-sparsity time $O(\mathrm{nnz}(A)\log\left(\frac{n}{\epsilon}\right) + n^3\log^2(n))$ for matrices satisfying $\mathrm{nnz}(A) \ll mn$. The LSRN method [23] works for underdetermined problems with only black-box access to A.

To explain how these methods work, consider for simplicity an overdetermined least-squares problem $Ux = b$ with $U \in \mathbf{R}^{m\times n}$, $m \gg n$. Sketch-and-precondition computes a sketch $\Omega^T U$ using test matrix $\Omega \in \mathbf{R}^{m\times s}$, performs a QR-decomposition $\Omega^T U = QR$ of the resulting $s \times n$ matrix, and uses R^{-1} as a preconditioner for U.

We can show UR^{-1} is well-conditioned using the *subspace embedding property*. Suppose $\Omega \in \mathbf{R}^{m\times s}$ is a Gaussian matrix with sketch size $s = O(n/\zeta^2)$ where $\zeta \in (0,1)$. Then the ζ-subspace embedding property holds [22, 33]: with high probability for all $x \in \mathbf{R}^n$,

$$(1-\zeta)\|Ux\|^2 \le \|\Omega^T Ux\|^2 \le (1+\zeta)\|Ux\|^2.$$

We can use this result to show that the preconditioned system UR^{-1} preserves the lengths of vectors in the range of U. To see how, let $x = R^{-1}z$. By the subspace embedding property,

$$\frac{1}{1+\zeta}\|\Omega^T UR^{-1}z\|^2 \le \|UR^{-1}z\|^2 \le \frac{1}{1-\zeta}\|\Omega^T UR^{-1}z\|^2.$$

Now using $\|\Omega^T UR^{-1}z\|^2 = \|Qz\|^2 = \|z\|^2$, the above display becomes

$$\frac{1}{1+\zeta}\|z\|^2 \le \|UR^{-1}z\|^2 \le \frac{1}{1-\zeta}\|z\|^2,$$

which bounds the condition number $\kappa(UR^{-1}) \le \sqrt{\frac{1+\zeta}{1-\zeta}}$. In particular, setting $\zeta = \frac{1}{2}$, we have $\kappa(UR^{-1}) \le \sqrt{3}$. Hence preconditioned conjugate gradient (PCG) applied to UR^{-1} converges rapidly.

In practice, the sketch can often be computed faster using a structured test matrix like a randomized trigonometric transform or sparse sign matrix; see Section 4. These structured test matrices also satisfy the ζ-subspace embedding property with high probability but generally require a larger sketch size [22].

Unfortunately, this approach to sketch-and-precondition is restricted to highly overdetermined or underdetermined problems, as it requires a QR decomposition of $\Omega^T U \in \mathbf{R}^{s\times n}$ at a cost of $O(n^3)$. It is not useful for square(ish) systems when the smaller dimension n is still large.

Nyström PCG. Nyström PCG provides an alternative to sketch-and-precondition that works for square systems $A \in \mathbf{R}^{n\times n}$, and generalizes to an efficient method for rectangular systems $Ux = b$, $U \in \mathbf{R}^{m\times n}$, by forming the normal equations $Ax := U^T Ux = U^T b$.

Nyström PCG uses the Nyström approximation to the matrix A: given a rank-s Nyström approximation

