SIAG/OPT Views and News

A Forum for the SIAM Activity Group on Optimization

Volume 30 Number 1 October 2022

Contents

Articles

Flows, Scaling, and Entropy Revisited: A Unified Perspective via Optimizing Joint Distributions
Jason M. Altschuler 1

Randomized Numerical Linear Algebra for Optimization
Madeleine Udell and Zachary Frangella 9

High-dimensional Optimization
Courtney Paquette and Elliot Paquette 17

Bulletin

Event announcements 29

Book announcements 29

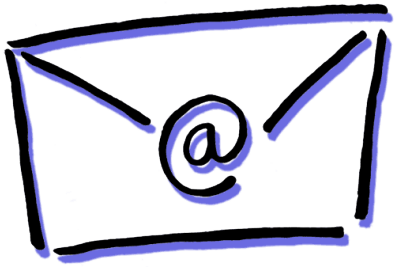
Chair's Column

Katya Scheinberg 29

Comments from the Editors

Dmitriy Drusvyatskiy, Matt Menickelly, Pietro Belotti 30

Are you receiving this by postal mail?
Do you prefer electronic delivery?



Email siagoptnews@lists.mcs.anl.gov

Articles

Flows, Scaling, and Entropy Revisited: A Unified Perspective via Optimizing Joint Distributions

Jason M. Altschuler

Laboratory for Information and Decision Systems (LIDS)
Massachusetts Institute of Technology
Cambridge, MA, 02139.
ja4775@nyu.edu
https://www.mit.edu/~jasonalt/

In this short expository note, we describe a unified algorithmic perspective on several classical problems which have traditionally been studied in different communities. This perspective views the main characters—the problems of Optimal Transport, Minimum Mean Cycle, Matrix Scaling, and Matrix Balancing—through the same lens of *optimization problems over joint probability distributions $P(x, y)$ with constrained marginals*. While this is how Optimal Transport is typically introduced, this lens is markedly less conventional for the other three problems. This perspective leads to a simple and unified framework spanning problem formulation, algorithm development, and runtime analysis.

Some fragments of this story are classical—for example, the approach for solving Optimal Transport via the Sinkhorn algorithm for Matrix Scaling dates back at least to the 1960s [58] and by now is well-known in multiple communities spanning economics, statistics, machine learning, operations research, and scientific computing [51, 37, 47, 27].

Yet, the story described in this note was only recently developed in full—for example, the use of probabilistic inequalities to prove near-optimal runtimes [13, 14], and the parallels between Optimal Transport and Minimum Mean Cycle which provide a framework for applying popular algorithmic techniques for the former problem to the latter [11].

	Fixed marginals	Symmetric marginals
Linear program	**Optimal Transport**	**Minimum Mean Cycle**
Entropic regularization	**Matrix Scaling**	**Matrix Balancing**
Polytope	Transportation polytope	Circulation polytope
Simple algorithm	Sinkhorn algorithm	Osborne algorithm
Algorithm in dual	Block coordinate descent	Entrywise coordinate descent
Per-iteration progress	KL divergence	Hellinger divergence

Table 1: Although not traditionally viewed in this way, the four bolded optimization problems can be viewed under the same lens: each optimizes a joint probability distribution $P(x, y)$ with similar constraints and objectives. The purpose of this note is to describe the unifying connections in this table and how they can be exploited to obtain practical algorithms with near-optimal runtimes for all four problems.

These developments all hinge on the perspective of optimizing distributions highlighted in this note. For some problems, this leads to rigorous guarantees that justify algorithms that have long been used in practice and are nowadays the default algorithms in many numerical software packages (e.g., POT, OTT, OTJulia, GeomLoss, MATLAB, R, Lapack, and Eispack); for other problems, this leads to even faster algorithms in practice and/or theory.

The goal of this note is to explain this story with an emphasis on the unifying connections as summarized in Table 1. There are several ways to tell this story. We start by introducing Optimal Transport and Minimum Mean Cycle as graph problems (§1) before re-formulating them as optimization problems over joint distributions (§2), which makes clear their connections to Matrix Scaling and Balancing (§3), and naturally leads to the development (§4) and analysis (§5) of scalable algorithms based on entropic regularization. Throughout this narrative, we keep two threads as equals: the fixed marginal setting (Table 1, left) and the symmetric marginal setting (Table 1, right). As the parallels between these two threads are nearly exact(!), the mental overhead of two threads is hopefully minimal and outweighed by the pedagogical benefit of unifying these two halves of the story—which have often been studied in separate papers and sometimes even separate communities.

The presentation of this article is based on Part I of the author's thesis [6], and in the interest of brevity, we refer the reader there for proofs and further references (we make no attempt to be comprehensive here given the short length of this note and the immense literature around each of the four problems, e.g., thousands of papers in the past decade just on Optimal Transport).

Notation. We associate the set of probability distributions on n atoms with the simplex $\Delta_n := \{p \in \mathbb{R}^n_{\geqslant 0} : \sum_i p_i = 1\}$, and the set of joint probability distributions on $n \times n$ atoms with $\Delta_{n\times n} := \{P \in \mathbb{R}^{n\times n}_{\geqslant 0} : \sum_{ij} P_{ij} = 1\}$. We write $\mathbf{1}$ to denote the all-ones vector in $\mathbb{R}^n$, and $G = (V, E, c)$ to denote a graph with vertex set V, edge set E, and edge weights $c : E \to \mathbb{R}$. One caution: we write $\exp[A]$ with brackets to denote the *entrywise* exponential of a matrix A.

1 Two classical graph problems

Here we introduce the first two characters in our story: the problems of Optimal Transport (OT) and Minimum Mean Cycle (MMC). We begin by introducing both in the language of graphs as this helps make the parallels clearer when we transition to the language of probability afterward.

1.1 Optimal Transport

In the language of graphs, the OT problem is to find a flow on a bipartite graph that routes "supplies" from one vertex set to "demands" in the other vertex set in a minimum-cost way. For convenience, we renormalize the supply/demand to view them as distributions and abuse notation by denoting both vertex sets by $[n] = \{1, \dots, n\}$.

Definition 1 (Optimal Transport)**.** Given a weighted bipartite graph $G = ([n] \cup [n], E, c)$ and distributions $\mu, \nu \in \Delta_n$, the OT problem is

$$\min_{\substack{f:E\to\mathbb{R}_{\geqslant 0} \\ \sum_{j\in[n]} f(i,j)=\mu_i,\ \forall i\in[n] \\ \sum_{i\in[n]} f(i,j)=\nu_j,\ \forall j\in[n]}} \sum_{e\in E} f(e)c(e)\,. \tag{1}$$

OT dates back to Monge in the 18th century when he asked: what is the least-effort way to move a mound of dirt into a nearby ditch of equal volume [41]? In operations research, OT appears in textbook problems where one seeks to, e.g., find the minimum-cost way to ship widgets from factories to stores [16]. Recently, OT has become central to diverse applications in data science—ranging from machine learning to computer vision to the natural sciences—due to the ability of OT to compare and morph complex data distributions beyond

	Fixed marginals	Symmetric marginals
Graphs	Bipartite Flows	Circulations
Matrices	Transportation polytope	Circulation polytope
Distributions	Couplings	Self-couplings

Table 2: Informal dictionary for translating between graphs, matrices, and distributions. See §2 for details.

just dirt mounds and widget allocations. Prototypical examples include data distributions arising from point clouds in statistics, images or 3D meshes in computer graphics, document embeddings in natural language processing, or cell phenotypes or fMRI brain scans in biology. For details on the many applications of OT, we refer to the recent monograph [47].

A central challenge in all these applications is scalable computation. Indeed, data-driven applications require computing OT when the number of data points n in each distribution is large. Although OT is a linear program (LP), it is a very large LP when n is in, say, the many thousands or millions, and it is a longstanding challenge to develop algorithms that can compute OT (even approximately) in a reasonable amount of time for large n. See, e.g., the standard texts [16, 52, 47, 3] for a discussion of the extensive literature on OT algorithms which dates back to Jacobi in the 19th century.

1.2 Minimum Mean Cycle

We now turn to the second character in our story. Below, recall that a cycle is a sequence of edges that starts and ends at the same vertex.

Definition 2 (Minimum-Mean-Cycle)**.** Given a weighted directed graph $G = (V, E, c)$, the MMC problem is

$$\min_{\text{cycle } \sigma \text{ in } G} \frac{1}{|\sigma|} \sum_{e \in \sigma} c(e)\,. \tag{2}$$

MMC is a classical problem in algorithmic graph theory which has received significant attention over the past half century due to its many applications. A canonical textbook example is that, at least in an idealized world, finding arbitrage opportunities on Wall Street is equivalent to solving an MMC problem [25, §24]. Other classical applications range from periodic optimization (e.g., MMC is equivalent to finding an optimal policy for a deterministic Markov Decision Process [59]), to algorithm design (e.g., MMC provides a tractable alternative to the bottleneck step in the Network Simplex algorithm [34]), to control theory (e.g., MMC characterizes the spectral quantities in Max-Plus algebra [20]).

Just as for OT, a central challenge for MMC is large-scale computation. The first polynomial-time algorithm[1] was based on dynamic programming, due to Karp in 1972 [38]. However, its runtime is $O(n^3)$, and an extensive literature has sought to reduce this cubic runtime in both theory and practice; see e.g., the references within the recent papers [11, 22, 33].

2 Graphs, matrices, and distributions

As written, the optimization problems (1) and (2) defining OT and MMC appear quite different. Here we describe reformulations that are strikingly parallel. This hinges on a simple but useful connection between graph flows, non-negative matrices, and joint distributions, as summarized in Table 2. Below, let C denote the $n \times n$ matrix whose ij-th entry is the cost $c(i, j)$ if (i, j) is an edge, and ∞ otherwise.

OT is linear optimization over joint distributions with fixed marginals. Consider a feasible flow for the OT problem in (1), i.e., a flow $f : E \to \mathbb{R}_{\geqslant 0}$ routing the supply distribution μ to the demand distribution ν. This flow is naturally associated with a matrix $P \in \mathbb{R}^{n \times n}_{\geqslant 0}$ whose ij-th entry is the flow $f(i, j)$ on that edge. The netflow constraints on f then simply amount to constraints on the row and column sums of P, namely $P\mathbf{1} = \mu$ and $P^T\mathbf{1} = \nu$. Thus, OT can be re-written as the LP

$$\min_{P \in \Delta_{n \times n} \,:\, P\mathbf{1} = \mu,\ P^T\mathbf{1} = \nu} \langle P, C \rangle\,. \tag{3}$$

This decision set $\{P \in \Delta_{n \times n} : P\mathbf{1} = \mu,\ P^T\mathbf{1} = \nu\}$ is called the transportation polytope and can be equivalently viewed as the space of "couplings"—a.k.a., joint distributions $P(x, y)$ with first marginal μ and second marginal ν.

MMC is linear optimization over joint distributions with symmetric marginals. MMC admits a similar formulation by taking an LP relaxation. Briefly, the idea is to re-write the objective in terms of matrices, as above, and then take the convex hull of the discrete decision set. Specifically, re-

[1] It is worth remarking that MMC is polynomial-time solvable while the seemingly similar problem of finding a cycle with minimum (total) weight is NP-hard [52, §8.6b].

write the objective $\frac{1}{|\sigma|}\sum_{e\in\sigma} c(e)$ as $\langle P_\sigma, C\rangle$ by associating to a cycle σ the $n\times n$ matrix P_σ with ij-th entry $1/|\sigma|$ if the edge $(i,j)\in\sigma$, and zero otherwise. By the Circulation Decomposition Theorem [16, Problem 7.14], the convex hull of the set $\{P_\sigma : \sigma \text{ cycle}\}$ of normalized cycles is the set $\{P\in\Delta_{n\times n} : P\mathbf{1} = P^T\mathbf{1}\}$ of normalized circulations, and so the LP relaxation of MMC is

$$\min_{P\in\Delta_{n\times n}\,:\,P\mathbf{1}=P^T\mathbf{1}} \langle P, C\rangle, \tag{4}$$

and moreover this LP relaxation is exact. For details see, e.g., [3, Problem 5.47]. This decision set $\{P\in\Delta_{n\times n} : P\mathbf{1} = P^T\mathbf{1}\}$ can be equivalently viewed as the space of "self-couplings"—a.k.a., joint distributions $P(x,y)$ with symmetric marginals.

3 Entropic regularization and matrix pre-conditioning

Together, (3) and (4) put OT and MMC on equal footing in that they are both LP over spaces of joint distributions $P(x,y)$ with constrained marginals—fixed marginals for OT, and symmetric marginals for MMC. We now move from problem formulation to algorithm development, continuing in a parallel way.

The approach discussed in this note, motivated by the interpretation of OT and MMC as optimizing distributions, is to use entropic regularization. Namely, add $-\eta^{-1}H(P)$ to the objectives in (3) and (4), where $H(P) = \sum_{ij} P_{ij}\log P_{ij}$ denotes the Shannon entropy of P. See Tables 3 and 4, bottom left. This regularization is convex because the entropy function is concave (in fact, strongly concave by Pinsker's inequality [19, Section 4.3]). The regularization parameter $\eta > 0$ has a natural tradeoff: intuitively, smaller η makes the regularized problem "more convex" and thus easier to optimize, but less accurate for the original problem.

But let's step back. Why use entropic regularization? The modern optimization toolbox has many other convex regularizers. The key benefit of entropic regularization is that it reduces the problems of OT and MMC to the problems of Matrix Scaling and Matrix Balancing, respectively. This enables the application of classical algorithms for the latter two problems to the former two problems. Below we introduce these two matrix pre-conditioning problems in §3.1 and then explain this reduction in §3.2.

3.1 Matrix pre-conditioning

We now introduce the final two characters in our story: Matrix Scaling and Matrix Balancing. In words, these two problems seek to left- and right-multiply a given matrix K by diagonal matrices in order to satisfy certain marginal constraints—fixed marginals for Matrix Scaling, and symmetric marginals for Matrix Balancing. For the former, the scaling is of the form XKY; for the latter, it is a similarity transform XKX^{-1}.

Definition 3 (Matrix Scaling)**.** Given $K\in\mathbb{R}^{n\times n}_{>0}$ and $\mu,\nu\in\Delta_n$, find positive diagonal matrices X, Y such that $P = XKY$ has marginals $P\mathbf{1} = \mu$ and $P^T\mathbf{1} = \nu$.

Definition 4 (Matrix Balancing)**.** Given $K\in\mathbb{R}^{n\times n}_{>0}$, find a positive diagonal matrix X such that $P = XKX^{-1}$ has symmetric marginals $P\mathbf{1} = P^T\mathbf{1}$.

Both problems are defined here in a simplified way that suffices for the purposes of this note. See the discussion section for details.

Matrix Scaling and Matrix Balancing are classical problems in their own right and have been studied in many communities over many decades under many names. See the review [37] for a historical account. The most famous application of these problems is their use as pre-conditioning subroutines before numerical linear algebra computations [54, 43, 48]. For example, Matrix Balancing is nowadays used by default before eigenvalue decomposition and matrix exponentiation in standard numerical packages such as R, MATLAB, Lapack, and Eispack.

3.2 Reduction

As alluded to above, entropic regularization leads to the following reductions. Below, $K = \exp[-\eta C]$ denotes the matrix with entries $K_{ij} = \exp(-\eta C_{ij})$. For simplicity, assume henceforth that G is complete so that K is strictly positive, which ensures existence and uniqueness of the Matrix Scaling/Balancing solutions P. The general case is similar but requires combinatorial properties of the sparsity pattern [30, 50].

Lemma 3.1 (Entropic OT is Matrix Scaling)**.** *For any $\eta > 0$, the entropic OT problem has a unique solution. It is the solution $P = XKY$ to the Matrix Scaling problem for $K = \exp[-\eta C]$.*

Lemma 3.2 (Entropic MMC is Matrix Balancing)**.** *For any $\eta > 0$, the entropic MMC problem has a unique solution. It is the solution $P = XKX^{-1}$ to the Matrix Balancing problem for $K = \exp[-\eta C]$, modulo normalizing P by a constant so that the sum of its entries is 1.*

Both lemmas are immediate from first-order optimality conditions and are classical facts that have

	Primal	Dual
Optimal Transport	$\min_{P\in\Delta_{n\times n}\,:\,P\mathbf{1}=\mu,P^T\mathbf{1}=\nu}\langle P,C\rangle$	$\max_{x,y\in\mathbb{R}^n}\min_{ij}(C_{ij}-x_i-y_j)+\langle\mu,x\rangle+\langle\nu,y\rangle$
Matrix Scaling	$\min_{P\in\Delta_{n\times n}\,:\,P\mathbf{1}=\mu,P^T\mathbf{1}=\nu}\langle P,C\rangle-\frac{1}{\eta}H(P)$	$\max_{x,y\in\mathbb{R}^n}\operatorname{smin}_{ij}(C_{ij}-x_i-y_j)+\langle\mu,x\rangle+\langle\nu,y\rangle$

Table 3: Primal/dual LP formulations of OT (top) and its regularization (bottom). The regularization is entropic in the primal and softmin smoothing in the dual. The regularized problem is a convex formulation of the Matrix Scaling problem for the matrix $K=\exp[-\eta C]$.

	Primal	Dual
Minimum Mean Cycle	$\min_{P\in\Delta_{n\times n}\,:\,P\mathbf{1}=P^T\mathbf{1}}\langle P,C\rangle$	$\max_{x\in\mathbb{R}^n}\min_{ij}C_{ij}+x_i-x_j$
Matrix Balancing	$\min_{P\in\Delta_{n\times n}\,:\,P\mathbf{1}=P^T\mathbf{1}}\langle P,C\rangle-\frac{1}{\eta}H(P)$	$\max_{x\in\mathbb{R}^n}\operatorname{smin}_{ij}C_{ij}+x_i-x_j$

Table 4: Analog to Table 3 in the setting of symmetric marginals rather than fixed marginals. The story is mirrored, with OT and Matrix Scaling replaced by MMC and Matrix Balancing, respectively.

been re-discovered many times; see [37] for a historical account. There is also an elegant dual interpretation. Briefly, entropic regularization in the primal is equivalent to softmin smoothing in the dual, i.e., replacing $\min_i a_i$ by $\operatorname{smin}_i a_i := -\eta^{-1}\log\sum_i\exp(-\eta a_i)$ when writing the dual LP in saddle-point form. See Tables 3 and 4, bottom right. Modulo a simple transformation, the optimal scaling matrices X,Y correspond to the optimal solutions to these dual regularized problems—a fact that will be exploited and explained further below.

