Recent and Upcoming Developments in Randomized Numerical Linear Algebra for ML

Michał Dereziński and Michael W. Mahoney

University of Michigan and ICSI / LBNL / UC Berkeley

December 11, 2023

Outline

Part I
Foundations of RandNLA

Part II
Recent and Upcoming Advances

Part I

Foundations of RandNLA

Part I: Foundations of RandNLA

"Concluding thoughts"

RandNLA:

- **Major interdisciplinary area**: b/w TCS, NLA, scientific computing, ML
- **Recent years: major developments** in implementation (e.g., GPUs, hardware) and application (ML vs scientific/engineering ML) motivations
- Existing (strong) theoretical foundations: needs revisiting (updating)
- **Recent years: major developments** in theory
- Leads to improvements: in {inferential objectives, iterative algorithms, etc.} for {old, new} ML problems
- **RandBLAS/RandLAPACK**: implement theory "lower in the stack"

Main theme for today:

- Identify core linear algebraic structures and build algorithmic / statistical methods around them (rather than tacking them on later as a "band aid")
- A good way to build robust (theoretical and practical) ML pipelines and avoid lots of problems later ...

Part I: Foundations of RandNLA

RandNLA: Randomized Numerical Linear Algebra

Matrices provide a natural structure with which to **model data**.

- $A \in \mathbb{R}^{m \times n}$ can encode information about m objects, each of which is described by n *features*; etc.
- A positive definite $A \in \mathbb{R}^{n \times n}$ can encode the correlations/similarities between all pairs of n objects; etc.

Motivated by data problems, recent years have witnessed **many exciting developments** in the theory and practice of matrix algorithms.

- Particularly remarkable is the use of *randomization*.
- Typically, it is assumed to be a property of the input data due (*e.g.*, to noise in the data generation mechanisms).
- Here, it is used as an algorithmic or computational resource.

RandNLA: Randomized Numerical Linear Algebra

An *interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems.*

- Foundational perspective: roots in theoretical computer science (TCS); deep connections with convex analysis, probability theory, and metric embedding theory, etc.; and strong connections with scientific computing, signal processing, and numerical linear algebra (NLA).
- Implementational perspective: well-engineered RandNLA algorithms beat highly-optimized software libraries for problems such as very over-determined least-squares and scale well to parallel/distributed environments.
- Data analysis perspective: strong connections with machine learning and statistics and many "non-methodological" applications of data analysis.

Growing interest in providing an *algorithmic and statistical foundation for modern large-scale data analysis.*

Randomized numerical linear algebra (RandNLA)

Lots of reviews of the past from multiple different perspectives.

- Tutorials, light on prerequisites
  - "RandNLA: randomized numerical linear algebra," by Drineas and Mahoney [DM16]
  - "Lectures on randomized numerical linear algebra," by Drineas and Mahoney [DM18]
- Broad and proof-heavy resources
  - "Sketching as a tool for numerical linear algebra," by Woodruff [Woo14]
  - "An introduction to matrix concentration inequalities," by Tropp [Tro15]
  - "Lecture notes on randomized linear algebra," by Mahoney [Mah16]
- Perspectives on theory, light on proofs
  - "Randomized algorithms for matrices and data," by Mahoney [Mah11]
  - "Determinantal point processes in randomized numerical linear algebra," by Dereziński and Mahoney [DM21]
- Deep investigations of specific topics
  - "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," by Halko, Martinsson, and Tropp [HMT11]
  - "Randomized algorithms in numerical linear algebra," by Kannan and Vempala [KV17]
  - "Randomized methods for matrix computations," by Martinsson [Mar18]
  - "Randomized numerical linear algebra: Foundations and Algorithms," by Martinsson and Tropp [MT20]

We will be describing and highlighting upcoming and future trends.

RandNLA: Randomized Numerical Linear Algebra

- **"Classical" RandNLA**:
  - Sample/project and then solve subproblem or construct preconditioner
  - Theory from TCS/NLA, typically based on JL / subspace embeddings
  - Lots of data/ML and scientific computing applications
  - Initial proof-of-principle implementations (low-rank approximation, least-squares, optimization, etc.)
  - Relatively large theory-practice gap (esp. when used in ML pipelines)
- **"Modern" RandNLA**:
  - More sophisticated theory going beyond worst-case JL / subspace embeddings, with stronger connections to RMT
  - Improved statistical analysis and improved optimization algorithms
  - Implementations in **RandBLAS/RandLAPACK**, and more demands from GPU-based ML model training and scientific computing
  - Smaller theory-practice gap
- Opens up door to new theory, new implementations, new applications, ...

Basic Principles of "Classical" RandNLA [DM16]

Basic RandNLA method: given an input matrix:

- Construct a "sketch" (a smaller or sparser matrix that represents the essential information in the original matrix) by random sampling.
- Use that sketch as a surrogate to compute quantities of interest.

Basic design principles[1] underlying RandNLA:

- Randomly sample (in a careful data-dependent manner) a small number of elements to create a much sparser sketch of the original matrix.
- Randomly sample (in a careful data-dependent manner) a small number of columns and/or rows to create a much smaller sketch of the original matrix.
- Preprocess an input matrix with a random-projection-type matrix and then do uniform sampling of rows/columns/elements in order to create a sketch.

[1] First two principles deal with identifying nonuniformity structure. Third principle deals with preconditioning input (*i.e.*, uniformizing nonuniformity structure) s.t. uniform random sampling performs well.

Part I: Foundations of RandNLA

Part I: Foundations of RandNLA

Approximating Matrix Multiplication [DKM06]

Problem Statement: Given an $m \times n$ matrix A and an $n \times p$ matrix B, approximate the product $A \cdot B$.

Approximating Matrix Multiplication [DKM06]

Problem Statement: Given an $m \times n$ matrix A and an $n \times p$ matrix B, approximate the product $A \cdot B$.

OR, equivalently,

Problem Statement: Approximate the sum of n rank-one matrices.

$$A \cdot B = \sum_{k=1}^{n} \underbrace{\begin{pmatrix} \\ A_{*k} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \qquad B_{k*} \qquad \end{pmatrix}}_{\in \mathbb{R}^{m \times p}}$$

Approximating Matrix Multiplication [DKM06]

A sampling approach:

1. Fix a set of probabilities p_i, $i = 1, \ldots, n$, summing up to 1.
2. For $t = 1, \ldots, c$,

   set $j_t = i$, where $\mathbb{P}[j_t = i] = p_i$.

   (Pick c terms of the sum, with replacement, with respect to the p_i.)
3. Approximate the product AB by summing the c terms, after scaling.

$$A \cdot B = \sum_{k=1}^{n} \begin{pmatrix} \\ A_{*k} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \quad B_{k*} \quad \end{pmatrix} \approx \sum_{t=1}^{c} \frac{1}{cp_{j_t}} \begin{pmatrix} \\ A_{*j_t} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \quad B_{j_t*} \quad \end{pmatrix}$$

Approximating Matrix Multiplication [DKM06]

The same algorithm, in matrix notation:

1. Pick c columns of A to form an $m \times c$ matrix C and the corresponding c rows of B to form a $c \times p$ matrix R.
2. Rescale the columns/rows prior to including them in C/R.
3. Approximate $A \cdot B$ by $C \cdot R$.

$$\underset{m \times n}{\left(A \right)} \left(B \right) \approx \underset{m \times c}{\left(C \right)} \underset{c \times p}{\left(R \right)}$$

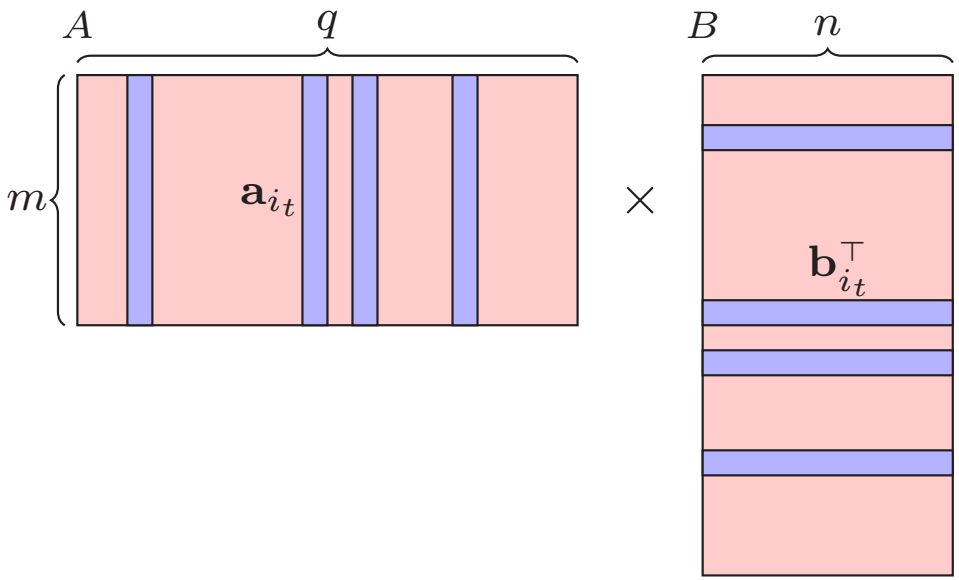
Can use a "sampling matrix" formalism:

- Let S be $n \times c$ matrix whose t^{th} column $(t = 1, \ldots, c)$ has one non-zero:

$$S_{j_t t} = \frac{1}{\sqrt{c p_{j_t}}}$$

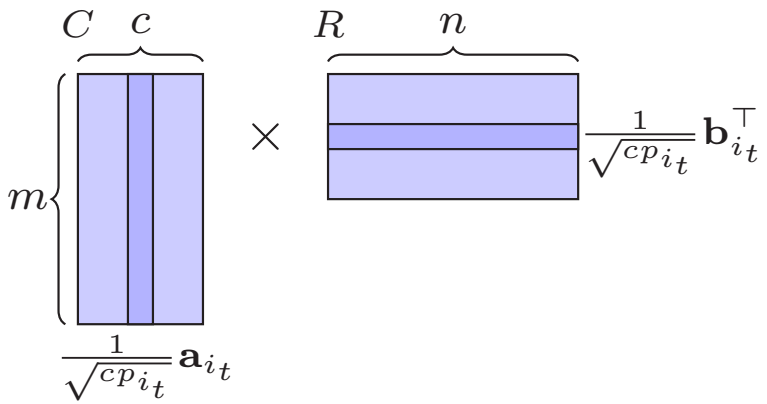
- Clearly: $A \cdot B \approx C \cdot R = (AS) \cdot \left(S^T B\right)$.

Approximating Matrix Multiplication [DKM06]



$$O(mqn)$$

$$AB \approx \frac{1}{c}\sum_{t=1}^{c}\frac{1}{p_{i_t}}a_{i_t}b_{i_t}^\top = CR \qquad \Downarrow$$



$$O(mcn)$$

Approximating Matrix Multiplication [DKM06]

Some simple lemmas:

- For any sampling probabilities:

$$\begin{aligned} \mathbb{E}[(CR)_{ij}] &= (AB)_{ij} \\ \mathrm{Var}[(CR)_{ij}] &= \frac{1}{c}\sum_{k=1}^{n}\frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2 \end{aligned}$$

- From these, it's easy to bound $\mathbb{E}\left[\|AB - CR\|_F\right]$.
- Remove the expectation with Markov's inequality or a martingale argument.
- To minimize $\mathbb{E}\left[\|AB - CR\|_F\right]$, use these probabilities:

$$\mathbb{P}[j_t = i] = \frac{\|A_{*i}\|_2 \|B_{i*}\|_2}{\sum_{j=1}^{n} \|A_{*j}\|_2 \|B_{j*}\|_2} \tag{1}$$

- This gives:

$$\mathbb{E}\left[\|AB - CR\|_F\right] = \mathbb{E}\left[\|AB - ASS^T B\|_F\right] \leq \frac{1}{\sqrt{c}}\|A\|_F \|B\|_F \tag{2}$$

- Similar bounds to (2) if approximate probabilities (1) in one of many ways.

Approximating Matrix Multiplication [DKM06]

This *Frobenius norm* bound is used in *many* places in RandNLA, but ...

a "better" *spectral norm* bound is possible via Chernoff/Bernstein inequalities.

Lemma ([DMMS10] Thm 4)

Assume:

- $\|A\|_2 \leq 1$: *("not important," just normalization)*
- $\|A\|_F \geq 0.2$: *("not important," simplifies bounds)*

Set:

$$c = \Omega\left(\frac{\|A\|_F^2}{\epsilon^2} \ln\left(\frac{\|A\|_F^2}{\epsilon^2\sqrt{\delta}}\right)\right).$$

Then, for any $\epsilon \in (0,1)$, *w.p.* $\geq 1-\delta$, *we have:*

$$\|AA^T - CC^T\|_2 = \|AA^T - ASS^TA^T\|_2 \leq \epsilon.$$

Approximating Matrix Multiplication [DKM06]

The spectral norm bound is "better," but:

- It only holds for $B = A^T$, so it doesn't hold for arbitrary AB.
- The "not important" conditions mean it doesn't hold for arbitrary A.

The "main use case" for the spectral norm bound:

- Let A^T be an $n \times d$ matrix U with orthonormal columns, where $n \gg d$.
- Then $U^T U = I_d$, and we want to show that

$$\|U^T S S^T U - U^T U\|_2 = \|U^T S S^T U - I_d\|_2 \leq \epsilon \in (0, 1).$$

- Using the Frobenius norm bound, we get

$$\|U^T S S^T U - I\|_2 \leq \|U^T S S^T U - I\|_F \leq \frac{1}{\sqrt{c}} \|U\|_F^2 = \frac{d}{\sqrt{c}}.$$

- Using the spectral norm bound, we get

$$\|U^T S S^T U - I\|_2 \lesssim \frac{\ln c}{\sqrt{c}} \|U\|_F \|U\|_2 = \frac{\sqrt{d} \ln c}{\sqrt{c}}.$$

Subspace Embeddings [Mah11, Woo14]

Definition

Let U be an $m \times n$ orthogonal matrix, and let S be any $n \times m$ matrix. Then, S is a *subspace embedding* if

$$\|U^T U - (SU)^T SU\|_2 = \|I - (SU)^T SU\|_2 \leq \epsilon.$$

Things to note:

- Many constructions (random sampling and projection methods, deterministic constructions, hashing functions, etc.) satisfy this condition.
- First used in data-aware context with leverage score sampling [DMM06, DMM08]
- Used in data-oblivious context with Hadamard-based projections [Sar06, DMMS10]
- For NLA, this is an acute perturbation.
- For TCS, this is a subspace analogue of JL lemma.

This is a "must must have" for TCS; for everyone else, it's optional.

- Numerical implementations: loosing rank still gives a good preconditioner.
- Statistics and machine learning: loosing rank introduces a bit of bias.

Part I: Foundations of RandNLA

Least-squares approximation

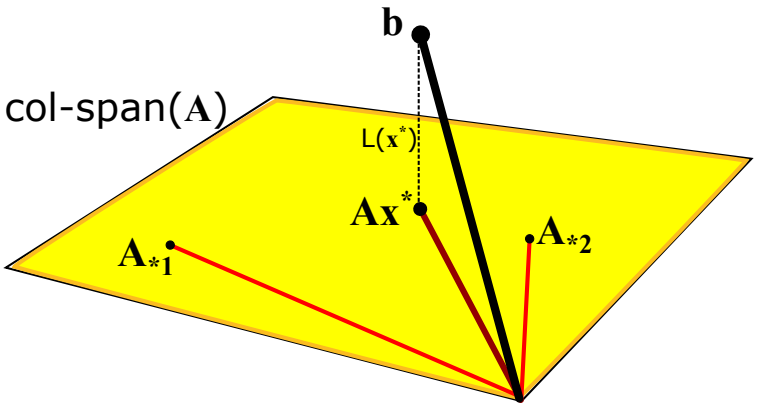
Least-squares (LS) : given $m \times n$ matrix A and m-dimensional vector b, solve

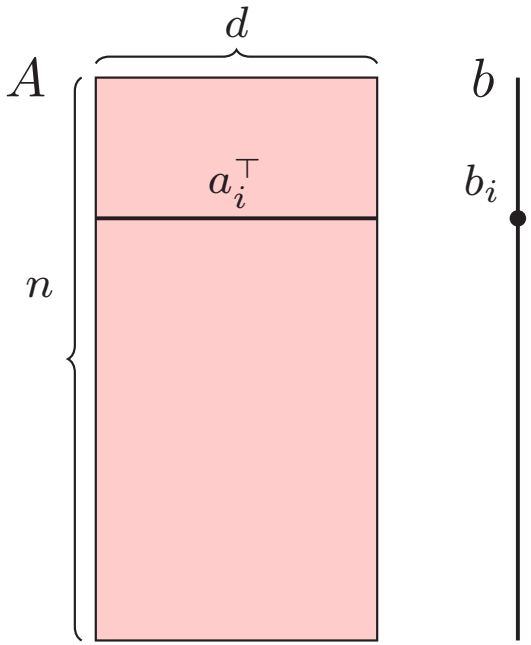
$$x_{opt} = \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_2.$$