$$\hat{A}_{\text{nys}} = V\hat{\Lambda}V^T \approx A \in \mathbf{S}^n_+,$$

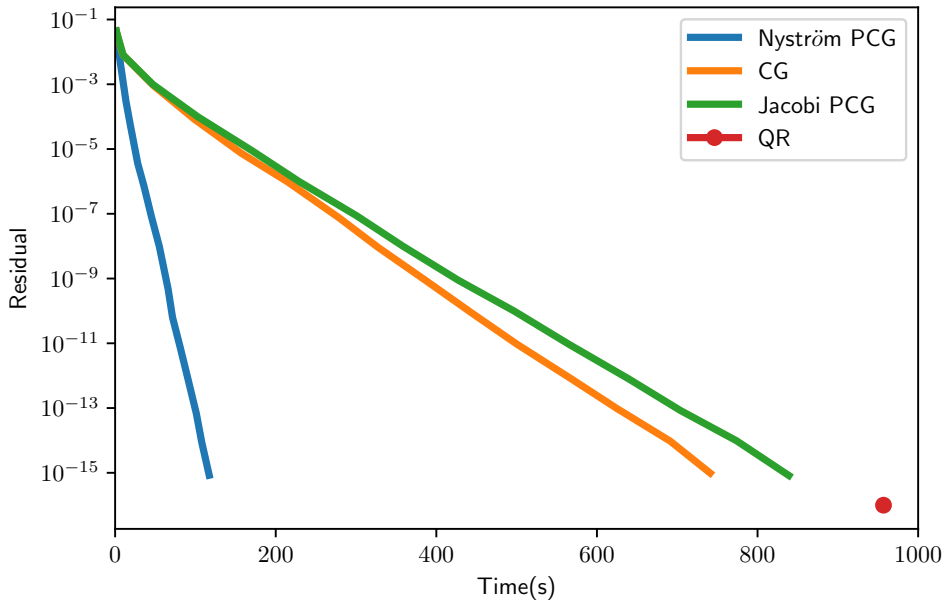

Figure 2: Nyström PCG is significantly faster than traditional NLA methods on YearMSD dataset with 15,000 random features.

the *Nyström preconditioner* for the regularized system $(A + \mu I)x = b$ is

$$P_{\text{nys}} = \frac{1}{\hat{\lambda}_s + \mu} V(\hat{\Lambda} + \mu I)V^T + (I - VV^T).$$

The inverse of this preconditioner can be applied in $O(ns)$:

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

We can bound the number of iterations required to achieve a solution of accuracy ϵ using the Nyström preconditioner in terms of the *effective dimension* at μ, a smoothed count of eigenvalues $\geq \mu$:

$$d_{\text{eff}}(\mu) = \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_j + \mu}.$$

The effective dimension bounds sketch size required for the preconditioner to achieve constant condition number [12]:

Theorem 1 *Construct the randomized Nyström preconditioner P with rank $s = 2\lceil 1.5 d_{\text{eff}}(\mu)\rceil + 1$. Then*

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

High probability bounds ensuring small condition number may be established by using a slightly larger sketch size [12, 39].

Contrast this result with sketch-and-precondition: while sketch-and-precondition methods rely on the subspace embedding property and can accelerate the solution of very skinny or fat rectangular linear systems, Nyström PCG operates on the principle of low-rank approximation and so is useful for square and squareish systems. The main requirement is that the effective dimension is small, or equivalently, that the spectrum of A decays quickly. This property is common in statistical learning problems. Fig. 2 compares Nyström PCG to traditional numerical linear algebra methods for a regularized least-squares problem.

Iterative sketching Sketch-and-solve requires a sketch size of $O(1/\epsilon^2)$ to ensure an ϵ-approximate solution, so it is not practical for high precision solutions. Instead, a natural idea is to solve a sequence of linear systems with sketch-and-solve to converge to higher accuracy. For example, given an approximate solution $x^{(0)}$ to $Ax = b$, iterative refinement via Richardson's iteration provides a classical technique to improve the solution:

$$x^{(k+1)} = x^{(k)} - \eta(Ax^{(k)} - b),$$

where η is a suitably chosen stepsize. Unfortunately, Richardson's iteration converges quite slowly in practice: $O(\kappa \log(1/\epsilon))$ iterations for an ϵ-accurate solution, where κ is the condition number of A. This complexity is a factor $\sqrt{\kappa}$ worse than CG.