4 Simple scalable algorithms

Since entropic OT and entropic MMC are respectively equivalent to Matrix Scaling and Matrix Balancing (Lemmas 3.1 and 3.2), it suffices to solve the latter two problems. For both, there is a simple algorithm that has long been the practitioner's algorithm of choice. For Matrix Scaling, this is the Sinkhorn algorithm; for Matrix Balancing, this is the Osborne algorithm. These algorithms have been re-invented many times under different names, see the survey [37, §3.1]. Pseudocode is in Algorithms 1 and 2. For shorthand, we write $r(P)=P\mathbf{1}$ and $c(P)=P^T\mathbf{1}$ to denote row and column sums, and write $\odot$ and ./ to denote entrywise multiplication and division. We additionally write $\mathbb{D}(v)$ to denote the diagonal matrix whose diagonal is the vector v.

Both algorithms have natural geometric interpretations as alternating projection in the primal and coordinate descent in the dual. We explain both interpretations as they provide complementary insights.

Primal interpretation: alternating projection. The most direct interpretation of the Sinkhorn algorithm is that it alternately projects[2] the current matrix $P=XKY$ onto either the subspace $\{P\in\Delta_{n\times n}:P\mathbf{1}=\mu\}$ with correct row marginals, or the subspace $\{P\in\Delta_{n\times n}:P^T\mathbf{1}=\nu\}$ with correct column marginals. Note that when it corrects one constraint, it potentially violates the other. Nevertheless, the algorithm converges to the unique solution at the subspaces' intersection, see Figure 1 for a cartoon illustration. The Osborne algorithm is analogous, except that it alternately projects $P=XKX^{-1}$ onto n subspaces: the subspaces $\{P\in\Delta_{n\times n}:(P\mathbf{1})_i=(P^T\mathbf{1})_i\}$ defined by equal i-th row and column sums, for all $i\in[n]$. Here there is a choice for the order of subspaces to project onto; see [13] for a detailed discussion of this.

Algorithm 1 Sinkhorn's Algorithm for scaling a matrix K to have marginals μ,ν. To solve OT to $\pm\varepsilon$, run on $K=\exp[-\eta C]$ where $\eta\approx\varepsilon^{-1}\log n$.

```
Initialize X, Y ← I                          No scaling
Until convergence:
    X ← X ⊙ 𝔻(μ./r(XKY))                     Fix rows
    Y ← Y ⊙ 𝔻(ν./c(XKY))                     Fix columns
```

Algorithm 2 Osborne's Algorithm for balancing a matrix K to have symmetric marginals. To solve MMC to $\pm\varepsilon$, run on $K=\exp[-\eta C]$ where $\eta\approx\varepsilon^{-1}\log n$.

```
Initialize X ← I                             No balancing
Until convergence:
    Choose coordinate i ∈ [n] to fix
    X_ii ← X_ii · sqrt(c_i(XKX^{-1})/r_i(XKX^{-1}))
```

Dual interpretation: coordinate descent. The Sinkhorn algorithm also admits an appealing dual interpretation. In the dual, entropic OT has $2n$ variables—the Lagrange multipliers $x,y\in\mathbb{R}^n$ respec-

[2] This projection is to the closest point in KL divergence rather than Euclidean distance. In the language of information geometry, this is an I-projection.

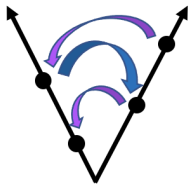


Figure 1: In the primal, the Sinkhorn algorithm alternately projects the iterate $P = XKY \in \mathbb{R}^{n\times n}$ onto the two subspaces corresponding to the row/column marginal constraints. The Osborne algorithm is analogous, but with $P = XKX^{-1}$ and n subspaces.

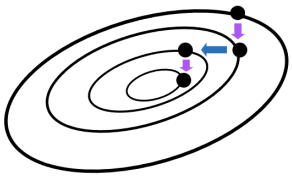


Figure 2: In the dual, the Sinkhorn algorithm performs exact block coordinate descent by iteratively updating the scaling matrices X or Y so as to optimally improve the dual objective given that all other entries are fixed. The Osborne algorithm is analogous but updates individual entries of X rather than blocks.

tively corresponding to the row and column marginal constraints in the primal. See Table 3, bottom right. Via the transformation $X_{ii} = e^{\eta x_i}$ and $Y_{ii} = e^{\eta y_i}$, these $2n$ dual variables are in correspondence with the $2n$ diagonal entries of the scaling matrices X and Y that the Sinkhorn algorithm seeks to find. One can verify that the Sinkhorn algorithm's row update (Line 3 of Algorithm 1) is an exact block coordinate descent step that maximizes the dual regularized objective over all $x \in \mathbb{R}^n$ given that y is fixed; vice versa for the column update. See Figure 2 for a cartoon illustration. The Osborne algorithm is analogous, except that now there are only n dual variables $x \in \mathbb{R}^n$ since there are only n marginal constraints in the primal, see Table 4, bottom right. When the Osborne algorithm updates $X_{ii} = e^{\eta x_i}$ to equate the i-th entry of the row and column marginals, this corresponds to an exact coordinate descent step on x_i.

5 Runtime analysis

We now turn to convergence analysis. The key challenge is how to measure progress. Indeed, an iteration of the Sinkhorn algorithm corrects either the row or column marginals, but messes up the others. Similarly, an iteration of the Osborne algorithm corrects one row-column pair, but potentially messes up the $n-1$ others. From the cartoons in Figures 1 and 2, we might hope to make significant progress in each iteration—but how do we quantify this?

There's an entire literature on this question – or at least for Matrix Scaling; for Matrix Balancing, things were much less clear until quite recently: even polynomial convergence was unknown for half a century until the breakthrough paper [44], let alone near-linear time convergence [13]. In the 1980s, Franklin and Lorenz established that Sinkhorn iterations contract in the Hilbert projective metric [31]; however, the contraction rate incurs large factors of n, which leads to similarly large factors of n in the final runtime. Another approach uses auxiliary Lyapunov functions to measure progress, for example the permanent of the scaled matrix [39]; however this too incurs extraneous factors of n. See the survey [37].

As it turns out, there is a short, simple, and strikingly parallel analysis approach that leads to runtimes for both the Sinkhorn and Osborne algorithms that scale in the dimension n as $\tilde{O}(n^2)$ [13, 14]. These are near-optimal[3] runtimes in n in the sense that in the absence of further structure, it takes $\Theta(n^2)$ time to even read the input for any of the matrix/graph problems discussed in this note, let alone solve them.

At a high level, this analysis hinges on using the regularized dual objective as a Lyapunov function (an idea dating back to the 1990s for Matrix Scaling [35]), and observing that each iteration of the Osborne/Sinkhorn algorithm significantly improves this Lyapunov function by an amount related to how violated the marginal constraints are for the current iterate P ("progress lemma"). In its simplest form, the analysis argues that if the current iterate has very violated marginal constraints, then the Lyapunov function improves significantly; and since the Lyapunov function is bounded within a small range, this can only continue for a small number of iterations before we arrive at an iterate with reasonably accurate marginals—and this must be a reasonably accurate solution.

The progress lemma is itself composed of two steps, both of which leverage the probabilistic perspective highlighted in this note. The first step is a direct calculation which shows that an iteration improves the dual objective by the current imbalance between the marginal distributions as measured in the KL divergence for Sinkhorn, or the Hellinger divergence for Osborne. The second step uses a probabilistic inequality to analyze this imbalance. The

[3] The "near" in "near-optimal" refers to the logarithmic factors suppressed by the $\tilde{O}$ notation.

point is that since probabilistic inequalities apply for (infinite-dimensional) continuous distributions, they are independent of the dimension n. Operationally, this allows switching between the dual (where progress is measured) and the primal (where solutions are desired) without incurring factors of n, thereby enabling a final runtime without extraneous factors of n. Full details can be found in [14, 29] for the Sinkhorn algorithm and [13] for the Osborne algorithm.

6 Discussion

For each of the four problems in this note, much more is known and also many questions remain. Here we briefly mention a few selected topics.

Algorithm comparisons and the nuances therein. Each optimization problem in this note has been studied for many decades by many communities, which as already mentioned, has led to the development of many approaches besides the Sinkhorn and Osborne algorithms. Which algorithm is best? Currently there is no consensus. We suspect that the true answer is nuanced because different algorithms are often effective for different types of problem instances. For example, specialized combinatorial solvers blow competitors out of the water for small-to-medium problem sizes by easily achieving high accuracy solutions [28], whereas Sinkhorn and Osborne are the default algorithms in most numerical software packages for larger problems where moderate accuracy is acceptable. Both parameter regimes are important, but typically for different application domains. For example, high precision may be relevant for safety-critical or scientific computing applications. Whereas the latter regime of moderate-accuracy solutions for large-scale problems is typically relevant for modern data-science applications of OT, since there is no need to solve beyond the inherent modeling error (OT is usually just a proxy loss in machine learning applications) and discretization error (μ, ν are often thought of as samples from underlying true distributions). The latter regime is also relevant for pre-conditioning matrices before eigenvalue computation, since Matrix Scaling/Balancing optimize objectives that are just proxies for fast convergence of downstream eigenvalue algorithms [43, 48].

Theory vs practice. Comparisons between algorithms are further muddled by discrepancies between theory and practice. Sometimes algorithms are more effective in practice than our best theoretical guarantees suggest—is this because current analysis techniques are lacking, or because the input is an easy benchmark, or because of practical considerations not captured by a runtime theorem? For instance, the Sinkhorn algorithm is "embarrasingly parallelizable" and interfaces well with modern GPU hardware [27]. On the flip-side, some algorithms with incredibly fast theoretical runtimes are less practical due to large constant or polylogarithmic factors hidden in the $\tilde{O}$ runtime. At least for now, this includes the elegant line of work (e.g., [56, 24, 4, 17]) based on the Laplacian paradigm, which very recently culminated in the incredible theoretical breakthrough [22] that solves the more general problem of minimum cost flow in time that is almost-linear in the input sparsity and polylogarithmic in $1/\varepsilon$. I am excited to see to what extent theory and practice are bridged in the upcoming years.

Exploiting structure. All algorithms discussed so far work for generic inputs. This robustness comes at an unavoidable $\Omega(n^2)$ cost in runtime/memory from just reading/storing the input. This precludes scaling beyond n in the several tens of thousands, say, on a laptop. For larger problems, it is essential to exploit "structure" in the input. What structure? This is a challenging question in itself because it is tied to the applications and pipelines relevant to practice. For OT, typically μ, ν are distributions over $\mathbb{R}^d$ and the cost C is given by pairwise distances, raised to some power $p \in [1, \infty)$. This is the p-Wasserstein distance, which plays an analogous role to the ℓ_p distance [57]. Then the $n \times n$ matrix C is implicit from the $2n$ points in μ, ν. In low dimension d, this at least enables reading the input in $O(n)$ time—can OT also be solved in $O(n)$ time? A beautiful line of work in the computational geometry community has worked towards this goal, a recent breakthrough being $n \cdot (\varepsilon^{-1} \log n)^{O(d)}$ runtimes for $(1 \pm \varepsilon)$ multiplicatively approximating OT for $p = 1$ [49, 1]. We refer to those papers for a detailed account of this literature and the elegant ideas therein. Low-dimensional structure can also be exploited by the Sinkhorn algorithm. The basic idea is that the n^2 runtime arises only through multiplying the $n \times n$ kernel matrix $K = \exp[-\eta C]$ by a vector—a well-studied task in scientific computing related to Fast Multipole Methods [15]—and this can be done efficiently if the distributions lie on low-dimensional grids [55], geometric domains arising in computer graphics [55], manifolds [5], or algebraic varieties [12]. Preliminary numerics suggest that in these structured geometric settings, Sinkhorn can scale to millions of data points n while maintaining reasonable accuracy [5]. Many questions remain

open in this vein, for example graceful performance degradation in the dimension d, instance-optimality, and average-case complexity for "real-world inputs".

Optimizing joint distibutions with many constrained marginals. This note focuses on optimizing joint distributions $P(x, y)$ with $k = 2$ constrained marginals, and it is natural to ask about $k \geqslant 2$. These problems are called Multimarginal OT in the case of fixed marginals and linear objectives, and arise in diverse applications in fluid dynamics [18], barycentric averaging [2], graphical models [36], distributionally robust optimization [23], and much more; see the monographs [47, 46, 42]. The setting of symmetric marginals arises in quantum chemistry via Density Functional Theory [26, 21]. A central challenge in all these problems is that in general, it is intractable even to store a k-variate probability distribution, let alone solve for the optimal one. Indeed, a k-variate joint distribution in which each variable takes n values corresponds to a k-order tensor that has n^k entries—an astronomical number even for tiny n, k, say $n, k = 20$. As such, a near-linear runtime in the size of P (the goal in this note for $k = 2$) is effectively useless for large k, and it is essential to go beyond this by exploiting structure via implicit representations. See part II of the author's thesis [6] and the papers upon which it is based [8, 10, 7, 9] for a systematic investigation of what structure enables $\text{poly}(n, k)$ time algorithms, and for pointers to the extensive surrounding literature.

Matrix Balancing in ℓ_p norms. The original papers [45, 43] studied Matrix Balancing in the following setting: given $K \in \mathbb{C}^{n\times n}$, find diagonal X such that the i-th row and column of XKX^{-1} have equal ℓ_p norm, for all $i \in [n]$. Definition 4 is equivalent for any finite p (which suffices for this note) and elucidates the connection to optimizing distributions. See [13, §1.4] for details. For the case $p = \infty$, similar algorithms have been developed and were recently shown to converge in polynomial time in the breakthrough work [53]. It would be interesting to reconcile the case $p = \infty$ as the analysis techniques there seem different and the connection to optimizing distributions seems unclear.

Entropic OT. In this note, entropically regularized OT was motivated as a means to an end for computing OT efficiently. However, it has emerged as an interesting quantity in its own right and is now used in lieu of OT in many applications due to its better statistical and computational properties [27, 32, 40], as well as its ability to interface with complex deep learning architectures [47]. Understanding these improved properties is an active area of research bridging optimization, statistics, and applications. We are not aware of an analogous study of entropic MMC and believe this may be interesting.

Acknowledgments. I am grateful to Jonathan Niles-Weed, Pablo Parrilo, and Philippe Rigollet for a great many stimulating conversations about these topics. This work was partially supported by NSF Graduate Research Fellowship 1122374 and a TwoSigma PhD Fellowship.

References

[1] Pankaj K Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. "Deterministic, near-linear ε-approximation algorithm for geometric bipartite matching". In: *Symposium on Theory of Computing*. 2022, pp. 1052–1065.

[2] Martial Agueh and Guillaume Carlier. "Barycenters in the Wasserstein space". In: *SIAM Journal on Mathematical Analysis* 43.2 (2011), pp. 904–924.

[3] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows*. Pearson, 1988.

[4] Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. "Much faster algorithms for matrix scaling". In: *Symposium on the Foundations of Computer Science*. 2017.

[5] Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. "Massively scalable Sinkhorn distances via the Nyström method". In: *Conference on Neural Information Processing Systems*. 2019, pp. 4429–4439.

[6] Jason M Altschuler. "Transport and beyond: efficient optimization over probability distributions". PhD thesis. MIT, 2022.

[7] Jason M Altschuler and Enric Boix-Adserà. "Hardness results for Multimarginal Optimal Transport problems". In: *Discrete Optimization* 42 (2021), p. 100669.

[8] Jason M Altschuler and Enric Boix-Adserà. "Polynomial-time algorithms for Multimarginal Optimal Transport problems with structure". In: *Math. Prog.* (2022), pp. 1–72.

[9] Jason M Altschuler and Enric Boix-Adserà. "Wasserstein barycenters are NP-hard to compute". In: *SIAM Journal on Mathematics of Data Science* 4.1 (2022), pp. 179–203.

[10] Jason M Altschuler and Enric Boix-Adserà. "Wasserstein barycenters can be computed in polynomial time in fixed dimension". In: *Journal of Machine Learning Research* 22 (2021), pp. 1–19.

[11] Jason M Altschuler and Pablo A Parrilo. "Approximating Min-Mean-Cycle for low-diameter graphs in near-optimal time and memory". In: *SIAM Journal on Optimization* 32.3 (2022), pp. 1791–1816.

[12] Jason M Altschuler and Pablo A Parrilo. "Kernel approximation on algebraic varieties". In: *Preprint at arXiv:2106.02755* (2021).

[13] Jason M Altschuler and Pablo A Parrilo. "Near-linear convergence of the Random Osborne algorithm for Matrix Balancing". In: *Mathematical Programming* (2022), pp. 1–35.

[14] Jason M Altschuler, Jonathan Weed, and Philippe Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration". In: *Conference on Neural Information Processing Systems*. 2017.

[15] Rick Beatson and Leslie Greengard. "A short course on fast multipole methods". In: *Wavelets, multilevel methods and elliptic PDEs* 1 (1997), pp. 1–37.

[16] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA, 1997.

[17] Jan van den Brand, Yin-Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. "Bipartite matching in nearly-linear time on moderately dense graphs". In: *Symposium on Foundations of Computer Science*. IEEE. 2020, pp. 919–930.

[18] Yann Brenier. "Generalized solutions and hydrostatic approximation of the Euler equations". In: *Physica D: Nonlinear Phenomena* 237.14-17 (2008), pp. 1982–1988.

[19] Sébastien Bubeck et al. "Convex optimization: Algorithms and complexity". In: *Foundations and Trends in Machine Learning* 8.3-4 (2015), pp. 231–357.