- If $m \gg n$, it is overdetermined/overconstrained.
- Compute solution in $O(mn^2)$ time (in RAM model) with one of several methods: normal equations; QR decompositions; or SVD.
- RandNLA provides faster algorithms for this ubiquitous problem.
  - **TCS**: faster in terms of low-precision asymptotic worst-case theory.
  - **NLA**: faster in terms of high-precision wall-clock time.
  - **Implementations**: can compute (in Spark/MPI/etc.) low, medium, and high precision solutions on up to terabyte-sized data.
  - **Data Applications**: faster algorithms and/or implicit regularization for many machine learning and data science problems.
- *The basic RandNLA approach extends to many other matrix problems.*

Visualizing least-squares

Given: n points a_i, each in d-dimensional space,[2] with labels $b_i \in \mathbb{R}$
Goal: Minimize loss $L(x) = \sum_i (a_i^\top x - b_i)^2 = \|Ax - b\|^2$
Goal′: Find $x^* = \text{argmin}_x L(x)$





$O(nd^2)$ operations

Let col-span(A) be the column span of A. Then:

$$Ax^* = \underset{v \in \text{col-span}(A)}{\text{argmin}} \|v - b\|^2 = \underbrace{AA^\dagger}_{\text{projection}} \cdot b$$

[2] Oops: $\{n, d\} \leftrightarrow \{m, n\}$; be careful!!!

Two important notions: leverage and condition [Mah11]

- **Statistical leverage.** (Think: eigenvectors. Important for low-precision.)
  - The *statistical leverage scores* of A (assume $m \gg n$) are the diagonal elements of the projection matrix onto the column span of A.
  - They equal the ℓ_2-norm-squared of any orthogonal basis spanning A.
  - They measure:
    - how well-correlated the singular vectors are with the canonical basis
    - which constraints have largest "influence" on the LS fit
    - a notion of "coherence" or "outlierness"
  - Computing them exactly is as hard as solving the LS problem.
- **Condition number.** (Think: eigenvalues. Important for high-precision.)
  - The *ℓ_2-norm condition number* of A is $\kappa(A) = \sigma_{\max}(A)/\sigma^{+}_{\min}(A)$.
  - $\kappa(A)$ bounds the number of iterations; for ill-conditioned problems (e.g., $\kappa(A) \approx 10^6 \gg 1$), the convergence speed is very slow.
  - Computing $\kappa(A)$ is generally as hard as solving the LS problem.

These are for the ℓ_2-norm. Generalizations exist for the ℓ_1-norm, etc.

Meta-algorithm for ℓ_2-norm regression (1 of 3)

1: Using the ℓ_2 statistical leverage scores of A, construct an importance sampling distribution $\{p_i\}_{i=1}^m$.

2: Randomly sample a small number of constraints according to $\{p_i\}_{i=1}^m$ to construct a subproblem.

3: Solve the ℓ_2-regression problem on the subproblem.

A *naïve* version of this meta-algorithm:

- gives a $1+\epsilon$ relative-error approximation, that fails with probability δ, in roughly $O(mn^2/\epsilon)$ time [DMM06, DMM08] (Ugh—seems bad—why would one do this?)

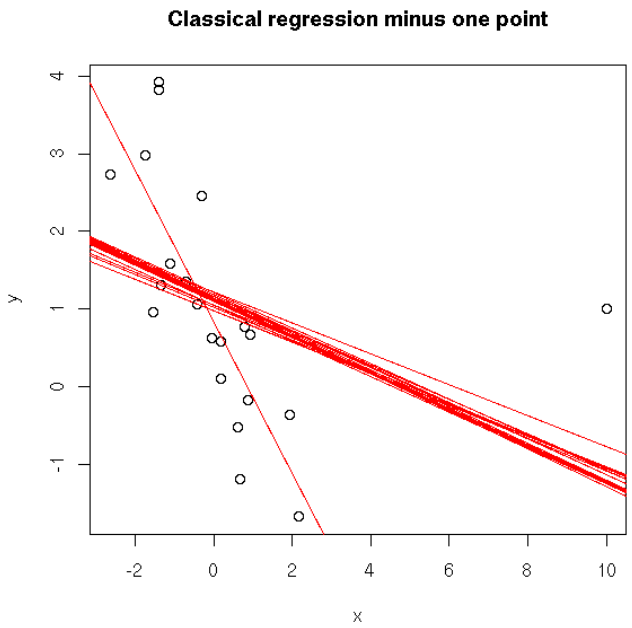
A *non-naïve* version of this meta-algorithm:

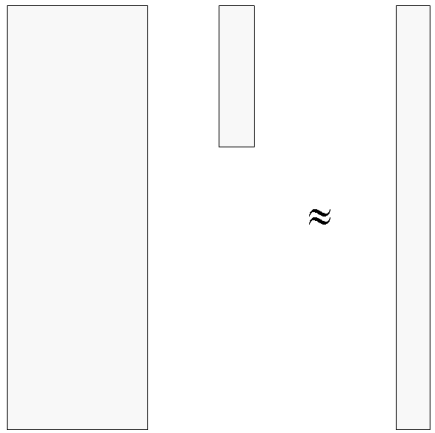
- gives the best worst-case algorithm in RAM.
- beats LAPACK for high precision in wall-clock time.
- super-terabyte-scale implementations in parallel/distributed environments.
- provides the foundation for low-rank approximations and the rest of RandNLA.

(Drineas, Mahoney, etc., starting with [DMM06]; [DMM08]; [Mah11])

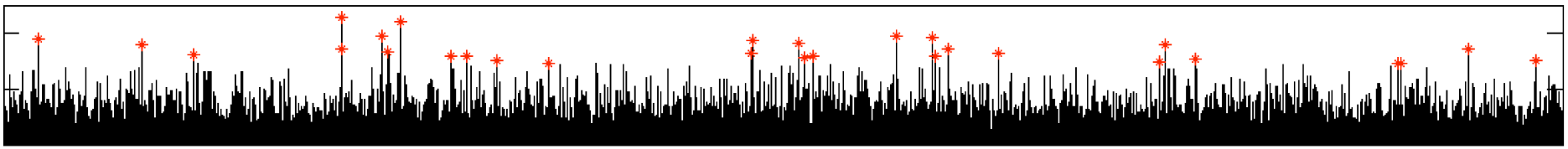
Meta-algorithm for ℓ_2-norm regression (2 of 3)

- Randomly sample high-leverage constraints
- Solve the subproblem





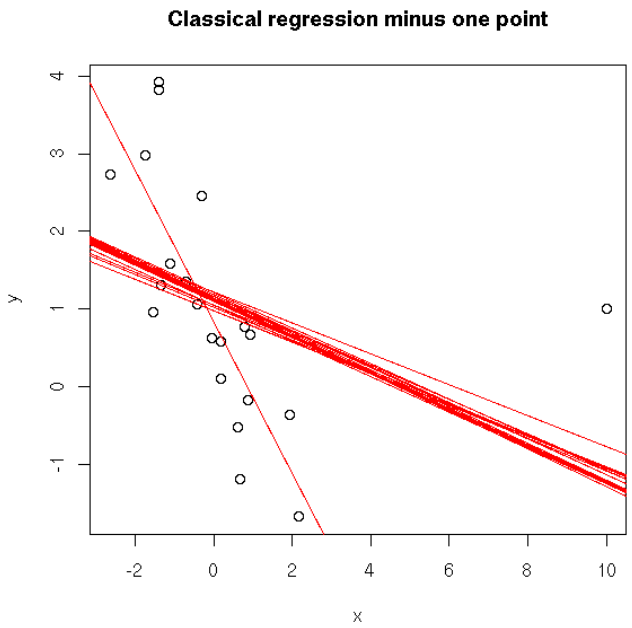
(In many moderately large-scale applications, one uses "ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)