Pilanci and Wainwright [25] close this gap for overdetermined unconstrained least-squares problems using a preconditioned Richardson's iteration,

$$x^{(k+1)} = x^{(k)} - \left(\frac{1}{m} A_{S^{(k)}}\right)^{-1} (b - Ax^{(k)})$$

where $A = U^T U$, $A_{S^{(k)}} = U^T (S^{(k)})^T S^{(k)} U$, and $b = U^T y$. Applying the preconditioner $\left(\frac{1}{m} A_{S^{(k)}}\right)^{-1}$ requires solving the sketched linear system, which explains the name iterative sketching. With a sketch size $m = \Omega(p)$, [25] shows that iterative sketching yields an ϵ-accurate solution after $O\left(\log\left(\frac{1}{\epsilon}\right)\right)$ iterations, independent of the condition number, and offers extensions for constrained least-squares problems such as the lasso. More recently, [18] extend the iterative Hessian sketch to ridge regression and obtain analogous results provided the sketch size satisfies $m = \Omega(d_{\text{eff}}(\mu))$. Gower et al. [15] propose a third approach: at each iteration the linear system is sketched and the next iterate is chosen to minimize the distance to the previous iterate among all solutions to the sketched system.

Iterative sketching, like sketch-and-precondition, can accelerate optimization methods by replacing linear system solves inside optimization algorithms (such as Newton's method [26, 14]) with faster sketched linear system solves. However, in the experience of the authors, sketch-and-precondition (and Nyström PCG in particular) work as well if not better [12]. See Fig. 5 below for an example.

3.2 Statistical learning problems

Consider the *composite* optimization problem

$$\text{minimize} \quad \ell(Ax) + r(x)$$

where $\ell : \mathbf{R}^n \to \mathbf{R}$ is smooth, $A \in \mathbf{R}^{m \times n}$ is a feature matrix, and $r : \mathbf{R}^n \to \mathbf{R}$ is *proxable*: that is, suppose there is an easy (even, closed form) solution to $\mathbf{prox}_r(x) = \text{argmin}_y \, r(y) + \frac{1}{2}\|x - y\|^2$ [24]. For example, for $r(x) = \|x\|_1$, $\mathbf{prox}_r(x)$ is soft-thresholding operator. As examples, we have three important problems in statistical learning:

- the lasso,
$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|_2^2 + \gamma\|x\|_1,$$
with squared error loss $\ell(x) = \frac{1}{2}\|Ax - b\|_2^2$;
- ℓ_1-regularized logistic regression,
$$\text{minimize} \quad \ell_{\text{logistic}}(Ax) + \gamma\|x\|_1$$
with logistic loss $\ell(Ax) = \ell_{\text{logistic}}(Ax) = \sum_{i=1}^n \log(1 + \exp(-b_i(Ax)_i)$; and

- the support vector machine (SVM) problem

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

  with loss $\ell(x) = \frac{1}{2}x^T \mathrm{diag}(b) K \mathrm{diag}(b) x - \mathbf{1}^T x$.

The state-of-the-art solvers for each of these problems are different: for lasso, glmnet uses coordinate descent [13]; for logistic regression, SAGA is a stochastic average gradient method [9]; and for SVM, LIBSVM, uses a sequential minimal optimization (pairwise coordinate descent) method [6].

NysADMM. However, all of these problems can be addressed simply using an operator splitting framework like the alternating directions method of multipliers (ADMM Algorithm 1; see [5] for a tutorial overview), in which the major computational challenge is the solution of an unconstrained minimization involving a large-scale data matrix (step 4 of Algorithm 1).

Algorithm 1 ADMM

1: **Input:** loss function ℓ, regularization r, stepsize ρ,
2: initial z^0, $u^0 = 0$
3: **for** $k = 0, 1, \ldots$ **do**
4: $\quad x^{k+1} = \mathrm{argmin}_x \{\ell(Ax) + \frac{\rho}{2}\|x - z^k + u^k\|_2^2\}$
5: $\quad z^{k+1} = \mathrm{argmin}_z \{r(z) + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2\}$
6: $\quad u^{k+1} = u^k + x^{k+1} - z^{k+1}$
return $x_\star$ (nearly) minimizing $\ell(x) + r(x)$

Our recent paper [39], with coauthor Shipu Zhao, shows how to accelerate ADMM for these problems using ideas from randNLA. To improve the runtime of ADMM, recall that *inexact ADMM*, which solves step 4 approximately with error ε^k at iteration k, converges if $\sum_k \varepsilon^k < \infty$ [11]. To employ our randNLA toolbox, approximate step 4 by a quadratic optimization, and solve the resulting linear system with NyströmPCG.