[20] Peter Butkovič. *Max-linear systems: theory and algorithms*. Springer Science & Business Media, 2010.

[21] Giuseppe Buttazzo, Luigi De Pascale, and Paola Gori-Giorgi. "Optimal-transport formulation of electronic density-functional theory". In: *Physical Review A* 85.6 (2012), p. 062502.

[22] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. "Maximum flow and minimum-cost flow in almost-linear time". In: *arXiv preprint arXiv:2203.00671* (2022).

[23] Louis Chen, Will Ma, Karthik Natarajan, David Simchi-Levi, and Zhenzhen Yan. "Distributionally robust linear and discrete optimization with marginals". In: *Operations Research* 70.3 (2022), pp. 1822–1834.

[24] Michael B Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. "Matrix Scaling and Balancing via Box Constrained Newton's Method and Interior Point Methods". In: *Symposium on the Foundations of Computer Science*. IEEE. 2017, pp. 902–913.

[25] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 2009.

[26] Codina Cotar, Gero Friesecke, and Claudia Klüppelberg. "Density functional theory and optimal transportation with Coulomb cost". In: *Communications on Pure and Applied Mathematics* 66.4 (2013), pp. 548–599.

[27] Marco Cuturi. "Sinkhorn distances: lightspeed computation of optimal transport". In: *Conference on Neural Information Processing Systems* 26 (2013), pp. 2292–2300.

[28] Yihe Dong, Yu Gao, Richard Peng, Ilya Razenshteyn, and Saurabh Sawlani. "A Study of Performance of Optimal Transport". In: *Preprint at arXiv:2005.01182* (2020).

[29] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. "Computational Optimal Transport: complexity by Accelerated Gradient Descent Is Better Than by Sinkhorn's Algorithm". In: *International Conference on Machine Learning*. 2018.

[30] B Curtis Eaves, Alan J Hoffman, Uriel G Rothblum, and Hans Schneider. "Line-sum-symmetric scalings of square nonnegative matrices". In: *Mathematical Programming* 25 (1985), pp. 124–141.

[31] Joel Franklin and Jens Lorenz. "On the scaling of multidimensional matrices". In: *Linear Algebra and its Applications* 114 (1989), pp. 717–735.

[32] Aude Genevay, Lénaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. "Sample complexity of Sinkhorn divergences". In: *International Conference on Artificial Intelligence and Statistics*. 2019, pp. 1574–1583.

[33] Loukas Georgiadis, Andrew V Goldberg, Robert E Tarjan, and Renato F Werneck. "An experimental study of minimum mean cycle algorithms". In: *Workshop on Algorithm Engineering and Experiments*. SIAM. 2009, pp. 1–13.

[34] Andrew V Goldberg and Robert E Tarjan. "Finding minimum-cost circulations by canceling negative cycles". In: *Journal of the ACM* 36.4 (1989), pp. 873–886.

[35] Leonid Gurvits and Peter N Yianilos. "The deflation-inflation method for certain semidefinite programming and maximum determinant completion problems". In: Technical report, NECI, 1998.

[36] Isabel Haasler, Rahul Singh, Qinsheng Zhang, Johan Karlsson, and Yongxin Chen. "Multi-marginal optimal transport and probabilistic graphical models". In: *IEEE Transactions on Information Theory* (2021).

[37] Martin Idel. "A review of matrix scaling and Sinkhorn's normal form for matrices and positive maps". In: *Preprint at arXiv:1609.06349* (2016).

[38] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of Computer Computations*. Springer, 1972, pp. 85–103.

[39] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. "A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents". In: *Symposium on Theory of Computing*. 1998, pp. 644–652.

[40] Gonzalo Mena and Jonathan Niles-Weed. "Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem". In: *Conference on Neural Information Processing Systems*. 2019, pp. 4541–4551.

[41] Gaspard Monge. "Mémoire sur la théorie des déblais et des remblais". In: *Histoire de l'Académie Royale des Sciences de Paris* (1781).

[42] Karthik Natarajan. *Optimization with Marginals and Moments*. Dynamic Ideas LLC, 2021.

[43] EE Osborne. "On pre-conditioning of matrices". In: *Journal of the ACM* 7.4 (1960), pp. 338–345.

[44] Rafail Ostrovsky, Yuval Rabani, and Arman Yousefi. "Matrix balancing in L_p norms: bounding the convergence rate of Osborne's iteration". In: *Symposium on Discrete Algorithms*. SIAM. 2017, pp. 154–169.

[45] Beresford N Parlett and Christian Reinsch. "Balancing a matrix for calculation of eigenvalues and eigenvectors". In: *Numerische Mathematik* 13.4 (1969), pp. 293–304.

[46] Brendan Pass. "Multi-marginal optimal transport: theory and applications". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 49.6 (2015), pp. 1771–1790.

[47] Gabriel Peyré and Marco Cuturi. "Computational optimal transport". In: *Foundations and Trends in Machine Learning* 11.5-6 (2019), pp. 355–607.

[48] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press, 2007.

[49] Sharath Raghvendra and Pankaj K Agarwal. "A near-linear time ε-approximation algorithm for geometric bipartite matching". In: *Journal of the ACM* 67.3 (2020), pp. 1–19.

[50] Uriel G Rothblum and Hans Schneider. "Scalings of matrices which have prespecified row sums and column sums via optimization". In: *Linear Algebra and its Applications* 114 (1989), pp. 737–764.

[51] Michael H Schneider and Stavros A Zenios. "A comparative study of algorithms for matrix balancing". In: *Operations Research* 38.3 (1990), pp. 439–455.

[52] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003.

[53] Leonard J Schulman and Alistair Sinclair. "Analysis of a classical matrix preconditioning algorithm". In: *Journal of the ACM* 64.2 (2017), p. 9.

[54] Richard Sinkhorn. "Diagonal equivalence to matrices with prescribed row and column sums". In: *The American Mathematical Monthly* 74.4 (1967), pp. 402–405.

[55] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. "Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains". In: *ACM Transactions on Graphics* 34.4 (2015), pp. 1–11.

[56] Daniel A Spielman and Shang-Hua Teng. "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *Symposium on Theory of Computing*. 2004, pp. 81–90.

[57] Cédric Villani. *Topics in optimal transportation*. Graduate Studies in Mathematics 58. American Mathematical Society, 2003.

[58] Alan Geoffrey Wilson. "The use of entropy maximising models, in the theory of trip distribution, mode split and route split". In: *Journal of Transport Economics and Policy* (1969), pp. 108–126.

[59] Uri Zwick and Mike Paterson. "The complexity of mean payoff games on graphs". In: *Theoretical Computer Science* 158.1-2 (1996), pp. 343–359.

Randomized Numerical Linear Algebra for Optimization

Madeleine Udell
Stanford University
udell@stanford.edu
https://web.stanford.edu/~udell/

Zachary Frangella
Stanford University
zfran@stanford.edu

1 Introduction

Advances in randomized numerical linear algebra (randNLA) in the new millennium 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 the sum of a low rank matrix and a diagonal matrix 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.

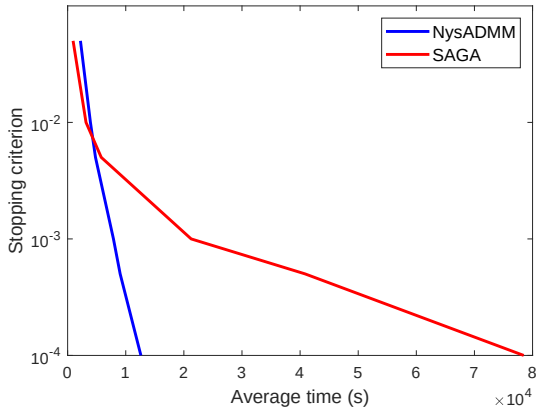


Figure 1: NysADMM vs. SAGA on ℓ^1-logistic regression with CIFAR-10 rf.

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 required to compute an orthonormal basis. The memory required is $O(ns)$, which is smaller than the memory required to store B if B is dense.

How well does the rangefinder work? Suppose $m \geqslant n$ and matrix $B \in \mathbf{R}^{m\times n}$ has singular values $\sigma_1 \geqslant \cdots \geqslant \sigma_n$. 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 $r \leqslant k$ 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 $Q^T B = \hat{U}\Sigma V^T$. 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

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

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 $\mathbf{E}\,\|A - \hat{A}\| \leqslant \epsilon$, then

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

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 the sum of a low rank matrix and an independent identically distributed (iid) Gaussian matrix. 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. Thus, 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 $\mu \geqslant 0$. 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 ($n \leqslant 50,000$), 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 $\lambda_s \leqslant \epsilon\mu$ [12], where $\lambda_1 \geqslant \cdots \geqslant \lambda_n$ are the eigenvalues of A. Hence, sketch-and-solve 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 improve the condition number by solving a related system. For any *preconditioner* $P \succ 0$,

$$Ax = b \iff P^{-1/2}Ax = P^{-1/2}b$$
$$P^{-1/2}AP^{-1/2}z = P^{-1/2}b$$

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, that is, $\kappa(P^{-1/2}AP^{-1/2}) \ll \kappa(A)$. Common preconditioners include the Jacobi preconditioner $P = \mathbf{diag}(A)$; incomplete Cholesky preconditioners, 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 $x \in \mathbf{R}^n$,

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

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,

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

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

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

which bounds the condition number $\kappa(UR^{-1}) \leqslant \sqrt{\frac{1+\zeta}{1-\zeta}}$. In particular, setting $\zeta = \frac{1}{2}$, we have $\kappa(UR^{-1}) \leqslant \sqrt{3}$. 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.

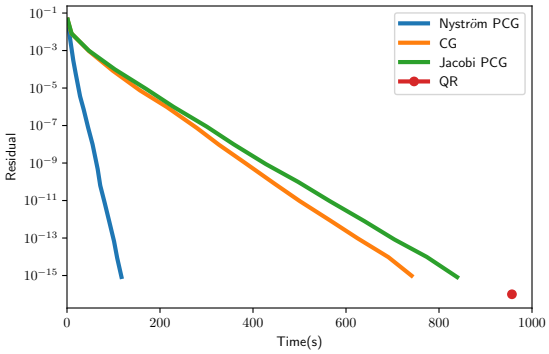


Figure 2: Nyström PCG is significantly faster than traditional NLA methods on YearMSD dataset with 15,000 random features.

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

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

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

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

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 $\geqslant \mu$:

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

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

Theorem 3.1. *Construct the randomized Nyström preconditioner P with rank $s = 2\lceil 1.5 d_{\mathrm{eff}}(\mu)\rceil + 1$. 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. Figure 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) works as well if not better [12]. See Figure 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 $\mathbf{prox}_r(x) = \arg\min_y r(y) + \frac{1}{2}\|x - y\|^2$ [24]. For example, for $r(x) = \|x\|_1$, $\mathbf{prox}_r(x)$ is the 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

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

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

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); see Algorithm 3 and [5] for a tutorial overview. In ADMM, the major computational challenge is the solution of an unconstrained minimization involving a large-scale data matrix (see Line 4 of Algorithm 3).

Algorithm 3 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: $\quad x^{k+1} = \operatorname{argmin}_x \{\ell(Ax) + \frac{\rho}{2}\|x - z^k + u^k\|_2^2\}$
5: $\quad z^{k+1} = \operatorname{argmin}_z \{r(z) + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2\}$
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 our present class of statistical learning problems using ideas from randNLA. To improve the runtime of ADMM, recall that *inexact ADMM*, which solves Line 4 approximately with error ε^k at iteration k, converges if $\sum_k \varepsilon^k < \infty$ [11]. To employ the randNLA toolbox, we approximate the solution of the problem in Line 4 by a quadratic optimization problem, 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

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

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 that the Hessian $A^T H_\ell(x^k) A$ involves the feature matrix A inside of it, and so generally exhibits fast spectral decay. Moreover, the stepsize ρ regularizes the 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 $\leqslant O(\log(1/\varepsilon^k))$ 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 by using a uniform sketch size $s = 50$. 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.

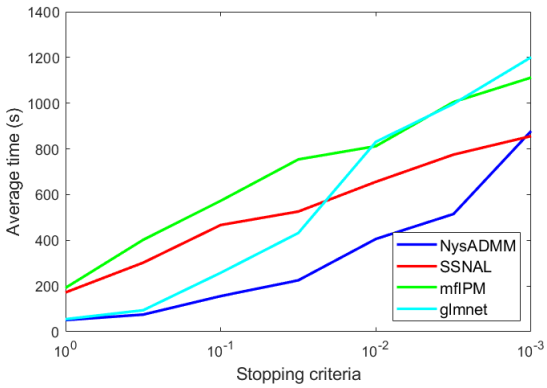


Figure 3: NysADMM outperforms other lasso solvers for moderate precision.

This simple paradigm yields substantial speedups over state-of-the-art solver; we saw in Figure 1 that NysADMM significantly outperforms SAGA on ℓ^1-logistic regression. NysADMM also delivers impressive numerical results on other problem classes as well. Figure 3 shows NysADMM outperforms standard solvers such as glmnet on a large Lasso problem instance ($m = 13,000, n = 27,648$), while in Figure 4 NysADMM runs almost 4× 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

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

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 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 $p = H^{-1} g$; the iterate is then updated as $x \leftarrow x - \eta p$

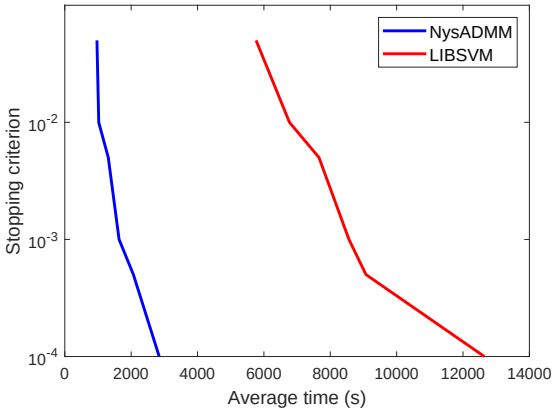


Figure 4: NysADMM outperforms LIBSVM on a kernelized SVM problem with CIFAR-10 rf.

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 $d_\star \coloneqq \sup_{x \in S(x_0)} d_{\text{eff}}^{\mu}(x)$, where $S(x_0) = \{x : F(x) \leqslant F(x_0)\}$ 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 $s = O(d_\star)$, the theory for the Nyström preconditioner in Section 3.1 guarantees we can solve the Newton system in a constant number of iterations. Moreover, under appropriate conditions this method exhibits superlinear convergence; see Figure 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

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

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 $O\left(T_{\text{mv}} d_\star \log(1/\epsilon)\right)$.

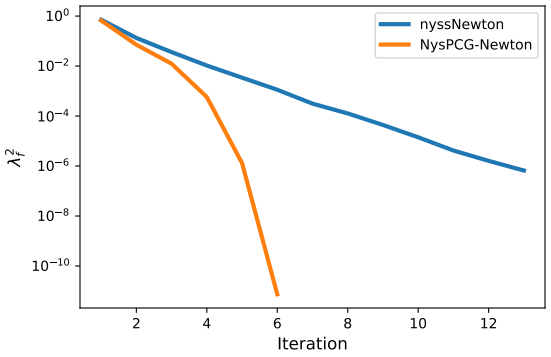


Figure 5: Convergence of the squared Newton decrement on an ℓ^2-logistic regression problem on the MNIST-rotated dataset with random features ($m = 62,000$, $n = 8,000$). NysPCG-Newton converges superlinearly on this problem, while nyssNewton converges linearly. Both methods were terminated when $\lambda_f^2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

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 algo-

rithm 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 (IPMs) 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 a signed permutation matrix, F is a discrete trigonometric 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 many coordinates uniformly at random. The cost of storing a RTT is $O(n\log(n))$, while computing $A\Omega$ costs only $O(n^2\log(k))$: this is 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: most implementations, e.g., the one in Matlab, exhibit complexity in $O(n^2\log(n))$ rather than in $O(n^2\log(k))$.

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 nonzero 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 $O(\mathrm{nnz}(A))$ time: much faster than the $O(n^2k)$ 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 those of RTTs: they require careful implementation to extract their full computational benefits, particularly in the distributed setting.

For more discussion, see the survey [22, Sections 9 and 10]; 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, the ONR Young Investigator Program 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". In: *The Annals of Statistics* 40.5 (2012), pp. 2452–2482.

[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". In: *SIAM Journal on Scientific Computing* 32.3 (2010), pp. 1217–1236.

[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". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[6] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), pp. 1–27.

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

[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*. 2013, pp. 81–90.

[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". In: *SIAM Journal on Computing* 36.1 (2006), pp. 158–183.

[11] Jonathan Eckstein and Dimitri P Bertsekas. "On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators". In: *Mathematical Programming* 55.1 (1992), pp. 293–318.

[12] Zachary Frangella, Joel A. Tropp, and Madeleine Udell. "Randomized Nyström Preconditioning". In: *arXiv preprint arXiv:2110.02820* (2021).

[13] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent". In: *Journal of Statistical Software* 33.1 (2010), p. 1.

[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". In: *SIAM Journal on Matrix Analysis and Applications* 36.4 (2015), pp. 1660–1690.

[16] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM Review* 53.2 (2011), pp. 217–288.

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

[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". In: *Entropy* 19.11 (2017), p. 617.

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

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

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

[23] Xiangrui Meng, Michael A Saunders, and Michael W Mahoney. "LSRN: A parallel iterative solver for strongly over-or underdetermined systems". In: *SIAM Journal on Scientific Computing* 36.2 (2014), pp. C95–C118.

[24] Neal Parikh and Stephen Boyd. "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[25] Mert Pilanci and Martin J Wainwright. "Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1842–1879.

[26] Mert Pilanci and Martin J Wainwright. "Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence". In: *SIAM Journal on Optimization* 27.1 (2017), pp. 205–245.

[27] Vladimir Rokhlin and Mark Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression". In: *Proceedings of the National Academy of Sciences* 105.36 (2008), pp. 13212–13217.