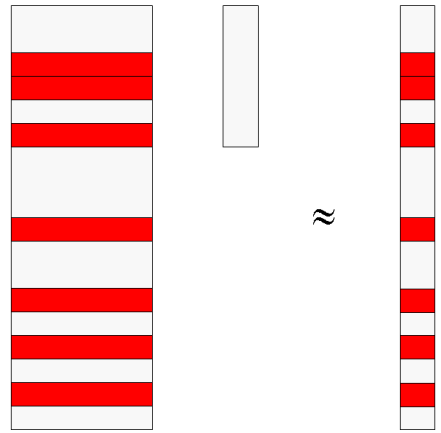


(Drineas, Mahoney, etc., starting with [DMM06]; [DMM08]; [Mah11])

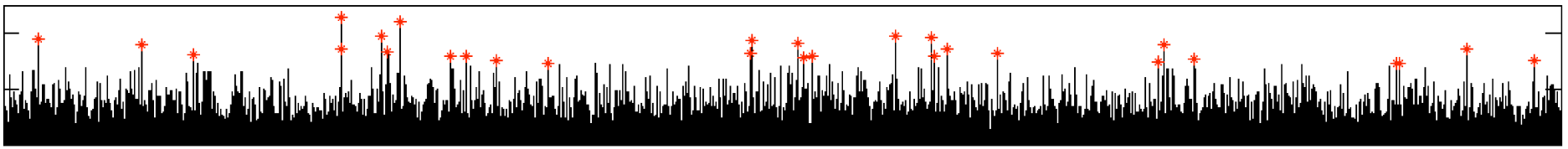
Meta-algorithm for ℓ_2-norm regression (2 of 3)

- Randomly sample high-leverage constraints
- Solve the subproblem





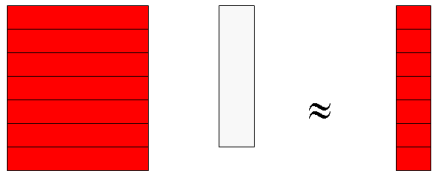
(In many moderately large-scale applications, one uses "ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)

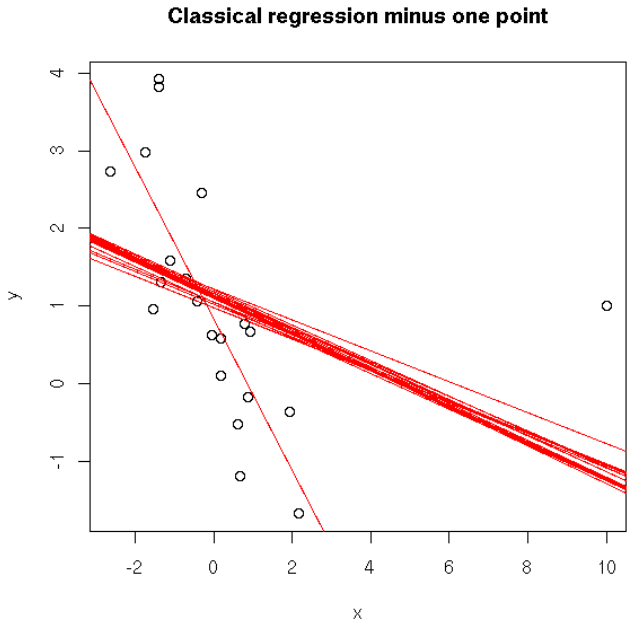


(Drineas, Mahoney, etc., starting with [DMM06]; [DMM08]; [Mah11])

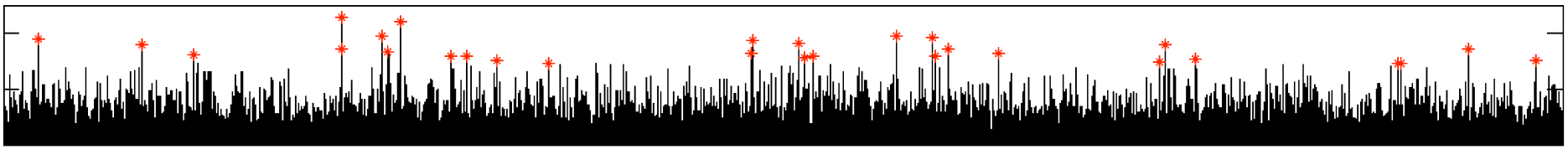
Meta-algorithm for ℓ_2-norm regression (2 of 3)

- Randomly sample high-leverage constraints
- Solve the subproblem





(In many moderately large-scale applications, one uses "ℓ_2 objectives," not since they are "right," but since other things are even more expensive.)



(Drineas, Mahoney, etc., starting with [DMM06]; [DMM08]; [Mah11])

Meta-algorithm for ℓ_2-norm regression (3 of 3)

We can make this meta-algorithm "fast" in RAM:[3]

- This meta-algorithm runs in $O(mn \log n/\epsilon)$ time in RAM if:
  - we perform a Hadamard-based random random projection and sample uniformly sampling in the randomly rotated basis, or
  - we quickly computing approximations to the statistical leverage scores and using those as an importance sampling distribution.

We can make this meta-algorithm "high precision" in RAM:[4]

- This meta-algorithm runs in $O(mn \log n \log(1/\epsilon))$ time in RAM if:
  - we use the random projection/sampling basis to construct a preconditioner and couple with a traditional iterative algorithm.
- See Blendenpik/LSRN for NLA-style wall-clock time comparisons.

Both can be improved (in theory) to run in almost $O(\mathbf{nnz}(A))$ time.

[3] (Sarlós [Sar06]; Drineas, Mahoney, Muthu, Sarlós [DMMS10]; Drineas, Magdon-Ismail, Mahoney, Woodruff [DMIMW12]

[4] (Rokhlin & Tygert [RT08]; Avron, Maymounkov, & Toledo [AMT10]; Meng, Saunders, & Mahoney [MSM14].)

Least-squares approximation: basic structural result

Consider the over-determined least-squares approximation problem:

$$\mathcal{Z}_2^2 = \min_{x \in \mathbb{R}^n} \|b - Ax\|_2^2 = \|b - Ax_{opt}\|_2^2$$

as well as the "preconditioned " the least-squares approximation problem:

$$\tilde{\mathcal{Z}}_2^2 = \min_{x \in \mathbb{R}^n} \|\Omega(b - Ax)\|_2^2 = \|b - A\tilde{x}_{opt}\|_2^2$$

where Ω is *any* matrix.

Theorem (Fundamental Structural Result for Least-Squares)

If Ω satisfies the two basic conditions (constants are somewhat arbitrary):

$$\begin{aligned} \sigma_{min}^2(\Omega U_A) &\geq 1/\sqrt{2} \\ \left\| U_A^T \Omega^T \Omega b^\perp \right\|_2^2 &\leq \epsilon \mathcal{Z}_2^2/2, \quad \text{where } b^\perp = b - U_A U_A^T A, \end{aligned}$$

then:

$$\begin{aligned} \|A\tilde{x}_{opt} - b\|_2 &\leq (1+\epsilon)\mathcal{Z}_2 \\ \|x_{opt} - \tilde{x}_{opt}\|_2 &\leq \frac{1}{\sigma_{min}(A)} \sqrt{\epsilon} \mathcal{Z}_2. \end{aligned}$$

DMMS [DMMS10, Mah11]

Least-squares approximation: satisfying the conditions

Both conditions are an approximate matrix-matrix multiplication result:

- First condition:

$$\|U_A^T U_A - U_A^T \Omega \Omega^T U_A\|_2^2 = \|I - U_A^T \Omega \Omega^T U_A\|_2^2 \leq \epsilon,$$

  w.p. $\geq 1 - \delta$, if $r = O\left(\frac{n}{\epsilon^2} \ln\left(\frac{n}{\epsilon^2 \sqrt{\delta}}\right)\right)$.