More precisely, if ℓ is twice differentiable, approximate the objective near the previous iterate x^k as

$$\begin{aligned} \ell(Ax) \;\approx\; & \ell(Ax^k) + (x - x^k)^T A^T \nabla \ell(x^k) \\ & + \frac{1}{2}(x - x^k)^T A^T H_\ell(x^k) A (x - x^k), \end{aligned}$$

where H_ℓ is the Hessian of ℓ. With this approximation, the problem reduces to a linear system: set $r^k = \rho z^k - \rho u^k + A^T H_\ell(x^k) A x^k - A^T \nabla \ell(x^k)$ and find x to solve

$$(A^T H_\ell(x^k) A + \rho I) x = r^k.$$

Observe the Hessian $A^T H_\ell(x^k) A$ has the feature matrix A inside of it, and so generally exhibits fast spectral decay. Moreover, the stepsize ρ regularizes linear system. Interestingly, this fact simplifies the choice of ρ for ADMM, which is often quite challenging: as larger ρ yields an easier-to-solve subproblem, erring on the side of large ρ is better. Empirically, we find a choice of 10 works well across a startlingly wide variety of problems [39].

For this method to work, in theory we must solve to tolerance ε^k at iteration k, where $\sum_k \varepsilon^k < \infty$; if the sketch size $s \approx d_{\text{eff}}(\rho)$, we will need $\leq O(\log(1/\varepsilon^k))$ CG steps per iteration. On the other hand, in practice we find good performance by setting ε^k as the geometric mean of the primal and dual residual (as recommended in [5], and using a sketch size $s = 50$ uniformly. If ℓ is quadratic (*e.g.*, lasso and SVM), $H_\ell(x^k) = H_\ell$ is constant, so we need only sketch $A^T H_\ell A$ once and can reuse the sketch at each iteration; otherwise (for logistic regression), we find it suffices to re-sketch infrequently, say, every 50 iterations.

This simple paradigm yields substantial speedups over state-of-the-art solvers, we saw in Fig. 1 that NysADMM significantly outperforms SAGA on ℓ^1-logistic regression. NysADMM also delivers impressive numerical results on other problem classes as well. Fig. 3 shows NysADMM outperforms standard solvers such as glmnet on a large Lasso problem instance ($m = 13,000, n = 27,648$), while in Fig. 4 NysADMM runs almost $4\times$ faster than LIBSVM on a kernelized SVM instance with $m = 60,000$. Thus NysADMM has potential to provide a unified framework for a wide class of statistical learning problems.

3.3 Smooth optimization

Consider the problem

$$\text{minimize } F(x) \coloneqq \quad f(x) + \frac{\mu}{2}\|x\|^2$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is twice differentiable and $\mu > 0$ is a regularization parameter. Newton's method minimizes this objective by solving a sequence of quadratic optimization problems, or, equivalently, linear systems: given