[28] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. "Falkon: An optimal large scale kernel method". In: *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". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9830–9840.

[30] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. "Fixed-rank approximation of a positive-semidefinite matrix from streaming data". In: *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". In: *SIAM Journal on Scientific Computing* 41.4 (2019), A2430–A2463.

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

[33] David P Woodruff. "Sketching as a Tool for Numerical Linear Algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157.

[34] Fan Yang, Sifan Liu, Edgar Dobriban, and David P Woodruff. "How to reduce dimension with PCA and random projections?" In: *IEEE Transactions on Information Theory* 67.12 (2021), pp. 8154–8189.

[35] Haishan Ye, Luo Luo, and Zhihua Zhang. "Approximate Newton Methods". In: *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*. PMLR. 2019, pp. 7272–7281.

[37] Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. "Scalable semidefinite programming". In: *SIAM Journal on Mathematics of Data Science* 3.1 (2021), pp. 171–200.

[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*. PMLR. 2017, pp. 1188–1196.

[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.

High-dimensional Optimization

Courtney Paquette
Google Research, Brain Team
courtney.paquette@mcgill.ca
https://cypaquette.github.io

Elliot Paquette
Department of Mathematics and Statistics
McGill University, Montreal, QC
elliot.paquette@mcgill.ca
https://elliotpaquette.github.io

1 Introduction

The structure of an optimization problem plays an important role in designing efficient algorithms. A common structure, motivated by empirical risk minimization (ERM), is the finite sum, that is, an optimization problem of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ f(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}) \right\}, \tag{1}$$

where the functions $f_i : \mathbb{R}^d \to \mathbb{R}$. Much has been written about the complexity of stochastic and deterministic algorithms for solving (1) under various general assumptions on f, such as smoothness and convexity [49, 11, 25, 15, 26, 10, 21, 7, 30, 29, 42, 19, 20, 31, 50, 40].

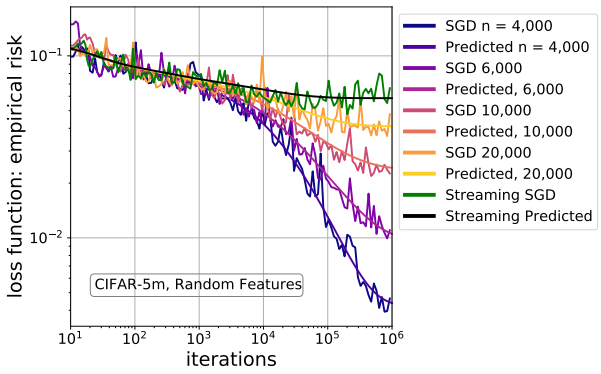


Figure 1: Single runs of SGD vs. predicted dynamics (solid line) on standardized CIFAR-5M [41] with car/plane class vector (1,000,000 samples); a standardized ReLU, random features model was applied with increasing number of samples n and fixed $d = 6000$. The predicted behavior, denoted by "predicted" (solid lines), without running SGD, matches the performance of single runs of SGD for finite n and streaming ($n = \infty$). See details in [47].

Motivated in part by the rise in machine learning, optimization research has focused on weakening these assumptions, as the problems of interest are both nonconvex and nonsmooth. This has led to tight upper bounds on complexity that match information theoretic lower bounds for very general finite-sum problems [4, 43, 23, 24], and yet in spite of this, there exists an enormous gap between these theoretical guarantees and observed performance in machine learning.

Indeed, even in the smooth, convex setting, there is a missing component in our understanding of finite sum problems in machine learning. One possibility is simply the size of the finite sums. An overarching trend in machine learning is to scale problems up in terms of model parameter count and data set size (see, for example, the literature on scaling laws [32]). In short, machine learning problems are *high-dimensional*.

Another aspect of machine learning problems, besides that they are high dimensional, is that they are all stochastic: the data are random, the learning algorithms are random, and the model initialization is random. We propose that this trifecta of randomness combined with high-dimensionality are the missing structure from a theory of optimization for machine learning.

The purpose of this article is to survey the recent advances in developing a framework that incorporates high-dimensionality into analyzing stochastic learning algorithms on a ℓ^2-regularized least squares problem [46, 47, 35]. The main idea, discussed in detail in Section 2, is to import mathematical ideas commonly used in random matrix theory. The resulting framework yields predictions for learning curves that are amenable to analysis and often exactly reproduce the behavior seen by popular algorithms (*e.g.*, stochastic gradient descent (SGD) [49]) on real data sets (see *e.g.*, Figure 1 and Figure 4). Finally we illustrate how one can use these predictions to draw important insights on average-case complexity and parameter selections such as learning rate, momentum parameter, and batch size (Section 4).

The use of tools from high-dimensional probability and random matrix theory for simplifying the analysis of optimization algorithms is a relatively new area in the machine learning literature [14, 18, 9, 17, 39, 51]. It has been used to model phenomena that, up until this point, had only been observed in deep neural networks (e.g., double descent), but which through random matrix theory are revealed to be an artifact of high-dimensional data [8, 22, 28, 38, 36, 2, 1, 53]. Beyond this, statistical assumptions to reduce complexity of analyzing algorithms were notably used in the compressed sensing community (see, for example, [12, 44]).

2 Problem Set-Up

In this section, we develop ideas from random matrix theory for incorporating high-dimensionality into the analysis of learning algorithms. To formalize the analysis, we define the *ℓ^2-regularized least squares problem*:

$$\begin{aligned}\arg\min_{\boldsymbol{x}\in\mathbb{R}^d}\Big\{ f(\boldsymbol{x}) &\stackrel{\text{def}}{=} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2 + \frac{\delta}{2}\|\boldsymbol{x}\|^2 \\ &= \sum_{i=1}^{n} \underbrace{\frac{1}{2}\left((\boldsymbol{a}_i\boldsymbol{x}-b_i)^2 + \frac{\delta}{n}\|\boldsymbol{x}\|^2\right)}_{\stackrel{\text{def}}{=} f_i(\boldsymbol{x})}\Big\}. \end{aligned} \tag{2}$$

The design matrix $\boldsymbol{A}$ is of size $n \times d$, with the idea that both the data size (n) and the model complexity d are big. The fixed parameter $\delta > 0$ controls the regularization strength and it is independent of n and d. We do not require that n and d are proportional. Instead, we need the following:

Assumption 1 (Polynomially related)**.** There is an $\alpha \in (0,1)$ so that

$$d^{\alpha} \leqslant n \leqslant d^{1/\alpha}.$$

The data matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and the labels $\boldsymbol{b}$ may be deterministic or random; we formulate the theorems for deterministic $\boldsymbol{A}$ and $\boldsymbol{b}$ in (2) satisfying various

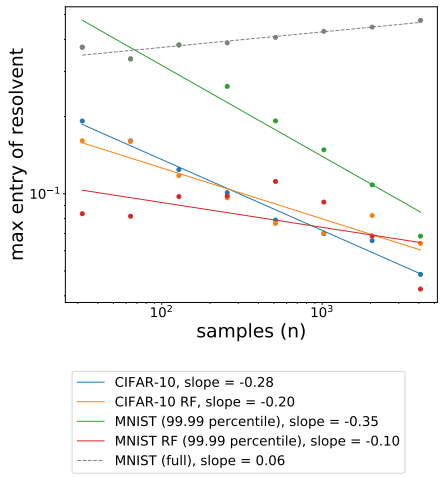


Figure 2: Maximum off-diagonal entry of the resolvent for CIFAR-5M [41] and MNIST [34] data sets with features d fixed (3072 and 784, respectively), varying samples $n = 2^k$ for $k = 5, 6, \cdots, 12$. Random features (RF) model was employed with $n_0 = 2000$. In the MNIST (99.99 percentile) data set large resolvent outliers were removed; when outliers not removed, MNIST data set does not satisfy the off-diagonal resolvent condition (5). For the other data sets, the off-diagonal resolvent condition is satisfied. The theory still works well for MNIST without modification (see Figure 4), which suggests that (5) could be weakened.

assumptions, and in the applications of these theorems to statistical settings, we shall give examples of random $\boldsymbol{A}$ and $\boldsymbol{b}$ which satisfy these assumptions. These assumptions are motivated by the empirical risk minimization (ERM) problem, and in particular the case where the augmented matrix $[\boldsymbol{A} \mid \boldsymbol{b}]$ has rows that are independent and sampled from some common distribution (see Section 2.1 for details). We also note that the problem (2) is homogeneous, in that if we simultaneously divide $\boldsymbol{A}$, $\boldsymbol{b}$ and $\sqrt{\delta}$ by any desired scalar, we produce an equivalent optimization problem. As such, we may also adopt the following normalization convention without loss of generality.

Assumption 2 (Data–target normalization)**.** There is a constant $C > 0$ independent of d and n such that the spectral norm of $\boldsymbol{A}$ is bounded by C and the target vector $\boldsymbol{b} \in \mathbb{R}^n$ is normalized so that $\|\boldsymbol{b}\|^2 \leqslant C$.

2.1 Detour into random matrix theory

We are looking for deterministic assumptions on $(\boldsymbol{A}, \boldsymbol{b})$ that capture the combination of the high-dimensionality of the problem with the intrinsic randomness seen in a machine learning setup.

As a motivating test case, consider the Gaussian design case $\boldsymbol{A} = \boldsymbol{Z}\sqrt{\boldsymbol{\Sigma}}$ for a covariance matrix $\boldsymbol{\Sigma}$. For the target, consider the generative model with noise $\boldsymbol{b} = \boldsymbol{A}\tilde{\boldsymbol{x}}_0 + \boldsymbol{\eta}$ for $\tilde{\boldsymbol{x}}_0$ and $\boldsymbol{\eta}$ independent of $\boldsymbol{A}$.

The Gaussian matrix $\boldsymbol{A}$ enjoys some spectacular distributional invariances. Most relevant here, it satisfies that for any $n \times n$ orthogonal matrix $\boldsymbol{O}$ the distribution of $\boldsymbol{OA}$ is the same as the distribution of $\boldsymbol{A}$. It follows as a consequence that in a singular value decomposition of $\boldsymbol{A} = \boldsymbol{U}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^T$, the matrix $\boldsymbol{U}$ is independent of $\boldsymbol{\Lambda}$ and $\boldsymbol{V}$ and moreover it can be taken uniformly distributed on the orthogonal group. For non-identity covariance $\boldsymbol{\Sigma}$, the same is not generally true of the matrix of right–singular vectors $\boldsymbol{V}$.

This means that the left singular vectors of $\boldsymbol{A}$ reveal nothing on either $\tilde{\boldsymbol{x}}$ or $\boldsymbol{\Sigma}$. It is, however, a type of optimization *structure*; and we shall further illustrate that this uniform distribution has profound consequences for the behavior of algorithms which operate on batch subproblems (in particular minibatch SGD).

On the other hand, this is far too much to ask for a general design $\boldsymbol{A}$, even for one with strong statistical assumptions such as independent identically distributed subgaussian rows. We would like to generalize this assumption and ideally identify a deterministic condition which captures some of the consequences of this uniform distribution of eigenvectors.

One of the key tools in random matrix theory, especially in the theory of *universality*[1], is the *resolvent* of a matrix $\boldsymbol{M}$ defined by

$$R(z; \boldsymbol{M}) = (z\mathbf{I} - \boldsymbol{M})^{-1} \quad z \in \mathbb{C} \setminus \sigma(\boldsymbol{M}), \tag{3}$$

with $\sigma(\boldsymbol{M})$ its spectrum.

Given a singular value decomposition for $\boldsymbol{A} = \boldsymbol{U}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^T$, this can be computed in terms of the singular vectors by

$$\begin{aligned} R(z; \boldsymbol{A}\boldsymbol{A}^T) &= \boldsymbol{U}(z\mathbf{I} - \boldsymbol{\Lambda})^{-1}\boldsymbol{U}^T \\ \text{and} \quad R(z; \boldsymbol{A}^T\boldsymbol{A}) &= \boldsymbol{V}(z\mathbf{I} - \boldsymbol{\Lambda})^{-1}\boldsymbol{V}^T. \end{aligned} \tag{4}$$

Hence in the case of the Gaussian design, the resolvent $R(z; \boldsymbol{A}\boldsymbol{A}^T)$ factorizes as a conjugation. Moreover $\boldsymbol{U}$ is independent of $\boldsymbol{\Lambda}$ and $\boldsymbol{U}$ is uniformly distributed over the orthogonal group.

[1] Universality is the property of random matrices by which eigenvalue and eigenvector statistics are common across all large matrices having a given first and second moment structure (and sometimes 3rd and 4th) in their entries. For example, sample covariance matrices, in which $\boldsymbol{A} = \boldsymbol{W}\sqrt{\boldsymbol{\Sigma}}$ for a matrix of iid mean 0 variance 1 entries, are known to have many common spectral properties.

Thus the off-diagonal and diagonal entries of $R(z; \boldsymbol{A}\boldsymbol{A}^T)$ can be estimated by

$$\begin{aligned} R(z; \boldsymbol{A}\boldsymbol{A}^T)_{ii} &\approx \frac{1}{n} \operatorname{\mathbf{tr}} R(z; \boldsymbol{\Lambda}) \\ \text{and} \quad |R(z; \boldsymbol{A}\boldsymbol{A}^T)_{ij}| &\lesssim n^{-1/2}. \end{aligned} \tag{5}$$

See Assumption 3 for a precise definition of $\sim$ and $\lesssim$. Further, these estimates hold with very high probability and uniformly over all i, j. The same does not hold for the other resolvent $R(z; \boldsymbol{A}^T\boldsymbol{A})$ on account of the non-uniform distribution of its singular vectors (save for in the special case that $\boldsymbol{\Sigma}$ is scalar). See Figure 2 for the behavior of the off-diagonal entries (5) on some popular data sets.

The property (5) generalizes to many other classes of random matrices with independent rows. This leads us to pose a family of assumptions on $\boldsymbol{A}$ and $\boldsymbol{b}$ which encode *some* of the flavor of the uniform distribution of left singular vectors of $\boldsymbol{A}$.

Assumption 3. Suppose Ω is a rectangle (contour) in the complex plane that encloses the spectrum of $\boldsymbol{A}\boldsymbol{A}^T$ with a diameter independent of n and d. Suppose there is a $\theta \in (0, \frac{1}{2})$ for which

1. $\max\limits_{z\in\Omega} \max\limits_{1\leqslant i\leqslant n} |\boldsymbol{e}_i^T R(z; \boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{b}| \leqslant n^{\theta-1/2}$.
2. $\max\limits_{z\in\Omega} \max\limits_{1\leqslant i\neq j\leqslant n} |\boldsymbol{e}_i^T R(z; \boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{e}_j^T| \leqslant n^{\theta-1/2}$.
3. $\max\limits_{z\in\Omega} \max\limits_{1\leqslant i\leqslant n} |\boldsymbol{e}_i^T R(z; \boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{e}_i - \frac{1}{n} \operatorname{\mathbf{tr}} R(z; \boldsymbol{A}\boldsymbol{A}^T)| \leqslant n^{\theta-1/2}$.

The latter two assumptions thus encode, in a weakened form, a consequence of the uniform distribution of left-singular vectors. The first assumption additionally adds the interaction of the data matrix with the target vector $\boldsymbol{b}$. For matrices $\boldsymbol{A}$ whose entries are standard Gaussians and targets $\boldsymbol{b}$ which come from some ground truth signal plus noise, Assumption 3 can easily be checked to hold. This assumption effectively shows that individual samples have a controlled influence on solving the minimization problem, which in some sense quantifies that we are dealing with a high-dimensional problem.

The relevance of the contour enclosing the spectrum is that this allows us to pass, by contour integration, from resolvent estimates to estimates of other matrix functions, such as the ones that appear in describing the trajectories of first order algorithms. For random matrices, there is rarely a special contour of importance, and moreover the complexity of checking that the bound on a contour is roughly the same as checking the bound holds at any z with a minimum separation from the spectrum of $\boldsymbol{A}\boldsymbol{A}^T$.

2.2 Algorithmic setup

Stochastic learning algorithms and their momentum variants are the workhorses in machine learning due to their relatively cheap computational cost and simple implementations.

We solve the ℓ^2-regularized least-squares problem (2) using stochastic learning algorithms, and in particular, stochastic gradient descent (SGD) with learning rate γ_k. For an initial vector $\boldsymbol{x}_0 \in \mathbb{R}^d$, we define a sequence of SGD iterates $\{\boldsymbol{x}_k\}_{k=0}^{\infty}$ which obey the recurrence,

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \gamma_k \nabla f_{i_k}(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - \gamma_k \boldsymbol{A}^T \boldsymbol{e}_{i_k} \boldsymbol{e}_{i_k}^T (\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}) - \tfrac{\gamma_k \delta}{n} \boldsymbol{x}_k \,. \end{aligned} \tag{6}$$

The rows $\{i_1, i_2, \dots\}$ are chosen uniformly at random, and thus the batch size is one. The work of [45] suggests that under similar (albeit more restrictive) assumptions, minibatch SGD with batch-size $\beta = o(n)$ produces the same dynamical behavior as SGD after sampling single-batch SGD at iteration counts $\beta\mathbb{N}$. Therefore, we content ourselves with the simpler case with batch size equal to one. In Section 4, we will explore the effects of large batches on SGD and its momentum variant, but for now, we consider only $\beta = o(n)$.

As we want to give descriptions of the dynamics of SGD which are consistent across increasing dimensions, we suppose that γ_k has a smoothly varying schedule. Specifically, we suppose:

Assumption 4. There is a continuous bounded function $\gamma : [0, \infty) \to [0, \infty)$ such that $\gamma_k = \gamma(k/n)$ for all k. As such

$$\widehat{\gamma} \stackrel{\text{def}}{=} \sup_t \gamma(t) < \infty.$$

Although the classic Robbins-Monro $\gamma_k = \frac{1}{k}$ does not technically fit into this framework, for problems in which Assumptions 2 and 3 are satisfied (or more generally where some non-trivial fraction of the samples are needed to commence learning), the classic $1/k$ rate is often too slow to produce any practically relevant results. Moreover, from a theoretical point of view, such a rate produces a behavior similar to gradient flow (see (10)), and it could be viewed as effectively non-stochastic. In our high-dimensional setting, a suitable analogue of the Robbins-Monro schedule that does satisfy our assumptions and yields nontrivial behavior is $\gamma_k = \frac{n}{n+k} = \frac{1}{1+k/n}$.