- Second condition:

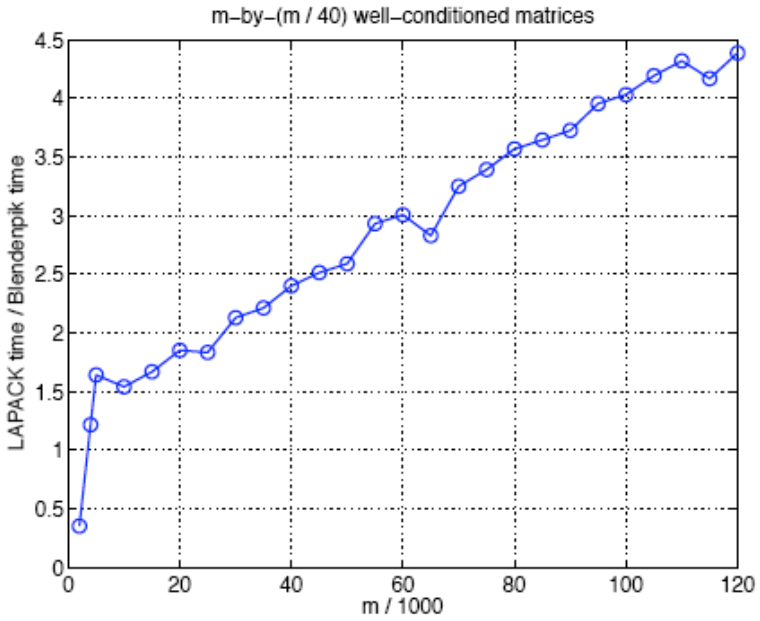
$$\mathbb{E}\left[\|U_A^T \Omega \Omega^T b^\perp - U_A^T b^\perp\|_2^2\right] \leq \frac{1}{r}\|U_A\|_F^2 \|b^\perp\|_2^2 = \frac{n}{r}\mathcal{Z}_2^2,$$

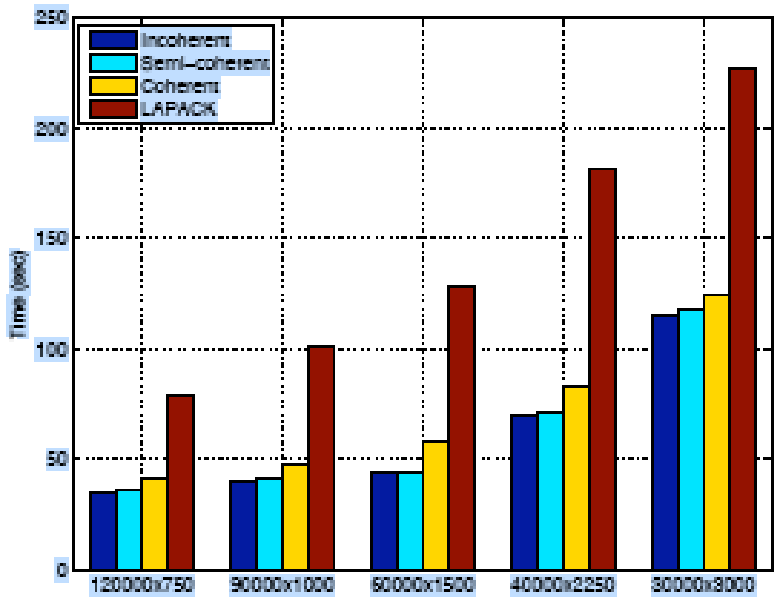
  and remove expectation with Markov.

Things to note:

- Many constructions (random sampling and projection methods, deterministic constructions, hasing functions, etc.) satisfy these conditions.
- Which construction you use depends on which you like.
- ϵs don't matter: TCS people don't care; NLA people precondition; ML/DA poeple have different pain points

Least-squares approximation: RAM implementations





Conclusions:

- *Randomized algorithms "beats Lapack's direct dense least-squares solver by a large margin on essentially any dense tall matrix."*
- *These results "suggest that random projection algorithms should be incorporated into future versions of Lapack."*

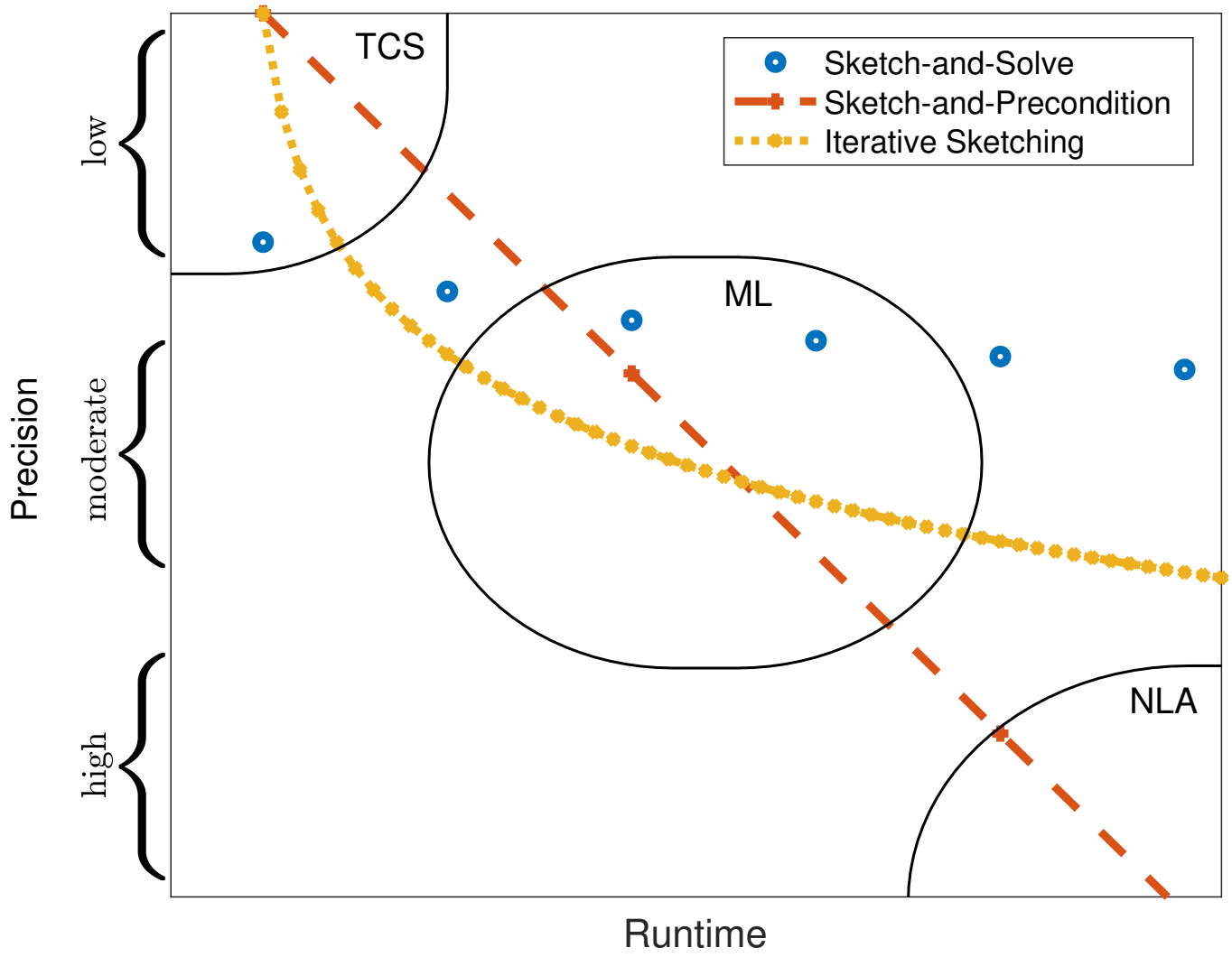
Avron, Maymounkov, and Toledo [AMT10]

Using RandNLA methods more generally ...

Three paradigms that apply more broadly than least squares:

1. Sketch-and-solve: Construct a *smaller* least squares problem; then solve it using a direct method.
   - Low-precision estimate, e.g., $\epsilon = 0.1$
   - Simplest to highlight structure of the theory
2. Iterative sketching: Repeatedly sketch/sub-sample the problem; and iteratively refine the estimate.
   - Medium (to high, depending on method) precision estimate, e.g., $\epsilon = 10^{-3}$
   - SGD, SGD++, sketch-and-project, preconditioned weighted SGD
3. Sketch-and-precondition: Construct an *equivalent* but well-conditioned problem; then use a deterministic iterative method.
   - High-precision solution, e.g., $\epsilon = 10^{-10}$
   - Best (usually) for high-quality numerical solutions