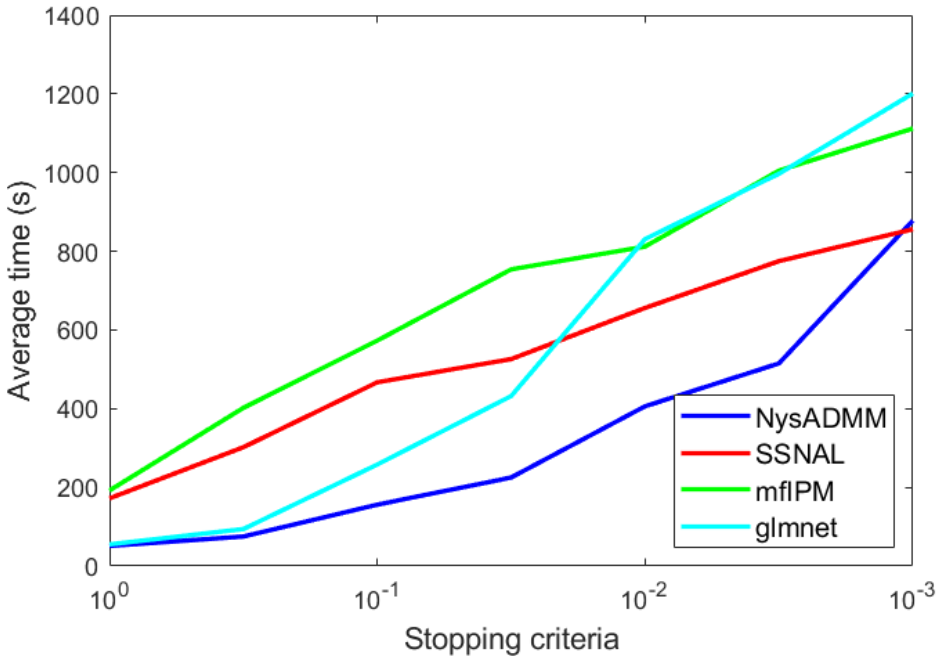

Figure 3: NysADMM outperforms other lasso solvers for moderate precision.

iterate x, Newton's method computes the gradient $g = \nabla f(x) + \mu x$, the Hessian $H = \nabla^2 f(x) + \mu I$, and the Newton direction $p = H^{-1}g$, and updates the iterate as $x \leftarrow x - \eta p$ where $\eta \in \mathbf{R}$ is a step-size that can be chosen using line-search. The main computational burden of Newton's method is in solving the linear system $Hp = g$ to compute the Newton direction, which costs $O(n^3)$ using a direct solver.

Let $d_\star := \sup_{x \in S(x_0)} d_{\text{eff}}^\mu(x)$, where $S(x_0) = \{x : F(x) \leq F(x_0)\}$ is the sublevel set of F at x_0, and $d_{\text{eff}}^\mu(x)$ is the effective dimension with respect to μ of the Hessian evaluated at x. Hence the effective dimension of the Hessian along the optimization path is bounded by $d_\star$.

NysPCG-Newton. A simple improvement for large scale problems, which we call *NysPCG-Newton*, solves for the Newton direction p using preconditioned CG. Suppose we use Nyström PCG to compute the search direction. Then if the preconditioner is constructed with a sketch size of $s = O(d_\star)$, the theory for the Nyström preconditioner in Section 3.1 guarantees we can