As for the initialization $\boldsymbol{x}_0$, we need to suppose that it does not interact too strongly with the *right* singular-vectors of $\boldsymbol{A}$. In the spirit of Assumption 3, it suffices to assume the following:

Assumption 5. Let Ω be the same contour as in Assumption 3 and let $\theta \in (0, \frac{1}{2})$. Then

$$\max_{z\in\Omega} \max_{1\leqslant i\leqslant d} |\boldsymbol{e}_i^T R(z; \boldsymbol{A}^T\boldsymbol{A})\boldsymbol{x}_0| \leqslant n^{\theta-1/2}.$$

Note that, as a simple but common case, this assumption is surely satisfied for $\boldsymbol{x}_0 = \boldsymbol{0}$. In principle, this assumption is general enough to allow for $\boldsymbol{x}_0$ which are correlated with $\boldsymbol{A}$ in a nontrivial way, but we do not have an application for such an initialization. For a large class of nonzero initializations independent from $(\boldsymbol{A}, \boldsymbol{b})$, Assumption 5 is satisfied, as a corollary of Assumption 3:

Lemma 2.1. *Suppose that Assumption 3 holds with some $\theta_0 \in (0, \frac{1}{2})$ and that $\boldsymbol{x}_0$ is chosen randomly, independent of $(\boldsymbol{A}, \boldsymbol{b})$, and with independent coordinates in such a way that for some C independent of d or n*

$$\|\mathbb{E}\,\boldsymbol{x}_0\|_\infty \leqslant C/n$$
$$\text{and} \quad \max_i \|(\boldsymbol{x}_0 - \mathbb{E}\,\boldsymbol{x}_0)_i\|_{\psi_2}^2 \leqslant Cn^{2\theta_0-1}.$$

Assumption 5 holds with any $\theta > \theta_0$ on an event of probability tending to 1 as $n \to \infty$.

Note that this assumption allows for deterministic $\boldsymbol{x}_0$ having maximum norm $\mathcal{O}(1/n)$, as well as iid centered subgaussian vectors of Euclidean norm $\mathcal{O}(1)$.

2.3 Examples of data matrix and target

We highlight two examples for which the data matrix $\boldsymbol{A}$ and target vector $\boldsymbol{b}$ satisfy Assumption 3 (proofs found in [47, Lemma 1.3 and Theorem 1.7]). In both cases below, we take the initialization vector $\boldsymbol{x}_0$ to be iid centered subgaussian with $\mathbb{E}[\|\boldsymbol{x}_0\|^2] = \widehat{R}$ for $\widehat{R} > 0$. Assumption 5 holds for this initialization vector.

Sample covariance matrices and generative models. Suppose that $\boldsymbol{\Sigma} \succeq 0$ is a $d\times d$ matrix with $\operatorname{tr}\boldsymbol{\Sigma} = 1$ and $\|\boldsymbol{\Sigma}\| \leq M/\sqrt{d} < \infty$. The data matrix $\boldsymbol{A}$ is a random matrix with $\boldsymbol{A} = \boldsymbol{Z}\sqrt{\boldsymbol{\Sigma}}$ where $\boldsymbol{Z}$ is an $n\times d$ matrix of independent, mean 0, variance 1 entries with subgaussian norm at most $M < \infty$, and we assume $n \leq Md$. Finally suppose that $\boldsymbol{b}$ satisfies a generative model, that is $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{\beta} + \boldsymbol{\eta}$ for $\boldsymbol{\beta}, \boldsymbol{\eta}$ iid centered subgaussian satisfying $\|\boldsymbol{b}\|^2 = R$ and $\|\boldsymbol{\eta}\|^2 = \tilde{R}\frac{n}{d}$ for some $\tilde{R}, R > 0$. For data matrix $\boldsymbol{A}$ and target vector generated this way, Assumption 3 holds.

Random features model of a linear ground truth. We follow the set-up based on [38, 1]. This model encompasses two-layer neural networks with a squared loss, where the first layer has random weights and the second layer's weights are given by the regression coefficients. Suppose the $n \times n_0$ data matrix $\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{\Sigma}^{1/2}/\sqrt{n_0}$ for an iid standard Gaussian $\boldsymbol{Z}$ and the covariance matrix $\boldsymbol{\Sigma}$ satisfies $1/n_0\,\mathbf{tr}(\boldsymbol{\Sigma}) = 1$ and $\|\boldsymbol{\Sigma}\| \leq C$ for some $C > 0$. We also suppose that $\boldsymbol{W}$ is an $n_0 \times d$ iid feature matrix having standard Gaussian entries and independent of $\boldsymbol{Z}$ so that $\boldsymbol{X}\boldsymbol{W}$ is a matrix whose rows are standardized. We now apply an activation function entry-wise. The activation function σ satisfies for $C_0, C_1 \geq 0$

$$\begin{gathered}|\sigma'(x)| \leq C_0 e^{C_1|x|}, \quad \text{for all } x \in \mathbb{R} \\ \text{and for all } Z \sim N(0,1), \quad \mathbb{E}[\sigma(Z)] = 0.\end{gathered} \tag{7}$$

We now transform the data $\boldsymbol{X} \in \mathbb{R}^{n_0\times d}$ by putting

$$\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{W}/\sqrt{n_0}) \in \mathbb{R}^{n\times d}.$$

For the target vector $\boldsymbol{b}$, we use a linear ground truth model, that is, $\boldsymbol{b} = \boldsymbol{X}\boldsymbol{\beta} + \eta\boldsymbol{w}$ with $\boldsymbol{\beta}, \boldsymbol{w}$ independent isotropic subgaussian vectors with $\mathbb{E}[\|\boldsymbol{\beta}\|^2] = 1/n_0$ and $\mathbb{E}[\|\boldsymbol{w}\|^2] = 1$ and η bounded, independent of n. Assumption 3 holds for $\boldsymbol{A}$ and $\boldsymbol{b}$ generated this way.

3 Predicting learning curves

A benefit of working in high-dimensional optimization is that seemingly challenging tasks such as understanding the noise produced by SGD, become much simpler due to concentration effects. In fact, the entire training path taken by SGD concentrates around a deterministic function. The function gives a simple description of the exact learning curve of SGD, depending only on the spectrum of $\boldsymbol{A}\boldsymbol{A}^T$, the target $\boldsymbol{b}$, and initial $\boldsymbol{x}_0$. In this way, one can predict the training behavior of SGD without ever running SGD. The idea hinges on exploiting the trifecta of randomness in the problem and high-dimensionality through concentration of measure. Moreover, these predictions are amenable to analysis and one can draw important insights on typical computational complexity and parameter selection policies (see Section 4).

In a high-dimensional setting, this empirical risk concentrates around a deterministic path Ψ_t. To define this path, we introduce the integrated learning rate Γ and kernel K, for any $d\times d$ matrix $\boldsymbol{P}$,

$$\begin{aligned}\Gamma(t) &= \int_0^t \gamma(s)\,\mathrm{d}s, \quad \text{and} \\ K(t,s;\boldsymbol{P}) &= \tfrac{1}{n}\gamma^2(s)\,\mathbf{tr}\Big(\boldsymbol{P}(\nabla^2\mathscr{L}) \\ &\qquad \times \exp\big(-2(\nabla^2\mathscr{L} + \delta\mathbf{I}_d)(\Gamma(t) - \Gamma(s))\big)\Big).\end{aligned} \tag{8}$$

The path Ψ_t satisfies the Volterra integral equation:

$$\Psi_t = \mathscr{L}\big(\mathscr{X}^{\mathrm{gf}}_{\Gamma(t)}\big) + \int_0^t K(t,s;\nabla^2\mathscr{L})\Psi_s\,\mathrm{d}s. \tag{9}$$

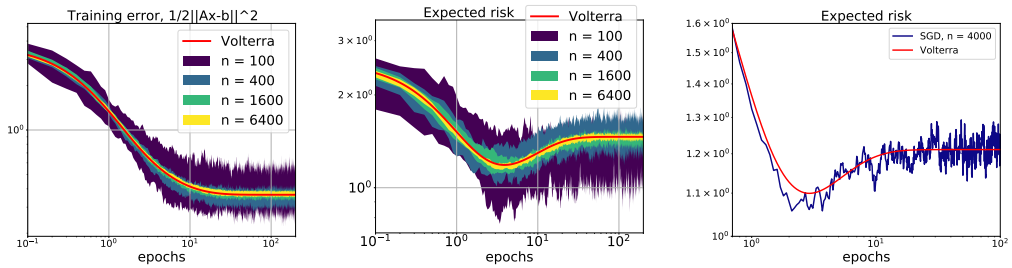


Figure 3: Concentration of SGD on training loss and expected risk, on a Gaussian random ℓ^2-regularized least-squares problem where $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{I}_d)$ is the ground truth signal and a generative model $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{\beta} + \boldsymbol{\eta}$ where entries of $\boldsymbol{\eta}$ iid standard normal with $\|\boldsymbol{\eta}\|_2^2 = 2.25$, $n = 0.9d$ with ℓ^2-regularization parameter $\delta = 0.1$, SGD was initialized at $\boldsymbol{x}_0 \sim N(\mathbf{0}, 4\mathbf{I}_d)$ (independent of $\boldsymbol{A}$, $\boldsymbol{\beta}$); an 80% confidence interval (shaded region) over 10 runs for each n, a constant learning rate for SGD was applied, $\gamma = 0.8$. For expected risk, the samples $\boldsymbol{a}$ generated from same covariance as $\boldsymbol{A}$. More volatility in the expected risk across runs even for large n in comparison to the training error (left and center). The predicted Ω_t matches the performance of SGD on the expected risk even for a single run (right).

The quantity $\mathscr{X}_t^{\mathrm{gf}}$ is *gradient flow* which is the solution to the differential equation

$$\mathrm{d}\mathscr{X}_t^{\mathrm{gf}} = -\nabla f(\mathscr{X}_t^{\mathrm{gf}})\,\mathrm{d}t, \quad \mathscr{X}_0^{\mathrm{gf}} = \boldsymbol{x}_0. \tag{10}$$

For the ℓ^2–regularized least squares problem, the solution to gradient flow is explicitly solvable in terms of $\boldsymbol{x}_0$, target $\boldsymbol{b}$, and eigenvalues of $\nabla^2\mathscr{L}$. Consequently, due to (9) and as Theorem 3.1 will show, the training dynamics of SGD are completely predictable solely from the spectrum of $\boldsymbol{A}^T\boldsymbol{A}$, target $\boldsymbol{b}$, and the initialization $\boldsymbol{x}_0$.

But one can do more. Generally, one wants to study not only the training dynamics but also the generalization performance of SGD, that is, how well the algorithm performs on unseen data. In this sense, we need to be able to evaluate the iterates of SGD, applied to f (2), on other statistics. We will focus our attention on quadratics.

Definition 3.1. *A function $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$ is quadratic if it is a degree-2 polynomial or equivalently if can be represented by*

$$\mathcal{R}(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{T}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{x} + c$$

for some $d \times d$ matrix $\boldsymbol{T}$, vector $\boldsymbol{u} \in \mathbb{R}^d$ and scalar $c \in \mathbb{R}$. For any quadratic, define the H^2–norm:

$$\begin{aligned}\|\mathcal{R}\|_{H^2} &\overset{def}{=} \|\nabla^2\mathcal{R}\| + \|\nabla\mathcal{R}(0)\| + |\mathcal{R}(0)| \\ &= \|\boldsymbol{T}\| + \|\boldsymbol{u}\| + |c|.\end{aligned} \tag{11}$$

For the learning path to concentrate on other quadratic statistics, we require an additional assumption in the same spirit as Assumption 3:

Assumption 6 (Quadratic statistics)**.** Suppose $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$ is quadratic, i.e. there is a symmetric matrix $\boldsymbol{T} \in \mathbb{R}^{d\times d}$, a vector $\boldsymbol{u} \in \mathbb{R}^d$, and a constant $c \in \mathbb{R}$ so that

$$\mathcal{R}(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{T}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{x} + c. \tag{12}$$

We assume that $\mathcal{R}$ satisfies $\|\mathcal{R}\|_{H^2} \le C$ for some C independent of n and d. Moreover, we assume the following (for the same Ω and θ) as in Assumption 3:

$$\max_{z,y\in\Omega}\max_{1\leqslant i\leqslant n} |\boldsymbol{e}_i^T\boldsymbol{A}\widehat{\boldsymbol{T}}\boldsymbol{A}^T\boldsymbol{e}_i - \tfrac{1}{n}\mathbf{tr}(\boldsymbol{A}\widehat{\boldsymbol{T}}\boldsymbol{A}^T)| \leqslant \|\boldsymbol{T}\|n^{-\epsilon}$$
$$\text{where} \quad \begin{cases}\widehat{\boldsymbol{T}} = R(z)\boldsymbol{T}R(y) + R(y)\boldsymbol{T}R(z), \\ R(z) = R(z;\boldsymbol{A}^T\boldsymbol{A})\end{cases} \tag{13}$$

This assumption ensures that the quadratic $\mathcal{R}$ has a Hessian which is not too correlated with any of the left singular–vectors of $\boldsymbol{A}$. Establishing Assumption 6 can be non–trivial in the cases when the quadratic has complicated dependence on $\boldsymbol{A}$. In simple cases, (especially for the case of the empirical risk and the norm) it follows automatically from Assumption 3.

Lemma 3.1. *Suppose that $\mathcal{R}$ satisfies (12) with $\boldsymbol{T} = p(\boldsymbol{A}^T\boldsymbol{A})$, where p is a polynomial having bounded coefficients, and suppose $\boldsymbol{u}$ and c are norm-bounded independently of n or d. Then supposing Assumptions 2 and 3 for some $\theta_0 \in (0, \frac{1}{2})$, for all n sufficiently large and for any $\theta > \theta_0$, Assumption 6 holds.*

Thus for example $\mathcal{R} = \mathscr{L}$ will satisfy Assumption 6, as will the simple Euclidean vector norm $\mathcal{R} = \|\cdot\|^2$.

The trajectory $\mathcal{R}(\boldsymbol{x}_t)$ concentrates around

$$\Omega_t = \mathcal{R}\big(\mathscr{X}_{\Gamma(t)}^{\mathrm{gf}}\big) + \int_0^t K(t,s;\nabla^2\mathcal{R})\Psi_s\,\mathrm{d}s. \tag{14}$$

Note the trajectories of gradient flows can be computed explicitly.

Finally the concentration theorem is the following:

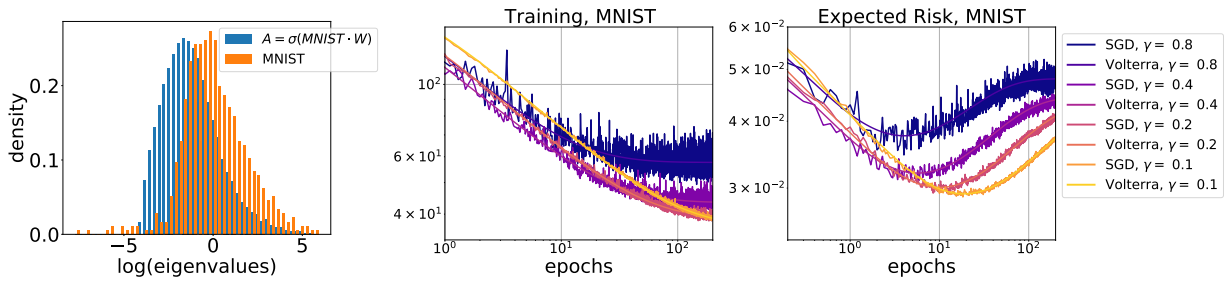


Figure 4: SGD vs Theory on MNIST: MNIST ($60000 \times 28 \times 28$) images. Random features model on MNIST used with $n = 4000$ images, random features $d = 2000$, and $n_0 = 28 \times 28$ was trained with one run of SGD (middle) for various learning rates and regularization parameter 0.01; entries of the random features $\boldsymbol{W}_{ij} \sim N(0,1)$ and a normalized ReLu activation function $\sigma(\cdot) = (\max\{0,\cdot\} - a)/b$ was applied. The Volterra equation matches the dynamics of the training loss (least-squares), $\mathcal{L}$, even with only one run of SGD. The log(eigenvalues) of the covariance of the MNIST dataset and the random features matrix used in the regression displayed (left). The expected risk, $\mathcal{R}(\boldsymbol{x}) = \frac{1}{2}\mathbb{E}[(b - \boldsymbol{x}^T\sigma(\boldsymbol{x}_i\boldsymbol{W}))^2]$ where $\boldsymbol{x}_i$ is an image from the MNIST test set, follows the predicted behavior Ω_t. Both the predicted Ψ_t and Ω_t match the performance of SGD in this non-idealized setting.

Theorem 3.1 (Concentration of SGD)**.** *Suppose n and d are related by Assumption 1. Suppose the ℓ^2-regularized least-squares problem (2) satisfies Assumptions 2 and 3 where $n \geq d^{\tilde{\varepsilon}}$ for some $\tilde{\varepsilon} > 0$. Suppose the learning rate schedule γ satisfies Assumption 4, and the initialization $\boldsymbol{x}_0$ satisfies Assumption 5. Let $\mathcal{R} : \mathbb{R}^d \mapsto \mathbb{R}$ be any quadratic statistic satisfying Assumption 6. Further assume that $\mathcal{R}$ and $\mathscr{L}$ have bounded $\|\mathcal{R}\|_{H^2}$ and $\|\mathscr{L}\|_{H^2}$ independent of n or d, for some C' sufficiently large. For any deterministic $T > 0$ and any $D > 0$, there is a $C > 0$ such that*

$$\Pr\left[\sup_{0\leqslant t\leqslant T}\left\|\begin{pmatrix}\mathscr{L}(\boldsymbol{x}_{\lfloor tn\rfloor})\\ \mathcal{R}(\boldsymbol{x}_{\lfloor tn\rfloor})\end{pmatrix} - \begin{pmatrix}\Psi_t\\ \Omega_t\end{pmatrix}\right\| > d^{-\tilde{\varepsilon}/2} \,\middle|\, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}_0\right] \leq C' d^{-D},$$

where Ω_t solves (14) and $\boldsymbol{x}_t$ are the iterates of SGD.