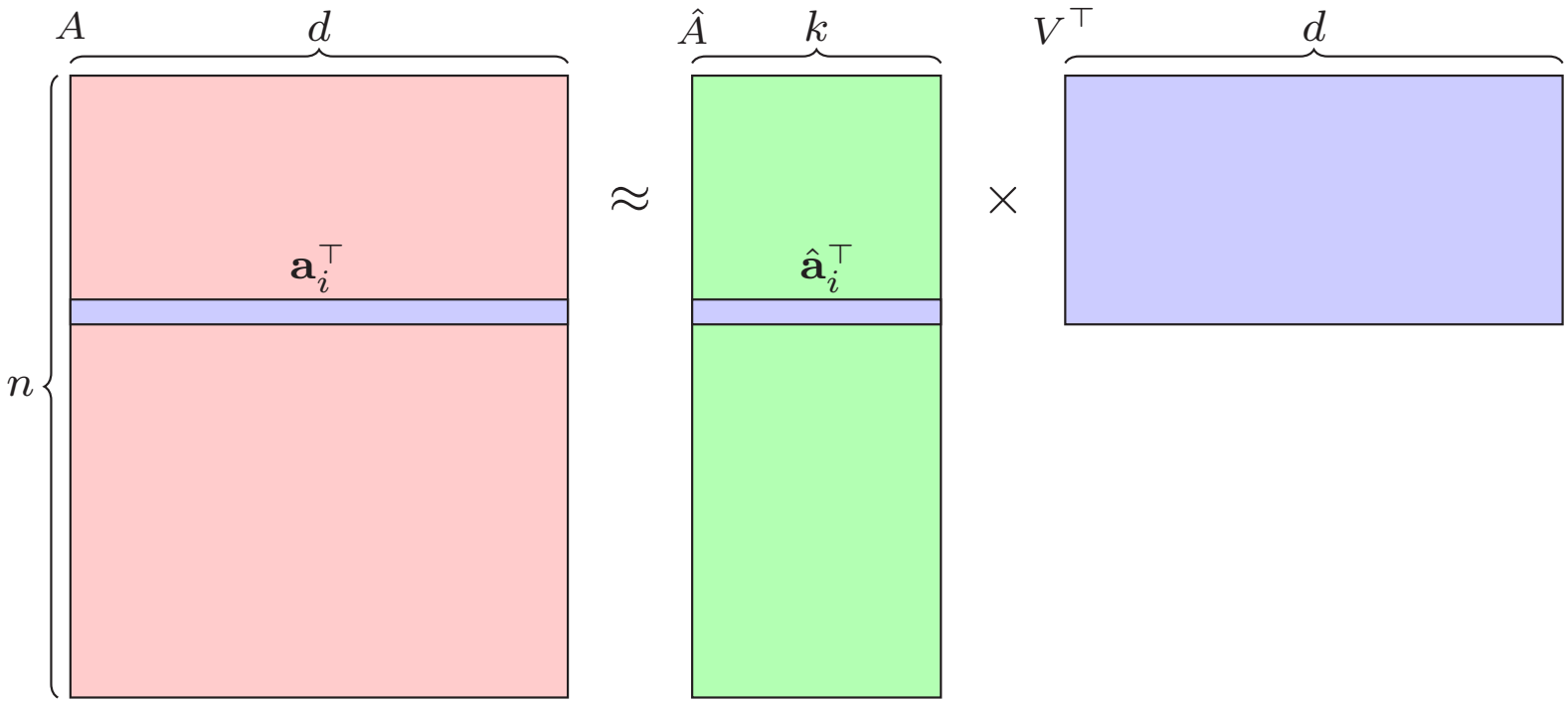
Using RandNLA methods more generally ...



Part I: Foundations of RandNLA

Low-rank approximation

$$A \approx \tilde{A} = \hat{A}V^\top = AVV^\top$$



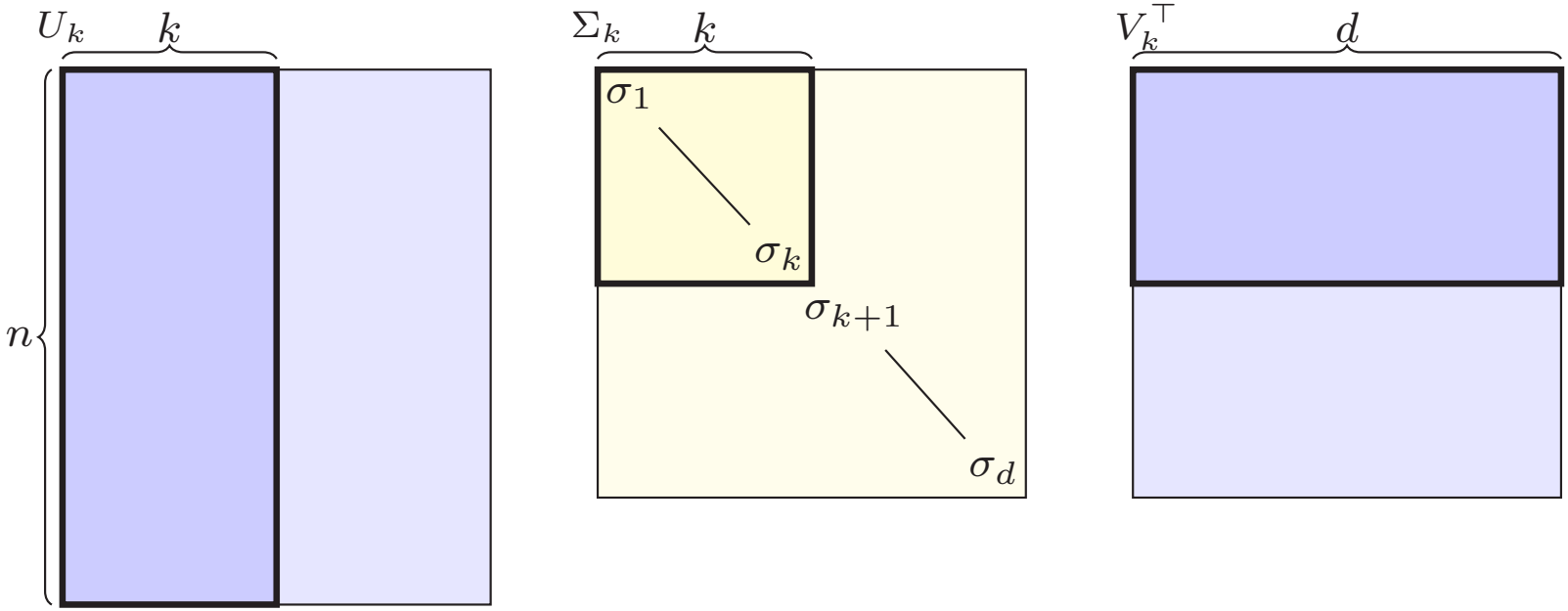
A - Original data matrix
V - Embedding between high and low dimension

$\hat{A} = AV$ - Low-dimensional representation
$\tilde{A} = \hat{A}V^\top$ - Low-rank approximation matrix

Truncated SVD

If A has rank higher than k (up to $\min\{n, d\}$), then:

- Use truncated SVD, $A \approx U_k \Sigma_k V_k^\top$ (top k singular values).
- Costs $O(nd^2)$ for $n \geq d$ (computing the SVD).



Recall that we order the singular values so that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_d$.

Error of truncated SVD

- How much information is lost in a truncated SVD?

$$A - A_k = \sum_{i>k} \sigma_i u_i v_i^\top$$

- In terms of spectral norm error, this is:

$$\|A - A_k\| = \sigma_{k+1}.$$

- In terms of Frobenius norm error, this is:

$$\|A - A_k\|_F = \left(\sum_{i>k} \sigma_i^2 \right)^{1/2}.$$

- For most norms, this is an optimal rank-k approximation:

$$A_k = \operatorname*{argmin}_{B\,:\,\mathrm{rank}(B)=k} \|A - B\|.$$

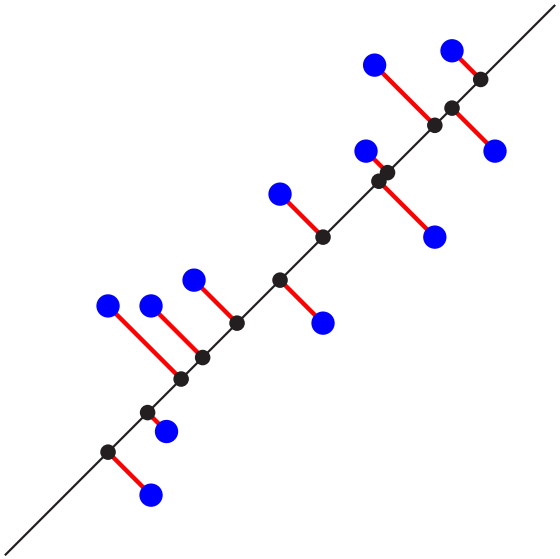
Truncated SVD as a projection

- $P = V_k V_k^\top$ is the projection onto the span of $v_1, ..., v_k$.
- We can use that to derive A_k as a projection of A:

$$AP = \sum_{i=1}^{d} \sigma_i u_i v_i^\top P = \sum_{i=1}^{k} \sigma_i u_i v_i^\top = U_k \Sigma_k V_k^\top = A_k,$$