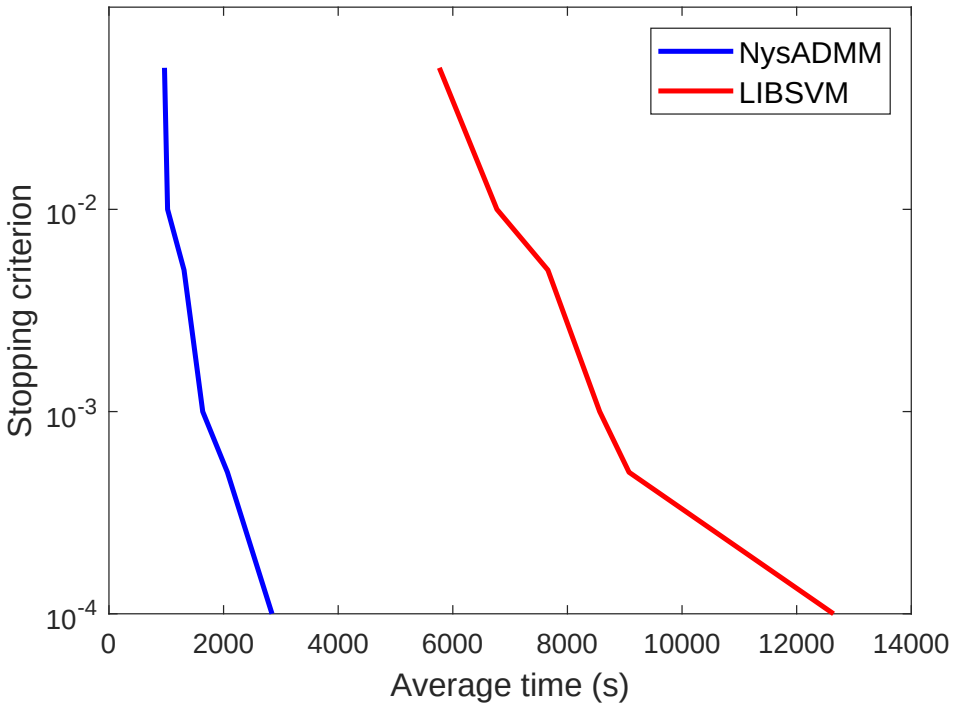

Figure 4: NysADMM outperforms LIBSVM on a kernelized SVM problem with CIFAR-10 rf.

solve the Newton system in a constant number of iterations. Moreover, under appropriate conditions this method exhibits superlinear convergence; see Fig. 5 for an example.

nyssNewton. A related quasi-Newton algorithm we call *Nyström sketch-and-solve Newton (nyssNewton)* uses a sketch-and-solve idea to replace the Hessian $\nabla^2 f(x)$ with a low rank approximation $\hat{H}_f$ in the computation of the Newton direction. It computes the following update

$$x_{k+1} = x_k - \eta_k(\hat{H}_{f_k} + \mu I)^{-1} g_k.$$

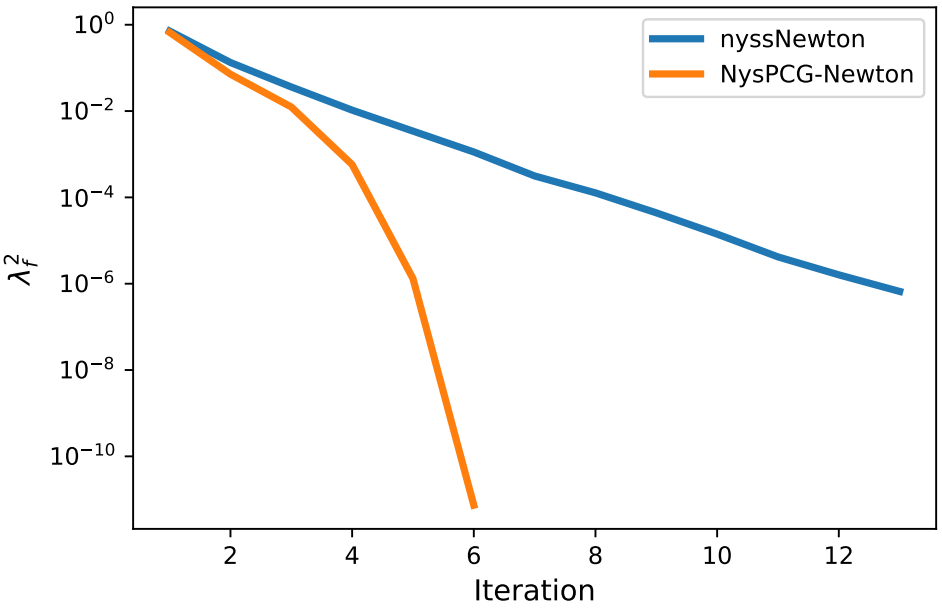

Figure 5: Convergence of the squared Newton decrement on an ℓ^2-logistic regression problem on the MNIST-rotated dataset with random features ($m = 62,000$, $n = 8,000$). NysPCG-Newton converges superlinearly on this problem, while nyssNewton converges linearly. Both methods were terminated when $\lambda_f^2 \leq 10^{-6}$. NysPCG-Newton took 75.9 seconds to run while nyssNewton took 92.7 seconds

If at each iteration k we construct $\hat{H}_{f_k}$ with sketch size $O(d_\star/\zeta)$ where $\zeta \in (0,1)$ is a user selected parameter, then with high probability,

$$\hat{H}_{f_k} + \mu I \preceq H_k \preceq (1+\zeta)(\hat{H}_{f_k} + \mu I).$$

Given this relation, if the step-size η_k is chosen via line-search, then Theorem 5 in [35] guarantees that when f is self-concordant nyssNewton returns an ϵ-suboptimal point in $O(\log(1/\epsilon))$-iterations. That is, nyssNewton converges linearly to the optimum independent of the condition number (compared to quadratic for Newton's method or superlinearly for NysPCG-Newton). This excellent convergence rate comes at a relatively cheap per-iteration cost: constructing and factoring $\hat{H}_{f_k}$ requires $O(T_{\text{mv}} d_\star)$ computation, so the total

complexity of the algorithm is $O\left(T_{\text{mv}} d_\star \log(1/\epsilon)\right)$.