A formal proof of Theorem 3.1 can be found in [47, Theorem 1.4]. The functions Ψ_t and Ω_t can be viewed as the expected training loss and generalization error, respectively. Theorem 3.1 then shows concentration around the mean. We remark that to solve (8) we need as input $\mathscr{L}(\mathscr{X}^{\mathrm{gf}}_{\Gamma(t)})$ which can be computed using (10).

The solution of Ψ_t can be found by repeatedly convolving the forcing term $\mathscr{L}(\mathscr{X}^{\mathrm{gf}}_{\Gamma(t)})$ with the kernel K (provided $\sup_{t\geq 0}\sup_{s\geq 0} K(t,s;\nabla^2\mathscr{L})$ is bounded [27]). Moreover, numerical approximations to (8) can be found by taking a large but finite number of convolutions in the expression above. The boundedness of this solution corresponds precisely to learning rate choices for which SGD is convergent.

In the case of constant learning rate $\gamma(s) \equiv \gamma$, more can be said. The Volterra equation (8) is of convolution–type, and in fact is a special case of the renewal equation [5] (allowing for *defective* and *excessive* variants). Specifically, the expression in (9) simplifies to

$$\Psi_t = \mathscr{L}(\mathscr{X}^{\mathrm{gf}}_{\Gamma(t)}) + \frac{\gamma^2}{n}\int_0^t \mathbf{tr}\left((\boldsymbol{A}^T\boldsymbol{A})^2 e^{-2(\boldsymbol{A}^T\boldsymbol{A}+\delta\mathbf{I}_d)(t-s)}\right)\Psi_s \, \mathrm{d}s. \tag{15}$$

In addition to fixed point algorithms, one can also use Laplace transform techniques. These solutions to (15) can be analyzed explicitly for convergence guarantees and rates of convergence, see [46, 45]. As a simple example writing $K(t,s;\boldsymbol{P}) = K(t-s;\boldsymbol{P})$, the convergence of (15) occurs precisely when $\int_0^\infty K(t;\nabla^2\mathscr{L})\,\mathrm{d}t \leqslant 1$.[2]

Under the assumption that $\gamma(s)$ stabilizes, i.e. $\gamma(s) \to \gamma$ as $s \to \infty$, we may still characterize the eventual behavior of the solution. In the case that $\mathcal{R}$ represents the population risk, the difference $\Omega_t - \mathcal{R}(\mathscr{X}^{\mathrm{gf}}_{\Gamma(t)})$ gains the interpretation of the excess risk of SGD over gradient flow. On taking $t \to \infty$, this converges to the excess risk of the SGD estimator over the ridge regression estimator:

Theorem 3.2. *If $\gamma(t) \to 0$ but $\Gamma(t) \to \infty$ as $t \to \infty$ (c.f. the Robbins-Monro setting), then $\Omega_t - \mathcal{R}(\mathscr{X}^{gf}_{\Gamma(t)}) \xrightarrow[t\to\infty]{} 0$. If, on the other hand, $\gamma(t) \to \widetilde{\gamma} > 0$, where the limiting learning rate satisfies*

$$\widetilde{\gamma} < 2\left(\tfrac{1}{n}\mathbf{tr}((\boldsymbol{A}^T\boldsymbol{A})^2(\boldsymbol{A}^T\boldsymbol{A}+\delta\boldsymbol{I}_d)^{-1})\right)^{-1}, \tag{16}$$

[2] See [5, Chapter V] for a general discussion. In the case that the norm is exactly 1, this remains true as it is a special case of the Blackwell renewal theorem. When the norm is larger than 1, in the event that the empirical risk of gradient flow is bounded away from 0, the training loss is divergent.

then with Ψ_∞ given by the limiting empirical risk:

$$\Psi_\infty = \mathscr{L}\big(\mathscr{X}^{gf}_\infty\big)\times\left(1-\frac{\widetilde{\gamma}}{2n}\,\mathbf{tr}\big((\nabla^2\mathscr{L})^2\big(\nabla^2\mathscr{L}+\delta\boldsymbol{I}_d\big)^{-1}\big)\right)^{-1}$$

the limiting excess risk of SGD over ridge regression is given by

$$\Omega_t - \mathcal{R}\big(\mathscr{X}^{gf}_{\Gamma(t)}\big) \xrightarrow[t\to\infty]{} \frac{\widetilde{\gamma}}{2n}\Psi_\infty \times \mathbf{tr}\left((\nabla^2\mathcal{R})(\nabla^2\mathscr{L})\big(\nabla^2\mathscr{L}+\delta\boldsymbol{I}_d\big)^{-1}\right).$$

Examples of statistics. We give some common statistics that illustrate the versatility of our set-up.

One important quadratic statistic which satisfies all assumptions in Section 2.3 is $\mathcal{R}(\cdot) = \frac{1}{2}\|\cdot-\boldsymbol{\beta}\|^2$ where $\boldsymbol{\beta}$ is the unknown, ground truth signal.

Another common statistic in the standard linear regression set-up is the *population risk*. We first address the in-distribution set-up, where the data is drawn from the same distribution as the population. Let $\boldsymbol{A}$ be generated by taking n independent d-dimensional samples from a centered distribution $\mathcal{D}_f$ with feature covariance $\boldsymbol{\Sigma}_f \in \mathbb{R}^{d\times d}$, that is $\boldsymbol{\Sigma}_f = \mathbb{E}[\boldsymbol{a}\boldsymbol{a}^T]$, where $\boldsymbol{a}\sim\mathcal{D}_f$. We suppose a new data point $(\boldsymbol{a}, b)$ is drawn from a distribution $\mathcal{D}$ on $\mathbb{R}^d\times\mathbb{R}$ with the property that $\mathbb{E}[b\,|\,\boldsymbol{a}] = \boldsymbol{\beta}^T\boldsymbol{a}$ where $(\boldsymbol{a}, b)\sim\mathcal{D}$ and the data $\boldsymbol{a}\sim\mathcal{D}_f$. As before, $\boldsymbol{\beta}$ is the ground truth signal. The vector $\boldsymbol{x}_t$ generated by SGD represents an estimate of $\boldsymbol{\beta}$, and the population risk is

$$\mathcal{R}(\boldsymbol{x}_t) \overset{\text{def}}{=} \frac{1}{2}\mathbb{E}\big[(b-\boldsymbol{x}_t^T\boldsymbol{a})^2\,|\,\boldsymbol{x}_t\big], \tag{17}$$

where $(\boldsymbol{a}, b)\sim\mathcal{D}$.

In the out-of-distribution case, the data matrix $\boldsymbol{A}$ is generated using one distribution but the sample $(\boldsymbol{a}, b)\sim\mathcal{D}$ is not drawn from the same distribution as $\boldsymbol{A}$, that is, $\boldsymbol{a}\sim\hat{\mathcal{D}}_f\neq\mathcal{D}_f$ but still $\mathbb{E}[b|\boldsymbol{a}] = \boldsymbol{\beta}^T\boldsymbol{a}$.

Finally, for random features with a linear ground truth, we would take

$$\mathcal{R}(\boldsymbol{x}_t) \overset{\text{def}}{=} \mathbb{E}\big[(b-\boldsymbol{x}_t^T\sigma(\boldsymbol{X}_i\boldsymbol{W}/\sqrt{n_0}))^2\,|\,\boldsymbol{x}_t,\boldsymbol{W}\big]. \tag{18}$$

All these examples are quadratic statistics for which Theorem 3.1 applies.

4 Average-case Complexity & Parameter Selections

The dynamics of training curves for generic objective functions, in general, are quite complicated. However, as we have seen, in the case of ℓ^2-regularized least squares problem under high-dimensionality, the dynamics for stochastic learning algorithms are simple. As such, one can go further and get additional information about the performance of these algorithms. In this section, we use the predictions of the exact training dynamics to draw important insights on typical computational complexity and parameter selection (*e.g.*, learning rate, batch size, and momentum parameters). We will focus our attention on the widely used stochastic gradient descent algorithm with momentum (SGD+M) on the ℓ^2-regularized least squares problem with regularization parameter $\delta = 0$ (*e.g.*, $\min_{\boldsymbol{x}} 1/2\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}\|^2$). Mini-batch SGD+M is defined by selecting uniformly at random a subset $B_k\subseteq\{1,2,\cdots,n\}$ of cardinality β and making the update

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \gamma\sum_{i\in B_k}\nabla f_i(\boldsymbol{x}_k) + \Delta(\boldsymbol{x}_k-\boldsymbol{x}_{k-1})\\ &= \boldsymbol{x}_k - \gamma\boldsymbol{A}^T\boldsymbol{P}_k(\boldsymbol{A}\boldsymbol{x}_k-\boldsymbol{b}) + \Delta(\boldsymbol{x}_k-\boldsymbol{x}_{k-1}),\\ \text{where}\quad \boldsymbol{P}_k &\overset{\text{def}}{=} \sum_{i\in B_k}\boldsymbol{e}_i\boldsymbol{e}_i^T,\end{aligned} \tag{19}$$

with $\boldsymbol{P}_k$ a random orthogonal projection matrix and $\boldsymbol{e}_i$ the i-th standard basis vector. Here $\gamma>0$ is the learning rate parameter, Δ is the momentum parameter, and the function f_i is the i-th element of the sum in (2). Note we are only considering the constant learning rate and momentum setting. We define the *batch fraction* ζ as the ratio of β/n.

When the stochastic gradient in (19) is replaced with the full-gradient $\nabla f(\boldsymbol{x})$ and the hyperparameters are chosen optimally, the resulting algorithm is the celebrated heavy-ball momentum (a.k.a. Polyak momentum) [48]. The optimal learning rate and momentum parameters are explicitly given by

$$\gamma = \frac{4}{(\sqrt{\sigma^2_{\max}}+\sqrt{\sigma^2_{\min}})^2} \quad\text{and}\quad \Delta = \left(\frac{\sqrt{\sigma^2_{\max}}-\sqrt{\sigma^2_{\min}}}{\sqrt{\sigma^2_{\max}}+\sqrt{\sigma^2_{\min}}}\right)^2. \tag{20}$$

It is well-known that heavy-ball is an optimal algorithm on the least squares problem in that it converges linearly at a rate of $\mathcal{O}(1/\sqrt{\kappa})$.

In the influential work of [52], the authors empirically show that SGD+M significantly improves training performance of deep neural networks. Despite its wide usage in machine learning practice, our understanding of it is more narrow. It has been hypothesized that SGD+M improves training because it is employed on a large batch of a data set [33], thereby emulating the speed-up one sees in full-batch settings. For many learning problems, the "large batch" setting is often paired with high-dimensional problems, meaning there are many samples (and

likely also many features to have interesting behavior). There have been some recent works in proving that for sufficiently large batch sizes SGD+M does achieve $\mathcal{O}(1/\sqrt{\kappa})$ [13, 16]; see also [29, 37, 3]. We can go further and find the correct batch size dependency in the high-dimensional regime.

First, we address how the batch effects the training dynamics.

Theorem 4.1 (Concentration of mini-batch momentum)**.** *Suppose the assumptions of Theorem 3.1 hold. For any deterministic $T > 0$ and any $D > 0$, there is a $C > 0$ such that*

$$\Pr\left[\sup_{0\leqslant k\leqslant T}\left|\mathscr{L}(\boldsymbol{x}_k)-\Psi_k\right|>d^{-\bar{\varepsilon}/2}\,\middle|\,\boldsymbol{A},\boldsymbol{b},\boldsymbol{x}_0\right]\leq C'd^{-D},$$

where Ψ_k solves a discrete convolution-type Volterra equation

$$\Psi_{k+1}=\mathscr{L}(\boldsymbol{x}_{k+1}^{\mathrm{GD+M}(\gamma\zeta)})+\sum_{t=0}^{k}K(k-t;\nabla^2\mathscr{L})\Psi_t. \quad (21)$$

Here $K(k)$ is a kernel completely determined by the spectrum of $\nabla^2\mathscr{L}$ and $\{\boldsymbol{x}_k^{\mathrm{GD+M}(\gamma\zeta)}\}$ are the iterates generated by running full-batched momentum (i.e., $\zeta = 1$ in (19)) with learning rate given by $\zeta\gamma$ and momentum parameter Δ.

The expression for (21) can be viewed as a discrete convolution-type Volterra equation with forcing term $\mathscr{L}(\boldsymbol{x}_k^{\mathrm{GD+M}(\gamma\zeta)})$ and kernel $K(t;\nabla^2\mathscr{L})$. The forcing term, $F(k) = \mathscr{L}(\boldsymbol{x}_k^{\mathrm{GD+M}(\gamma\zeta)})$ represents the mean (with respect to expectation over the mini-batches) behavior of SGD+M. For small learning rates γ, the forcing term controls the dynamics of Ψ_k. We denote the dominant term in $F(k)$ by $\lambda_{\max}(\gamma,\Delta,\zeta)$, that is $F(k) = \mathcal{O}(\lambda_{\max}^k)$. Specifically,

$$\lambda_j \overset{\text{def}}{=} \frac{-2\Delta+\Omega_j^2+\sqrt{\Omega_j^2(\Omega_j^2-4\Delta)}}{2},$$
$$\text{where}\quad \Omega_j \overset{\text{def}}{=} 1-\gamma\zeta\sigma_j^2+\Delta,$$
$$\lambda_{\max}\overset{\text{def}}{=}\max_{1\leq j\leq n}|\lambda_j|,\ \text{and}\ \sigma_j^2,\ \text{eigenvalue of}\ \boldsymbol{A}\boldsymbol{A}^T. \quad (22)$$

On the other hand, the kernel term, or convolution in (21), $\sum_{t=0}^k K(k-t,\nabla^2\mathscr{L})\Psi_t$, is due to the inherent stochasticity generated by uniformly at random selecting indices. The presence of Ψ_t (training loss) in this term is due to the fact that the noise generated by the k-th stochastic gradient is proportional to Ψ_t, and the function $K(k-t)$ represents the progress of the algorithm in sending this extra noise to 0. We note that the kernel $K(k-t)$ in (21) scales quadratically in the learning rate γ. Hence for *large* learning rates, the kernel dominates the decay behavior of Ψ_k.

There are also some key relationships between Theorem 3.1 (batch sizes $\zeta \to 0$) and Theorem 4.1, notably that (when $\Delta = 0$), gradient flow in the forcing term of (9) becomes gradient descent (21) – discrete gradient flow. A similar discretization is observed in the kernel with an integral replaced by a summation.

4.1 Convolution Volterra analysis

We begin by establishing sufficient conditions for the convergence of the solution to the Volterra equation (21), a special case of the *renewal equation* ([6]). Let us translate (21) into the form of the renewal equation as follows:

$$\psi(t+1)=F(t+1)+(\tilde{K}*\psi)(t), \quad (23)$$

where $(f*g)(t)=\sum_{k=0}^{\infty}f(t-k)g(k)$. Let the *kernel norm* be $\|\tilde{K}\|=\sum_{t=0}^{\infty}\tilde{K}(t)$. By [6, Proposition 7.4], we see that $\|\tilde{K}\|<1$ is necessary for our solution to the Volterra equation to be convergent. Indeed, we have the following result.

Proposition 4.1. *If the norm $\|\tilde{K}\| < 1$, the algorithm is convergent in that*

$$\Psi_\infty\overset{def}{=}\lim_{k\to\infty}\Psi_k=\frac{\lim_{k\to\infty}\mathscr{L}(\boldsymbol{x}_k^{\mathrm{GD+M}(\gamma\zeta)})}{1-\|\tilde{K}\|}. \quad (24)$$

Proposition 4.1 formulates the limit behaviour of the objective function in both the over-determined and the under-determined cases of least squares. When under-determined, the limiting loss value of $\mathscr{L}(\boldsymbol{x}_k^{\mathrm{GD+M}(\gamma\zeta)}) = 0$ and the limiting Ψ_∞ is 0; otherwise the limiting loss value is strictly positive. The result (24) only makes sense when the noise term K satisfies $\|K\| < 1$; the next proposition illustrates the conditions on the learning rate and the trace of the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$ such that the kernel norm is less than 1.

Proposition 4.2 (Convergence threshold)**.** *Under the learning rate condition $\gamma < \frac{1+\Delta}{\zeta\sigma_{max}^2}$ and trace condition $\frac{(1-\zeta)\gamma}{1-\Delta}\cdot\frac{1}{n}\mathbf{tr}(\boldsymbol{A}\boldsymbol{A}^T) < 1$, the kernel norm $\|\tilde{K}\| < 1$, i.e., $\sum_{t=0}^{\infty}\tilde{K}(t) < 1$.*

The *learning rate condition* quantifies an upper bound of good learning rates by the largest eigenvalue of the covariance matrix $\sigma_{\max}^2$, batch fraction ζ, and the momentum parameter Δ. The *trace condition* illustrates a constraint on the growth of $\sigma_{\max}^2$. Moreover, for a full batch gradient descent model ($\zeta = 1$), the trace condition can be dropped and we get the classical learning rate condition for gradient descent.