Total projection error equals squared Frobenius norm error:

$$\sum_{i=1}^{n} \|a_i - Pa_i\|^2 = \|A - A_k\|_F^2.$$



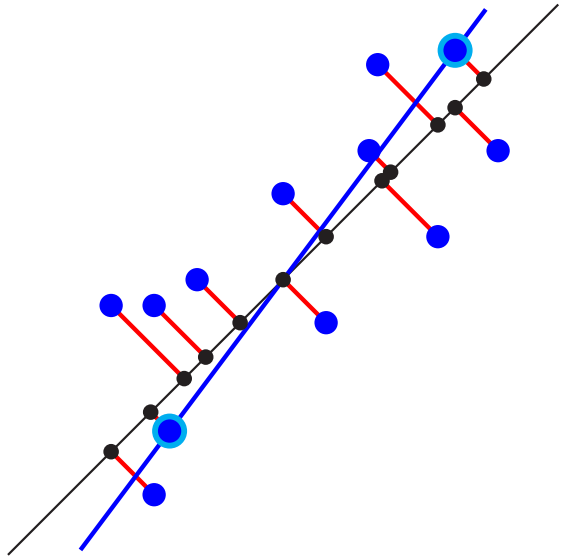
Column/row subset selection

Instead of the singular vectors, project onto the span of a subset of data points $\{a_i : i \in S\}$ for $S \subseteq \{1, ..., n\}$.

- Does not require computing SVD, but hard to find best S.
- Preserves the structure of the data.

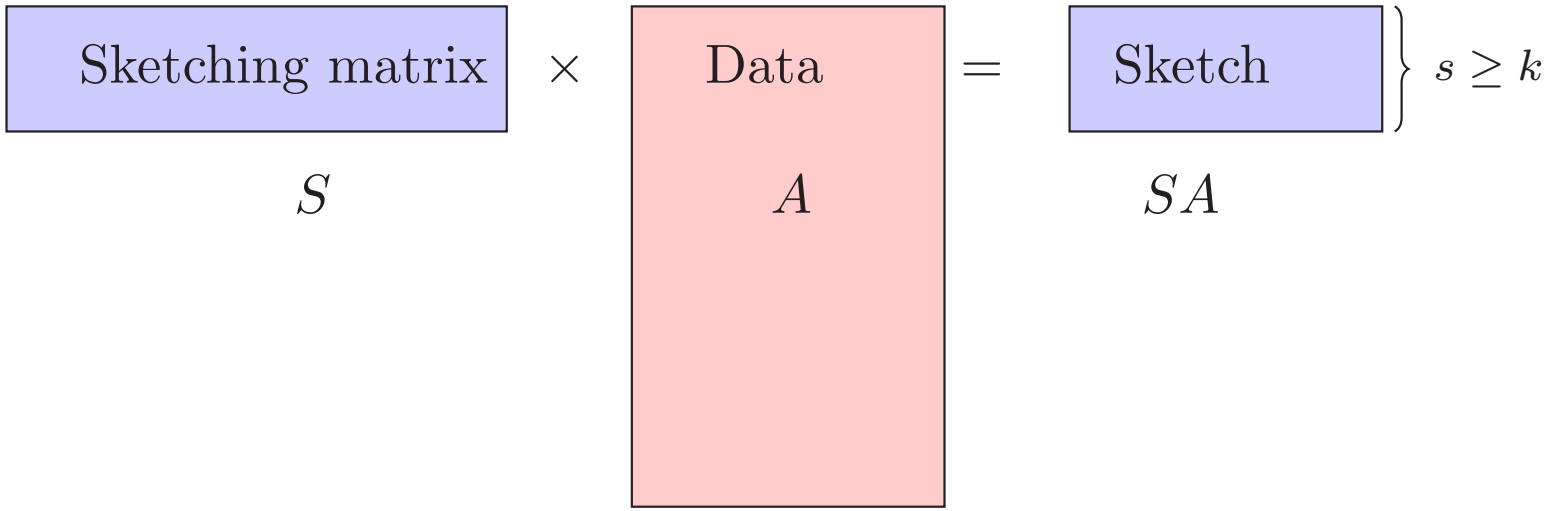
Let $P_S = VV^\top$ be a projection onto the row-span of A_{S*}, where V is the embedding matrix.

$$\text{Find} \quad \operatorname*{argmin}_{S:|S|=k} \|A - AP_S\|_F$$



Sketched low-rank approximation

Strategy: Extract top k singular vectors from a sketch



Let $P = VV^\top$ be the projection onto the row-span of SA.

$$\text{Approximation:} \qquad A \approx AP = \underbrace{(AV)}_{\hat{A}} V^\top$$

Extensions to Low-rank Approximation (Projections)

In scientific computing, goal is to find a good basis for the span of A ...

Input: $m \times n$ matrix A, target rank k and over-sampling parameter p

Output: Rank-$(k+p)$ factors U, Σ, and V s.t. $A \approx U\Sigma V^T$.

1. Draw a $n \times (k+p)$ Gaussian random matrix Ω.
2. Form the $n \times (k+p)$ sample matrix $Y = A\Omega$.
3. Compute an orthonormal matrix Q s.t. $Y = QQ^TY$.
4. Form the small matrix $B = Q^TA$.
5. Factor the small matrix $B = \hat{U}\Sigma V^T$.
6. Form $U = Q\hat{U}$.

Can prove bounds of the form:

$$\|A - QQ^TA\|_F \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2\right)^{1/2}$$

$$\|A - QQ^TA\|_2 \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2\right)^{1/2}$$

Question: How does one prove bounds of this form?

Halko, Martinsson, and Tropp [HMT11]

Extensions to Low-rank Approximation (Sampling)

Answer: **Basic structural result for RLA low-rank matrix approximation.**

Lemma (Fundamental Structural Result for Low-Rank [MD16])

Given $A \in \mathbb{R}^{m \times n}$, let $V_k \in \mathbb{R}^{n \times k}$ be the matrix of the top k right singular vectors of A. Let $\Omega \in \mathbb{R}^{n \times r}$ $(r \geq k)$ be any matrix such that $Y^T\Omega$ has full rank. Then, for any unitarily invariant norm ξ,

$$\|A - P_{A\Omega}A\|_\xi \leq \|A - A_k\|_\xi + \|\Sigma_{k,\perp}\left(V_{k,\perp}^T\Omega\right)\left(V_k^T\Omega\right)^+\|_\xi.$$

Given this structural result, we obtain results for

- the Column Subset Selection Problem ([BMD09])
- random projections for low-rank matrix approximations ([RST09, HMT11], etc.)
- improved Nyström-based low-rank SPSD matrix approximations ([GM16])
- developing improved feature selection methods (many)
- power, Lanczon, and other low-rank matrix approximation methods ...

BMD, "CSSP" 2009 [BMD09]; MD, "Structural properties ..." 2016 [MD16].

Extensions to Low-rank Approximation (SPSD)

- *SPSD Sketching Model.* Let A be an $n \times n$ positive semi-definite matrix, and let S be a matrix of size $n \times \ell$, where $\ell \ll n$. Take

$$C = AS \quad \text{and } W = S^T AS.$$

Then CW^+C^T is a low-rank approximation to A with rank at most ℓ.

Lemma (Fundamental Structural Result for SPSD Low-Rank)

Let A be an $n \times n$ SPSD matrix s.t. $A = U\Sigma U^T$, where U_1 is top k eigenvalues, $\Omega_1 = U_1^T S$, etc., and let S be a sampling/sketching matrix of size $n \times \ell$. Then

$$\begin{aligned}
\|A - CW^\dagger C^T\|_2 &\leq \|\Sigma_2\|_2 + \|\Sigma_2^{1/2}\Omega_2\Omega_1^\dagger\|_2^2, \\
\|A - CW^\dagger C^T\|_F &\leq \|\Sigma_2\|_F + \sqrt{2}\|\Sigma_2\Omega_2\Omega_1^\dagger\|_F + \|\Sigma_2^{1/2}\Omega_2\Omega_1^\dagger\|_F^2 \\
\|A - CW^\dagger C^T\|_{Tr} &\leq Tr(\Sigma_2) + \|\Sigma_2^{1/2}\Omega_2\Omega_1^\dagger\|_F^2
\end{aligned}$$

assuming Ω_1 has full row rank.