3.4 Conic optimization

A related line of work uses randNLA to sketch the *decision variable* and thereby reduce the *memory* required to run the algorithm. Examples include [38, 37]. This approach makes sense for matrix optimization problems whose solutions are expected to be low rank, and often results in a computational speedup as an additional benefit. Unfortunately, the requirement that the sketch be *linear* in the decision variable limits the class of algorithms that can use this trick to primal-dual algorithms like the conditional gradient method or [36].

In contrast, the ideas presented above in Section 3.2 sketch internal problem data (such as the constraint matrix or objective Hessian) in order to reduce the *time* required to perform the inner algorithm iterations, and work on a wide range of algoritmic templates, from conjugate gradient to Newton's method to ADMM.

A separate idea is to accelerate interior point methods using a randomized linear system solver to solve the internal Newton systems, as in *e.g.*[7]. We believe this idea, coupled with NysPCG-Newton or nyssNewton (see Section 3.3, has substantial potential to improve IPMs.

4 Structured test matrices

Structured test matrices can reduce the time required to compute the sketch of a matrix. Our discussion of this topic follows the excellent review in [22]. Most randNLA methods begin by computing a linear sketch $A\Omega$, where Ω is a random test matrix. The canonical choice takes Ω to be a Gaussian random matrix. While Gaussian test matrices work well in practice and facilitate analysis, they can be expensive to store ($O(nk)$ for Gaussian $\Omega \in \mathbf{R}^{n\times k}$) and compute with ($O(n^2k)$ to compute $A\Omega$ for dense $A \in \mathbf{R}^{n\times n}$).

In contrast, structured test matrices are designed so that Ω is efficient to store and to apply. Two classes of structured test matrices are particularly useful: randomized trigonometric transforms (RTTs), which work best when A is dense and unstructured, and sparse sign matrices (SSMs), which work best when A is sparse.

RTTs have the form

$$\Omega = \sqrt{\frac{n}{k}} \Pi F R,$$

where $\Pi \in \mathbf{R}^{n \times n}$ is signed permutation matrix, F is a discrete trignonometric transform such as the discrete cosine transform or Hadamard transform, and $R \in \mathbf{R}^{n \times k}$ is a random restriction operator that selects k-coordinates uniformly at random. The cost of storing a RTT is $O(n \log(n))$, while computing $A\Omega$ costs only $O(n^2 \log(k))$: an exponential (in k) improvement compared to a Gaussian test matrix! RTTs perform similarly to Gaussian test matrices in practice, despite offering weaker theoretical guarantees. The major limitation of RTTs is that they require explicit access to the columns of A, so they are not appropriate when only a black-box matvec oracle for A is available or when A is distributed. Moreover, a high-quality implementation of the relevant fast trigonometric transform is required to realize the full benefit of RTTs: the default implementation e.g. in matlab yields complexity $O(n^2 \log(n))$ rather than $O(n^2 \log(k))$.

SSMs take the form

$$\Omega = \sqrt{\frac{n}{k}} \left[\omega_1 | \omega_2 | \cdots | \omega_k \right] \in \mathbf{R}^{n \times k},$$

where each column $\omega_j \in \mathbf{R}^n$ has at most s non-zero entries. To construct each ω_j, we draw s random signs and place them in s coordinates selected uniformly at random. We may store Ω with $O(ns \log(n))$ numbers and can compute $A\Omega$ in $O(nsk)$ time. Variants of sparse embeddings differ in how they trade off sparsity compared to the number of samples needed to ensure a high quality sketch. SSMs are natural choices for sparse data. Some variants can compute $A\Omega$ in $O(\text{nnz}(A))$ time: much faster than the $O(n^2 k)$ time required by Gaussian test matrices! In contrast to RTTs, SSMs are compatible with matvec oracle access to A. However, unless the matvec oracle has special structure, computing $A\Omega$ for an SSM may be be no cheaper than using a Gaussian test matrix. The downsides of sparse sign matrices are similar to RTTs: they require careful implementation to extract their full computational benefits, particularly in the distributed setting.