4.2 The Malthusian exponent and complexity

The rate of convergence of Ψ_k is essentially the worse of two terms – the forcing term $F(t)$ and a discrete time convolution $\sum_{t=0}^{k} K(k-t;\nabla^2\mathscr{L})\Psi_t$ which depends on the kernel K. Intuitively, the forcing term captures the behavior of the expected value of SGD+M and the discrete time convolution captures the slowdown in training due to noise created by the algorithm. Note that $F(k)$ is always a lower bound for Ψ_k, but it can be that Ψ_k is exponentially (in k) larger than $F(k)$ owing to the convolution term. This occurs when something called the *Malthusian exponent*, denoted Ξ, of the convolution Volterra equation exists. The Malthusian exponent Ξ is given as the unique solution to

$$\gamma^2\zeta(1-\zeta)\sum_{t=0}^{\infty}\Xi^t K(t;\nabla^2\mathscr{L}) = 1, \text{ if solution exists.} \tag{25}$$

The Malthusian exponent enters into the complexity analysis in the following way:

Theorem 4.2 (Asymptotic rates)**.** *The inverse of the Malthusian exponent always satisfies $\Xi^{-1} > \Lambda$ for finite n. Moreover, for some $C > 0$, the convergence rate for SGD+M is*

$$\begin{aligned} &\Psi_k - \Psi_\infty \le C\max\{\lambda_{\max}, \Xi^{-1}\}^k \\ \textit{and}\quad &\lim_{t\to\infty}(\Psi_k - \Psi_\infty)^{1/k} = \max\{\lambda_{\max}, \Xi^{-1}\}. \end{aligned} \tag{26}$$

Thus to understand the rates of convergence, it is necessary to understand the Malthusian exponent as a function of γ and Δ.

4.3 Two regimes for the Malthusian exponent

On the one hand, the Malthusian exponent Ξ comes from the stochasticity of the algorithm itself. On the other hand, $\lambda_{\max}(\gamma,\Delta,\zeta)$ is determined completely by the problem instance information — the eigenspectrum of $\boldsymbol{A}\boldsymbol{A}^T$. (Note we want to emphasize the dependence of $\lambda_{\max}$ on the learning rate, the momentum parameter, and the batch fraction). Let $\sigma^2_{\max}$ and $\sigma^2_{\min}$ denote the maximum and minimum *nonzero* eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$, respectively. For a fixed batch fraction, the optimal parameters $(\gamma_\lambda, \Delta_\lambda)$ of $\lambda_{\max}$ are

$$\begin{aligned} \gamma_\lambda &= \frac{1}{\zeta}\left(\frac{2}{\sqrt{\sigma^2_{\max}}+\sqrt{\sigma^2_{\min}}}\right)^2 \\ \text{and}\quad \Delta_\lambda &= \left(\frac{\sqrt{\sigma^2_{\max}}-\sqrt{\sigma^2_{\min}}}{\sqrt{\sigma^2_{\max}}+\sqrt{\sigma^2_{\min}}}\right)^2. \end{aligned} \tag{27}$$

In the full batch setting, i.e. $\zeta = 1$, these optimal parameters γ_λ and Δ_λ for $\lambda_{\max}$ are exactly the Polyak momentum parameters (20). Moreover, in this setting, there is no stochasticity so the Malthusian exponent disappears and the convergence rate (26) is $\lambda_{\max}$. We observe from (27) that for all fixed batch fractions, the optimal momentum parameter, Δ_λ, is independent of batch size. The only dependence on batch size appears in the learning rate. At first it appears that for small batch fractions, one can take large learning rates, but in that case, the inverse of the Malthusian exponent Ξ^{-1} dominates the convergence rate of SGD+M (26) and you cannot take γ and Δ to be as in (27) (See Figure 5).

We will define two subsets of parameter space: the *problem constrained regime* and the *algorithmically constrained regime* (or stochastically constrained regime). The problem constrained regime is for some tolerance $\varepsilon > 0$

$$\{(\gamma,\Delta) : 1-\sqrt{\Xi} < (1-\sqrt{\lambda^{-1}_{\max}})(1-\varepsilon)\}. \tag{28}$$

The remainder we call the *algorithmically constrained* regime. To explain the tolerance: for finite n, it transpires that we always have $\Xi^{-1} > \lambda_{\max}$, but it could be vanishingly close to $\lambda_{\max}$ as a function of n. Hence we introduce the tolerance to give the correct qualitative behavior in finite n.

Proposition 4.3. *If the learning rate $\gamma \le \min(\frac{1+\Delta}{\zeta\sigma^2_{max}}, \frac{(1-\sqrt{\Delta})^2}{\zeta\sigma^2_{\min}})$, with the trace condition $\frac{8(1-\zeta)\gamma}{1-\Delta}\cdot\frac{1}{n}\mathbf{tr}(\boldsymbol{A}^T\boldsymbol{A}) < 1$, then (γ,Δ) is in the problem constrained regime with $\varepsilon = 1/2$.*

Therefore by (26), we have that

$$\Psi_t - \Psi_\infty \le D\left(\frac{4\lambda_{\max}}{(1+\sqrt{\lambda_{\max}})^2}\right)^t, \quad \text{for some } D > 0; \tag{29}$$

we note that the expression in the parenthesis is $1 - \frac{1}{2}(1-\lambda_{\max}) + \mathcal{O}((1-\lambda_{\max})^2)$.

In the problem constrained regime, it is worthwhile to note that the overall convergence rate is the same as full batch momentum with adjusted learning rate, *i.e.*, the batch size does not play an important role as long as we are in the problem constrained regime.

4.4 Performance of SGD+M: implicit conditioning ratio (ICR)

An advantage of the exact loss trajectory is that we give a rigorous definition of the large batch and small batch regimes which reflect a transition in the convergence behavior of SGD+M. To do this we introduce the *condition number* κ, the *average condition*

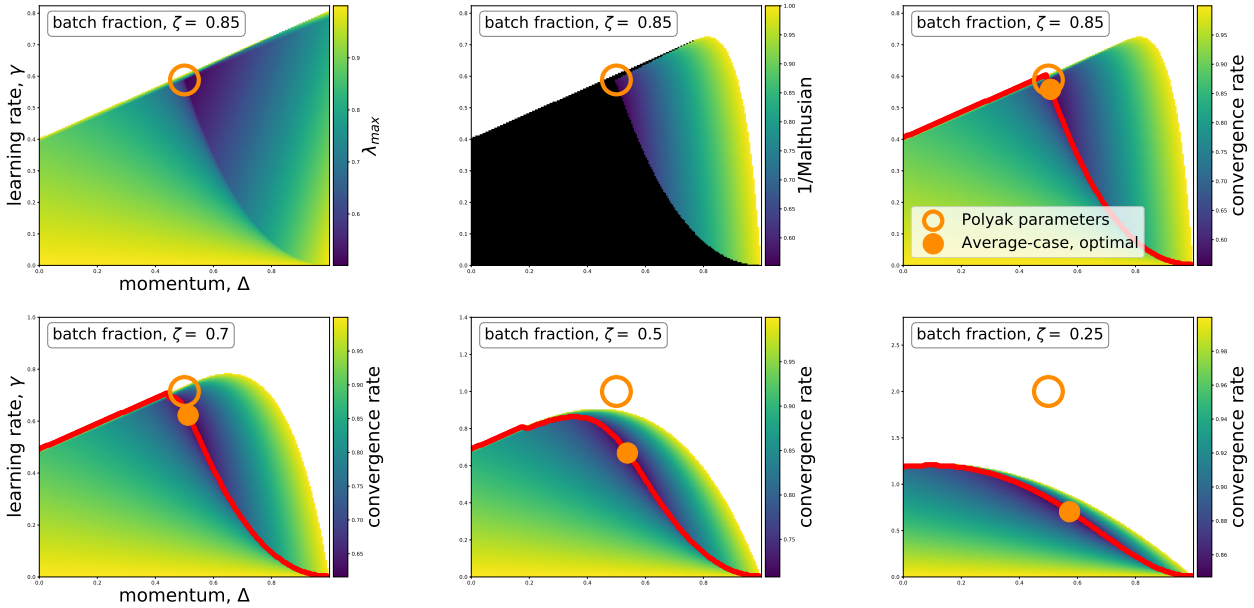


Figure 5: Different convergence rate regions: problem constrained regime versus algorithmically constrained regime for Gaussian random least squares problem with ($n = 2000 \times d = 1000$). Plots are functions of momentum (x-axis) and learning rate (y-axis). Analytic expression for $\lambda_{\max}$ (see (22)) – convergence rate of forcing term $F(t)$ – given in (top row, column 1) represents the problem constrained region. (top row, column 2) plots 1/(Malthusian exponent) ((25)); black region is where the Malthusian exponent Ξ does not exist. This represents the algorithmically constrained region. Finally, (top row, column 3 and bottom row) plots convergence rate of SGD+M $= \max\{\lambda_{\max}, \Xi^{-1}\}$, (see (26)), for various batch fractions. When the Malthusian exponent does not exist (black), $\lambda_{\max}$ takes over the convergence rate of SGD+M; otherwise the noise in the algorithm (i.e. Malthusian exponent Ξ) dominates. Optimal parameters that maximize $\lambda_{\max}$ denoted by Polyak parameters (orange circle, (27)) and the optimal parameters for SGD+M (orange dot); below red line is the problem constrained region; otherwise the algorithmic constrained region. When batch fractions $\zeta = 0.85$ and $\zeta = 0.7$ (top row and bottom row, column 1) (i.e., large batch), the SGD+M convergence rate is the deterministic momentum rate of $1/\sqrt{\kappa}$. As the batch fraction decreases ($\zeta = 0.25$), the convergence rate becomes that of SGD and the optimal parameters of SGD+M and Polyak parameters are quite far from each other. The Malthusian exponent (algorithmically constrained region) starts to control the SGD+M rate as batch fraction $\to 0$.

number $\bar{\kappa}$, and the *implicit conditioning ratio* (ICR) defined as

$$\bar{\kappa} \stackrel{\text{def}}{=} \frac{\frac{1}{n}\sum_{j\in[n]}\sigma_j^2}{\sigma_{\min}^2} < \frac{\sigma_{\max}^2}{\sigma_{\min}^2} \stackrel{\text{def}}{=} \kappa \quad \text{and} \quad \text{ICR} \stackrel{\text{def}}{=} \frac{\bar{\kappa}}{\sqrt{\kappa}}. \tag{30}$$

Here σ_j^2 are the eigenvalues of the Hessian of the least squares problem with $\sigma_{\max}^2$ and $\sigma_{\min}^2$ the largest and smallest (non-zero) eigenvalues. We refer to the *large batch* regime where $\zeta \geqslant$ ICR and the *small batch* regime where $\zeta \leqslant$ ICR.

We begin by giving a rate guarantee that holds in the problem constrained regime, for a specific choice of γ and Δ.

Proposition 4.4 (Good momentum parameters)**.** *Suppose the learning rate and momentum satisfy*

$$\gamma = \frac{(1-\sqrt{\Delta})^2}{\zeta\sigma_{\min}^2} \quad \textit{and}$$
$$\Delta = \max\left\{\left(\frac{1-\frac{\mathcal{C}}{\bar{\kappa}}}{1+\frac{\mathcal{C}}{\bar{\kappa}}}\right), \left(\frac{1-\frac{1}{\sqrt{2\kappa}}}{1+\frac{1}{\sqrt{2\kappa}}}\right)\right\}^2, \tag{31}$$
$$\textit{where } \mathcal{C} \stackrel{def}{=} \zeta/(8(1-\zeta)).$$

Then $\lambda_{\max} = \Delta$ *and for some* $C > 0$*, the convergence rate for SGD+M is*

$$\Psi_t - \Psi_\infty \leq C \cdot \Delta^t = C \cdot \max\left\{\left(\frac{1-\frac{\mathcal{C}}{\bar{\kappa}}}{1+\frac{\mathcal{C}}{\bar{\kappa}}}\right), \left(\frac{1-\frac{1}{\sqrt{2\kappa}}}{1+\frac{1}{\sqrt{2\kappa}}}\right)\right\}^{2t}. \tag{32}$$

Remark 4.1. *We note that for all* Δ *satisfying* $\frac{(1-\sqrt{\Delta})^2}{\zeta\sigma_{\min}^2} \leq \frac{(1+\sqrt{\Delta})^2}{2\zeta\sigma_{\max}^2}$ *with the learning rate* γ *as in*

(31), *we have that* $\lambda_{\max} = \Delta$. *By minimizing the* Δ *(i.e., by finding the fastest convergence rate), we get the formula for the momentum parameter in* (31).

The exact tradeoff in convergence rates (32) occurs when

$$\frac{\mathcal{C}}{\bar{\kappa}} = \frac{1}{\sqrt{2\kappa}}, \quad \text{or} \quad \zeta = \frac{\frac{8}{\sqrt{2}}\mathrm{ICR}}{1 + \frac{8}{\sqrt{2}}\mathrm{ICR}}. \tag{33}$$

As $\zeta \leqslant 1$, this condition is only nontrivial when $\mathrm{ICR} \ll 1$, in which case $\zeta = \frac{8}{\sqrt{2}}\mathrm{ICR}$, up to vanishing errors.

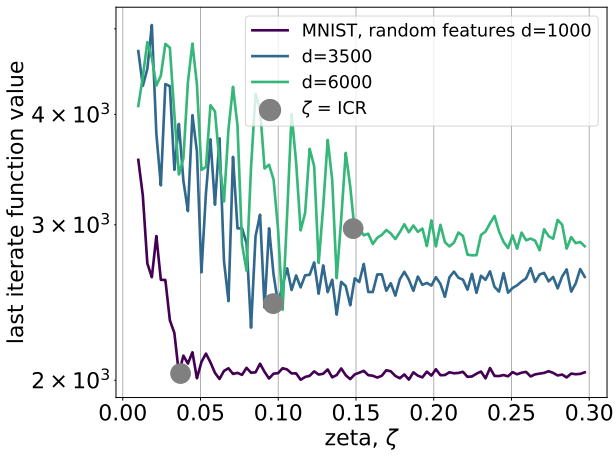


Figure 6: ICR and batch saturation on MNIST data. SGD with momentum using a batch fraction ζ on MNIST data [34]; training loss is given after 20 iterations. Increasing the batch size yields proportional complexity improvements up to a saturation point (gray dot, explicit formula in [35]) which occurs before the full gradient is deployed. This yields the first provable optimal linear rate for stochastic momentum learning algorithm that matches its deterministic equivalent.

Large batch ($\zeta \geqslant$ ICR). In this regime SGD+M's performance matches the performance of the heavy-ball algorithm with the Polyak momentum parameters (up to absolute constants). More specifically, with the choices of γ and Δ in Proposition 4.4, the linear rate of convergence of SGD+M is $1 - \frac{c}{\sqrt{\kappa}}$ for an absolute c. Note that ζ does not appear in the rate, and in particular there is no gain in convergence rate by increasing the batch fraction.

Small batch ($\zeta \leqslant$ ICR). In the small batch regime, the value of $\mathcal{C}$ is relatively small and the first term is dominant in (32), and so the linear rate of convergence of SGD+M is $1 - \frac{c\zeta}{\bar{\kappa}}$ for some absolute constant $c > 0$. In this regime, there is a benefit in increasing the batch fraction, and the rate increases linearly with the fraction. We note that on expanding the choice of constants in small ζ the choices made in Proposition 4.4 are

$$\Delta \approx 1 - \frac{\zeta}{8\bar{\kappa}} \quad \text{and} \quad \gamma \approx \frac{\zeta}{256\bar{\kappa}^2\sigma_{\min}^2}.$$

This rate can also be achieved by taking $\Delta = 0$, i.e. mini-batch SGD with no momentum. Moreover, it is not possible to beat this by using momentum; we show the following lower bound:

Proposition 4.5. *If* $\zeta \leqslant \min\{\frac{1}{2}, ICR\}$ *then there is an absolute constant* $C > 0$ *so that for convergent* (γ, Δ) *(those satisfying Proposition 4.2),* $\sqrt{\lambda_{\max}} \geqslant 1 - \frac{C\zeta}{\bar{\kappa}}$.

This is a lower bound on the rate of convergence by Theorem 4.2.

Conclusions. While the engineering side of machine learning has leapt ahead, the theoretical explanation for what is happening in ML training has largely been left behind. The needed theory of optimization to close this gap should fit 3 key aspects: (1) the algorithm is a gradient-based method, (2) the training loss is a high–dimensional "finite-sum", and (3) the model is "the right type" of nonconvex problem.

In this work, we presented a theory that does 2 of the 3; we outlined a framework for addressing this gap between theory and practice by incorporating a deterministic resolvent condition into the assumptions. For the ℓ^2-regularized least squares problem, the stochastic learning algorithms concentrate around a simple, predictable path. By analyzing this path, one can draw insights into average-case complexity and parameter selection properties, all of which have enormous practical implications for making machine learning work.

Clearly the most urgent direction of future research is away from the least squares setting, to handle more general losses and some types of nonconvexity. On the one hand, there is evidence that the right type of nonconvex problems are not so far from convex, taking to heart that, for example, wide neural networks degenerate to kernel regression problems, which are covered by this framework. On the other hand, as we move away from the least squares setting, we truly do not know what we do not know; there are many other important model problems which need the high-dimensional optimization treatment, such as generalized linear models, inverse problems like phase retrieval, and neural networks.

References

[1] B. Adlam and J. Pennington. "The Neural Tangent Kernel in High Dimensions: Triple Descent and a Multi-Scale Theory of Generalization". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. Vol. 119. 2020, pp. 74–84. URL: https://proceedings.mlr.press/v119/adlam20a.html.

[2] B. Adlam and J. Pennington. "Understanding Double Descent Requires A Fine-Grained Bias-Variance Decomposition". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 11022–11032. URL: https://proceedings.neurips.cc/paper/2020/file/7d420e2b2939762031eed0447a9be19f-Paper.pdf.

[3] Z. Allen-Zhu. "Katyusha: The first direct acceleration of stochastic gradient methods". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 8194–8244. URL: https://jmlr.org/papers/volume18/16-410/16-410.pdf.

[4] Y. Arjevani, Y. Carmon, and J.C. Duchi. "Lower bounds for non-convex stochastic optimization". In: *Math. Program.* 155 (2022). DOI: 10.1007/s10107-022-01822-7.