For more discussion, see sections 9 and 10 in the survey [22]; or for pseudocode implementations, see the appendix of [31].

5 Conclusion

We have seen how fundamental innovations in randomized numerical linear algebra yield important primitives for speeding up optimization problems, including linear system solvers, smooth optimization, structured composite problems such as lasso, regularized logistic regression, and SVMs, and even interior point methods. The methods presented here demonstrate potential speedups of 10-100x over standard approaches. Much more work remains to realize the promise of randNLA throughout optimization!

Acknowledgements

The authors gratefully acknowledge support from NSF CAREER Award IIS-1943131, ONR Award N000142312203 and N000142212825, and the Alfred P. Sloan Foundation.

References

[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] Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, 2015.

[3] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

[4] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, 2013.

[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[6] 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.

[7] Agniva Chowdhury, Palma London, Haim Avron, and Petros Drineas. Faster randomized infeasible interior point methods for tall/wide linear programs. *Advances in Neural Information Processing Systems*, 33:8704–8715, 2020.

[8] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the Forty-fifth Annual ACM Symposium On Theory of Computing*, pages 81–90, 2013.

[9] 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.

[10] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[11] 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.

[12] Zachary Frangella, Joel A. Tropp, and Madeleine Udell. Randomized Nyström preconditioning. *arXiv preprint arXiv:2110.02820*, 2021.

[13] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[14] Robert M Gower, Dmitry Kovalev, Felix Lieder, and Peter Richtárik. RSN: randomized subspace Newton. In *Advances in Neural Information Processing Systems*, 2019.

[15] Robert M Gower and Peter Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

[16] 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.

[17] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[18] Jonathan Lacotte and Mert Pilanci. Effective dimension adaptive sketching methods for faster regularized least-squares optimization. In *Advances in Neural Information Processing Systems*, 2020.

[19] Po-Ling Loh. On lower bounds for statistical learning theory. *Entropy*, 19(11):617, 2017.

[20] Miles Lopes, N Benjamin Erichson, and Michael Mahoney. Error estimation for sketched svd via the bootstrap. In *International Conference on Machine Learning*, pages 6382–6392. PMLR, 2020.

[21] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

[22] Per-Gunnar Martinsson and Joel A Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

[23] 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.

[24] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[25] Mert Pilanci and Martin J Wainwright. Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

[26] 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.

[27] 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.

[28] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. *Advances in Neural Information Processing Systems*, 30, 2017.

[29] Will Stephenson, Madeleine Udell, and Tamara Broderick. Approximate cross-validation with low-rank data in high dimensions. *Advances in Neural Information Processing Systems*, 33:9830–9840, 2020.

[30] 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.

[31] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM Journal on Scientific Computing*, 41(4):A2430–A2463, 2019.

[32] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 13, 2000.

[33] David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[34] Fan Yang, Sifan Liu, Edgar Dobriban, and David P Woodruff. How to reduce dimension with PCA and random projections? *IEEE Transactions on Information Theory*, 67(12):8154–8189, 2021.

[35] Haishan Ye, Luo Luo, and Zhihua Zhang. Approximate Newton methods. *Journal of Machine Learning Research*, 22(66), 2021.

[36] Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. A conditional-gradient-based augmented lagrangian framework. In *International Conference on Machine Learning*, pages 7272–7281. PMLR, 2019.

[37] Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

[38] Alp Yurtsever, Madeleine Udell, Joel Tropp, and Volkan Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Artificial intelligence and statistics*, pages 1188–1196. PMLR, 2017.

[39] Shipu Zhao, Zachary Frangella, and Madeleine Udell. NysADMM: faster composite convex optimization via low-rank approximation. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022.