[5] S. Asmussen. *Applied probability and queues*. Second. Vol. 51. Applications of Mathematics (New York). Stochastic Modelling and Applied Probability. Springer-Verlag, New York, 2003, pp. xii+438. URL: https://link.springer.com/book/10.1007/b97236.

[6] S. Asmussen. *Applied probability and queues*. Second. Vol. 51. Applications of Mathematics (New York). Stochastic Modelling and Applied Probability. Springer-Verlag, New York, 2003, pp. xii+438. URL: https://link.springer.com/book/10.1007/b97236.

[7] F. Bach and E. Moulines. "Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n)". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 26. 2013. URL: https://proceedings.neurips.cc/paper/2013/file/7fe1f8abaad094e0b5cb1b01d712f708-Paper.pdf.

[8] M. Belkin, D. Hsu, S. Ma, and S. Mandal. "Reconciling modern machine-learning practice and the classical bias-variance trade-off". In: *Proc. Natl. Acad. Sci. USA* 116.32 (2019), pp. 15849–15854. DOI: 10.1073/pnas.1903070116.

[9] G. Ben Arous, R. Gheissari, and A. Jagannath. "High-dimensional limit theorems for SGD: Effective dynamics and critical scaling". In: *arXiv preprint arXiv:2206.04030* (2022).

[10] D. Bertsekas. "A new class of incremental gradient methods for least squares problems". In: *SIAM J. Optim.* 7.4 (1997), pp. 913–926. DOI: 10.1137/S1052623495287022.

[11] D. Bertsekas and J. Tsitsiklis. "Gradient convergence in gradient methods with errors". In: *SIAM J. Optim.* 10.3 (2000), pp. 627–642. DOI: 10.1137/S1052623497331063.

[12] J. Blanchard, C. Cartis, and J. Tanner. "Compressed sensing: how sharp is the restricted isometry property?" In: *SIAM Rev.* 53.1 (2011), pp. 105–125. DOI: 10.1137/090748160.

[13] R. Bollapragada, T. Chen, and R. Ward. "On the fast convergence of minibatch heavy ball momentum". In: *arXiv preprint arXiv:2206.07553* (2022).

[14] B. Bordelon and C. Pehlevan. "Learning Curves for SGD on Structured Features". In: *International Conference on Learning Representations (ICLR)*. 2022. URL: https://openreview.net/forum?id=WPI2vbkAl3Q.

[15] L. Bottou, F.E. Curtis, and J. Nocedal. "Optimization methods for large-scale machine learning". In: *SIAM Review* 60.2 (2018), pp. 223–311. URL: https://epubs.siam.org/doi/10.1137/16M1080173.

[16] B. Can and M. Gurbuzbalaban. "Entropic Risk-Averse Generalized Momentum Methods". In: *arXiv preprint arXiv:2204.11292* (2022).

[17] M. Celentano, C. Cheng, and A. Montanari. "The high-dimensional asymptotics of first order methods with random data". In: *arXiv preprint arXiv:2112.07572* (2021). URL: https://arxiv.org/abs/2112.07572.

[18] K. Chandrasekher, A. Pananjady, and C. Thrampoulidis. "Sharp global convergence guarantees for iterative nonconvex optimization: A Gaussian process perspective". In: *arXiv preprint arXiv:2109.09859* (2021).

[19] D. Davis and D. Drusvyatskiy. "Stochastic model-based minimization of weakly convex functions". In: *SIAM J. Optim.* 29.1 (2019), pp. 207–239. ISSN: 1052-6234. DOI: 10.1137/18M1178244.

[20] A. Defazio, F. Bach, and S. Lacoste-Julien. "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014. URL: https://arxiv.org/pdf/1407.0202.pdf.

[21] A. Defossez and F. Bach. "Averaged Least-Mean-Squares: Bias-Variance Trade-offs and Optimal Sampling Distributions". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Vol. 38. Proceedings of Machine Learning Research. 2015, pp. 205–213. URL: http://proceedings.mlr.press/v38/defossez15.pdf.

[22] E. Dobriban and S. Wager. "High-dimensional asymptotics of prediction: ridge regression and classification". In: *Ann. Statist.* 46.1 (2018), pp. 247–279. DOI: 10.1214/17-AOS1549.

[23] Y. Drori and O. Shamir. "The Complexity of Finding Stationary Points with Stochastic Gradient Descent". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2658–2667. URL: https://proceedings.mlr.press/v119/drori20a.html.

[24] C. Fang, C. J. Li, Z. Lin, and T. Zhang. "SPIDER: Near-Optimal Non-Convex Optimization via Stochastic Path-Integrated Differential Estimator". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. 2018. URL: https://proceedings.neurips.cc/paper/2018/file/1543843a4723ed2ab08e18053ae6dc5b-Paper.pdf.

[25] S. Ghadimi and G. Lan. "Stochastic first- and zeroth-order methods for nonconvex stochastic programming". In: *SIAM J. Optim.* 23.4 (2013), pp. 2341–2368. URL: https://arxiv.org/pdf/1309.5549.pdf.

[26] R. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik. "SGD: General analysis and improved rates". In: *International Conference on Machine Learning (ICML)*. PMLR. 2019. URL: https://arxiv.org/pdf/1901.09401.pdf.

[27] G. Gripenberg. "On the resolvents of nonconvolution Volterra kernels". In: *Funkcial. Ekvac.* 23.1 (1980), pp. 83–95. ISSN: 0532-8721. URL: http://www.math.kobe-u.ac.jp/~fe/xml/mr0586277.xml.

[28] T. Hastie, A. Montanari, S. Rosset, and R.J. Tibshirani. "Surprises in high-dimensional ridgeless least squares interpolation". In: *arXiv preprint arXiv:1903.08560* (2019). URL: https://arxiv.org/abs/1903.08560.

[29] P. Jain, S. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. "Accelerating Stochastic Gradient Descent for Least Squares Regression". In: *Proceedings of the 31st Conference On Learning Theory (COLT)*. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 545–604. URL: http://proceedings.mlr.press/v75/jain18a/jain18a.pdf.

[30] P. Jain, S. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. "Parallelizing Stochastic Gradient Descent for Least Squares Regression: Mini-batching, Averaging, and Model Misspecification". In: *Journal of Machine Learning Research* 18.223 (2018), pp. 1–42. URL: http://jmlr.org/papers/v18/16-595.html.

[31] R. Johnson and T. Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in Neural Information Processing Systems*. 2013. URL: http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduc.

[32] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020).

[33] R. Kidambi, P. Netrapalli, P. Jain, and S. Kakade. "On the Insufficiency of Existing Momentum Schemes for Stochastic Optimization". In: *2018 Information Theory and Applications Workshop (ITA)*. 2018, pp. 1–9. DOI: 10.1109/ITA.2018.8503173.

[34] Y. LeCun, C. Cortes, and C. Burges. *"MNIST" handwritten digit database*. 2010. URL: http://yann.lecun.com/exdb/mnist.

[35] K. Lee, A.N. Cheng, E. Paquette, and C. Paquette. "Trajectory of Mini-Batch Momentum: Batch Size Saturation and Convergence in High-dimensions". In: *arXiv preprint arXiv:2206.01029* (2022).

[36] Z. Liao, R. Couillet, and M. Mahoney. "A Random Matrix Analysis of Random Fourier Features: Beyond the Gaussian Kernel, a Precise Phase Transition, and the Corresponding Double Descent". In: *arXiv preprint arXiv:2006.05013* (2020). URL: https://arxiv.org/pdf/2006.05013.pdf.

[37] C. Liu and M. Belkin. "Accelerating SGD with momentum for over-parameterized learning". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020. URL: https://arxiv.org/pdf/1810.13395.pdf.

[38] S. Mei and A. Montanari. "The generalization error of random features regression: precise asymptotics and the double descent curve". In: *Comm. Pure Appl. Math.* 75.4 (2022), pp. 667–766.

[39] F. Mignacco, F. Krzakala, P. Urbani, and L. Zdeborova. "Dynamical mean-field theory for stochastic gradient descent in Gaussian mixture classification". In: *arXiv preprint arXiv:2006.06098* (2000).

[40] E. Moulines and F. Bach. "Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 24. 2011. URL: https://proceedings.neurips.cc/paper/2011/file/40008b9a5380fcacce3976bf7c08af5b-Paper.pdf.

[41] P. Nakkiran, B. Neyshabur, and H. Sedghi. "The Deep Bootstrap Framework: Good Online Learners are Good Offline Generalizers". In: *International Conference on Learning Representations (ICLR)*. 2021. URL: https://openreview.net/pdf?id=guetrIHLFGI.

[42] D. Needell, N. Srebro, and R. Ward. "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm". In: *Math. Program.* 155.1-2, Ser. A (2016), pp. 549–573. DOI: 10.1007/s10107-015-0864-7.

[43] A. S. Nemirovsky and D. B. and Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience Series in Discrete Mathematics. Translated from the Russian and with a preface by E. R. Dawson. John Wiley & Sons, Inc., New York, 1983, pp. xv+388.

[44] S. Oymak, B. Recht, and M. Soltanolkotabi. "Sharp time-data tradeoffs for linear inverse problems". In: *IEEE Trans. Inform. Theory* 64.6 (2018), pp. 4129–4158. DOI: 10.1109/TIT.2017.2773497.

[45] C. Paquette, K. Lee, F. Pedregosa, and E. Paquette. "SGD in the Large: Average-case Analysis, Asymptotics, and Stepsize Criticality". In: *arXiv preprint arXiv:2102.04396* (2021). URL: https://arxiv.org/pdf/2102.04396.pdf.

[46] C. Paquette and E. Paquette. "Dynamics of Stochastic Momentum Methods on Large-scale, Quadratic Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 34. 2021. URL: https://papers.nips.cc/paper/2021/hash/4cf0ed8641cfcbbf46784e620a0316fb-Abstract.html.

[47] E. Paquette, C. Paquette, B. Adlam, and J. Pennington. "Homogenization of SGD in high-dimensions: exact dynamics and generalization properties". In: *arXiv preprint arXiv:2205.07069* (2022). URL: https://arxiv.org/abs/2205.07069.

[48] B.T. Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 04 (1964). DOI: 10.1016/0041-5553(64)90137-5.

[49] H. Robbins and S. Monro. "A Stochastic Approximation Method". In: *Ann. Math. Statist.* (1951).

[50] Mark Schmidt, Nicolas Le Roux, and Francis Bach. "Minimizing finite sums with the stochastic average gradient". In: *Mathematical Programming* (2017). DOI: 10.1007/s10107-016-1030-6.

[51] P. Sur and E. Candès. "A modern maximum-likelihood theory for high-dimensional logistic regression". In: *Proc. Natl. Acad. Sci. USA* 116.29 (2019), pp. 14516–14525. DOI: 10.1073/pnas.1810420116.

[52] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Atlanta, Georgia, USA: PMLR, 2013, pp. 1139–1147. URL: https://proceedings.mlr.press/v28/sutskever13.html.

[53] N. Tripuraneni, B. Adlam, and J. Pennington. "Covariate Shift in High-Dimensional Random Feature Regression". In: *arXiv preprint arXiv:2111.08234* (2021). URL: https://arxiv.org/abs/2111.08234.

Bulletin

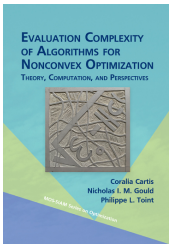
Email items to siagoptnews@lists.mcs.anl.gov *for consideration in the bulletin of forthcoming issues.*

Event Announcements

SIAM Conference on Optimization

The SIAM Conference on Optimization will be held in person between May 31 and June 3, 2023 in the Sheraton Grand Seattle Hotel, in Seattle, WA. It will be co-located with SIAM Conference on Applied and Computational Discrete Algorithms (ACDA23). The deadlines are October 31st, 2022 for submitting minisymposium proposals and November 28, 2022 for contributed presentations.

Books



Evaluation Complexity of Algorithms for Nonconvex Optimization: Theory, Computation and Perspectives
By Coralia Cartis, Nicholas I. M. Gould, Philippe L. Toint
Publisher: SIAM
ISBN: 978-1-611976-98-4
Published: 2022
https://my.siam.org/Store/Product/viewproduct/?ProductId=41813864

About the book: How many function evaluations are needed for solving a nonconvex optimization problem? This book addresses this question, covering composite and constrained optimization, derivative-free optimization, bounds on nonconvex problems. It also proposes alternative optimality measures and compares new with traditional methods from a complexity standpoint.

Audience: The book is intended for anyone who wants to solve nonconvex optimization problems. It is suitable for advanced undergraduate and graduate students in courses on advanced numerical analysis, data science, numerical optimization, and approximation theory.

Business Dynamics Models: Optimization-Based One Step Ahead Optimal Control
By Eugenius Kaszkurewicz and Amit Bhaya
Publisher: SIAM
ISBN: 978-1-611977-30-1
Published: 2022
https://my.siam.org/Store/Product/viewproduct/?ProductId=44350025

About the book: this book introduces optimal control methods, formulated as optimization problems, applied to business dynamics problems. Business dynamics is a combination of business management and financial objectives embedded in a dynamical system model. The model is subject to a control that optimizes a performance index and takes both management and financial aspects into account.

Audience: advanced undergraduate and gratudate students in applied mathematics, business, and engineering, who will enjoy the formulation-algorithm-example approach used by the authors.

Chair's Column

Katya Scheinberg, SIAG/OPT Chair
Cornell University, Ithaca, NY 18015-1582, USA
katyas@cornell.edu
https:/www.orie.cornell.edu/faculty-directory/katya-scheinberg

This is my third column for the Views and News as the end of my term as the SIAG/OPT Chair is approaching. Current SIAG/OPT officers are stepping down at the end of 2022 and election of the new officers are on the way. Please be sure to vote. The three years have gone by incredibly fast, largely overshadowed by the pandemic. But as we are seeing the return of our regular functions and in person events, I am happy to report that our SIAG remains healthy and thriving. As of May 15, 2022, the SIAG had 1,281 members, including 786 student members. We expect these numbers to increase once the registration for SIAM Conference on Optimization (OP23) starts rolling.

The conference is to be held in person, May 31st-June 3rd, 2023 in the Sheraton Grand Seattle Hotel, in Seattle, WA. It will be collocated with SIAM Conference on Applied and Computational Discrete Algorithms (ACDA23) and will feature plenary talks by **Russell Allgor**, Amazon, U.S., **Hedy Attouch**, Université Montpellier, France, **James V. Burke**, University of Washington, U.S., **Jesús A. De Loera**, University of California, Davis, U.S., **Fatma Kılınç-Karzan**, Carnegie Mellon University, U.S., **Andrea Walther**, Humboldt-Universität zu Berlin, Germany, **Wolfram Wiesemann**, Imperial College London, United Kingdom and **Tong Zhang**, The Hong Kong University of Science and Technology, Hong Kong. Coralia Cartis, University of Oxford, U.K., Jeff Linderoth, University of Wisconsin, U.S. and myself co-chair the conference program. The deadline for OP23 minisymposium proposal submission deadline is October 31st 2022 and for contributed submissions it is November 28th, 2022.

For the first time, there will be three SIAG prizes awarded at the meeting: the Best Paper prize, established in 1992, the Early Career prize, established in 2018, and the **newly established Test of Time award**, which will be awarded to an individual or group of researchers for an outstanding single piece of work that has had significant and sustained influence on the field of optimization over a time period of at least 10 years preceding the year of the award. The winners of all SIAG awards will be invited to present their work in a special award session at the conference. This information and more can be found on the new SIAG website at https://siagoptimization.github.io.

We are happy to announce two of SIAG/OPT highly deserving members to become SIAM Fellows in 2022: **Sharon Arroyo** from The Boeing Company and **James Crowley**, a former executive director of SIAM. We would like to see more new Fellows among the SIAG/OPT members in the upcoming years and would encourage current Fellows to nominate their eligible colleagues.

I would like to finish this column by welcoming two new co-editors of this newsletter – Dmitriy Drusvyatskiy (University of Washington) and Matt Menickelly (Argonne National Labs). Best wishes for the holiday season to all.

Comments from the Editors

This edition of "Views and News" includes three excellent contributed articles. The first article, by Jason Altschuler, elaborates on the deep (and often-ignored) connections between the optimal transport problem and the minimum mean cycle problem on graphs, and the matrix scaling and matrix balancing problems of preconditioning. These connections lead to cross-cutting benefits in algorithmic implementation and convergence analysis.

The second article, provided by Madeleine Udell and Zachary Frangella, highlights trends and tools

of randomized numerical linear algebra. Because so many optimization methods depend critically on linear system solves, which can become unwieldy when the problem data is large, employing randomization to decrease computational effort at the expense of accuracy is a fundamental tradeoff that merits continued and optimization-specific study. This article highlights precisely these considerations.

Our third and final article, written by Courtney Paquette and Elliot Paquette, suggests that the traditional analysis of empirical risk minimization through an optimization lens ignores – essentially – the fact that many such problems in practice are high-dimensional, both in the number of observations/data and the number of parameters/decision variables. High-dimensional stochastic optimization naturally benefits from many deep results in high-dimensional probability and random matrix theory. This article presents a number of results in this vein, and provides numerical evidence showing that high dimensional analysis can better predict the average behavior of stochastic gradient-type methods.

Let us remind you that all issues of *Views and News* are available at the online archive: http://wiki.siam.org/siag-op/index.php/View_and_News.

The SIAG/OPT Views and News mailing list, where editors can be reached for feedback, is siagoptnews@lists.mcs.anl.gov. Suggestions for new issues, comments, and papers are always welcome.

Pietro Belotti
DEIB, Politecnico di Milano
Email: pietro.belotti@polimi.it
Web: https://belotti.faculty.polimi.it

Dmitriy Drusvyatskiy
Mathematics Department, University of Washington
Email: ddrusv@uw.edu
Web: sites.math.washington.edu/~ddrusv

Matt Menickelly
Argonne National Laboratory
Email: mmenickelly@anl.gov
Web: www.mcs.anl.gov/~menickmj