- From this, easy to derive additive-error approximations for spectral and Frobenius norm (with scale set by Trace norm error) and relative-error approximation for Trace norm in "random projection time."

Gittens and Mahoney, "Revisiting the Nystrom Method ...," [GM16]

Randomized Block Power Method Analysis

- Initialize $\tilde{V}_0 \in \mathbb{R}^{d \times k}$ (e.g., i.i.d sub-Gaussian entries)
- $\tilde{V}_{i+1} = orthonormalize(A^\top A \tilde{V}_i)$
- Return $\tilde{V}_q$

Hope: $\tilde{V}_q$ converges to the span(V_k); and $AV_qV_q^T$ gives a good k-rank approx of A.

Lemma (e.g., 4.15 from [Woo14])

For any orthonormal $Z \in \mathbb{R}^{d \times k}$

$$\|A - AZZ^\top\|_2 \leq \|(A^\top A)^q - (A^T A)^q ZZ^\top\|_2^{\frac{1}{2q}}$$

With $q = \mathcal{O}(\log(d)/\epsilon)$ and a random $\tilde{V}_0$, we compute $\tilde{V}_q$ such that w.h.p.

$$\|A - A\tilde{V}_q\tilde{V}_q^T\|_2 \leq (1+\epsilon) \cdot \min_{V \in \mathbb{R}^{d \times k}} \|A - AVV^T\|_2$$

Runtime: $\mathcal{O}\left(ndkq + qdk^2\right)$

- $ndkq \to q$ matrix vector multiplications.
- $qdk^2 \to q$ orthonormalizations.

Conclusion: Leads to *high precision* low-rank approximations in spectral norm.

[HMT11, Woo14, MM15, DIKMI18]

Part I: Foundations of RandNLA

1. Initial thoughts
   - Overview
2. Foundations of "classical" RandNLA
   - Matrix Multiplication
   - Least-squares Approximation
   - Low-rank Approximation
3. Foundations of "modern" RandNLA
   - Algorithmic Gaussianization via Random Matrix Theory
   - RMT for Sampling via DPPs

Part I: Foundations of RandNLA

The proportional limit

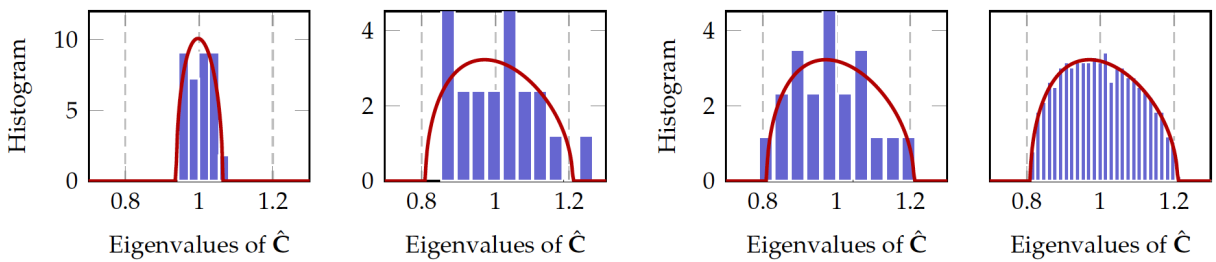


Figure: Histogram of the eigenvalues of $\hat{\mathbf{C}}$ (**blue**) versus the Marčenko-Pastur law (**red**), for $\mathbf{X}$ having standard Gaussian entries in different settings: (**left**: small versus large dimensional intuition) $p = 20, n = 1\,000p$ versus $p = 20, n = 100p$; and (**right**: non-asymptotic versus asymptotic MP law) $p = 20, n = 100p$ versus $p = 500, n = 100p$.

Consider $A \in \mathbb{R}^{n \times d}$ and iid Gaussian sketching matrix $S \in \mathbb{R}^{l \times d}$

Quality of $\tilde{A} = SA$ is often measured by $\mathrm{cond}(SU)$ for $U = \mathtt{orth}(A)$ (e.g., subspace embedding, quality of a preconditioner, etc.)

Thanks to the rotation invariance of Gaussian distribution, SU is also Gaussian, so we can use the Marchenko-Pastur law:

$$\sigma_{\min}(SU) \sim 1 - \sqrt{\frac{d}{l}}, \qquad \sigma_{\max}(SU) \sim 1 + \sqrt{\frac{d}{l}}$$

Question: Can we obtain similar results with non-Gaussian sketches?

RMT analysis in RandNLA

Consider sketching matrix $S \in \mathbb{R}^{l \times n}$ with iid Gaussian entries.

- Sketch-and-precondition: Construct R^{-1} from the QR of SA

$$\mathrm{cond}(AR^{-1}) \leq 6 \quad \text{with high probability for} \quad l \geq 2d.$$

- Sketch-and-solve: $\hat{x} = \mathrm{argmin}_x \, \|S(Ax - b)\|_2^2$

$$\mathbb{E}\|A(\hat{x} - x^*)\|_2^2 = \frac{d}{l - d - 1}\|Ax^* - b\|_2^2 \quad \text{for} \quad l \geq d + 2.$$

- Low-rank approximation: Compute $Q = \texttt{orth}(AS)$

$$\mathbb{E}\|A - QQ^\top A\|_F^2 \leq \left(1 + \frac{k}{l - k - 1}\right) \cdot \|A - A_k\|_F^2 \quad \text{for} \quad l \geq k + 2.$$

These are all easy to show for iid Gaussian matrices.

Inversion bias: the key challenge [DM19, DLDM21]

Given $n \times d$ data matrix A of rank d, where $n \geq d$,

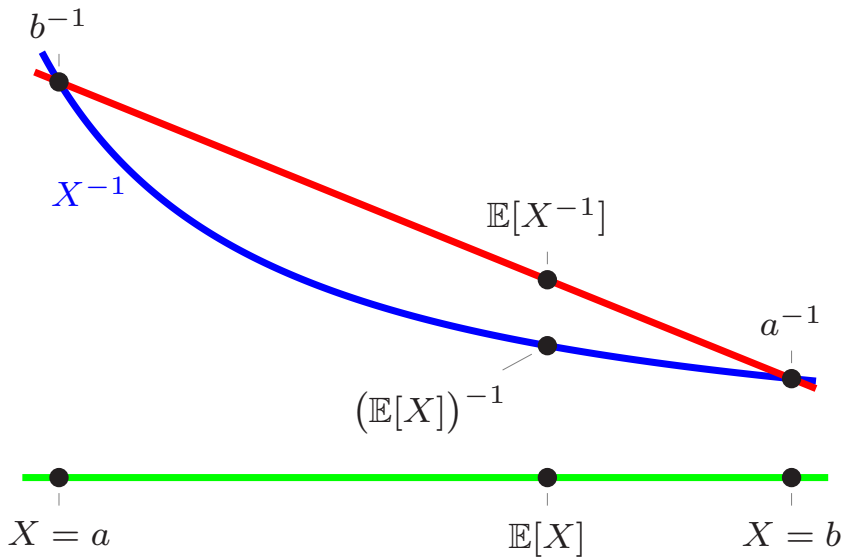
approximate $F((A^\top A)^{-1})$, where $F(\cdot)$ is a linear functional.

- $(A^\top A)^{-1}b$, for a vector b:
  - Is the OLS solution (multivariate statistical analysis, Newton's method in numerical optimization, etc.)
- $x^\top (A^\top A)^{-1}x$, for a vector x:
  - If $x = a_i$ is one of the rows of A, then it is leverage scores
  - If $x = \mathbf{e}_i$ is a standard basis vector, then this is the squared length of the confidence interval for the i-th coefficient in OLS
- $\operatorname{tr} C(A^\top A)^{-1}$ for a matrix C:
  - Used to quantify uncertainty
  - Used for experimental design criteria, e.g., A-designs and V-designs

Inversion bias: $\mathbb{E}[(\tilde{A}^\top \tilde{A})^{-1}] \neq (A^\top A)^{-1}$, even though $\mathbb{E}[\tilde{A}^\top \tilde{A}] = A^\top A$

General phenomenon: Inversion bias [DM19, DLDM21]

Inversion bias: $\mathbb{E}[X^{-1}] \neq \big(\mathbb{E}[X]\big)^{-1}$ for random X



Extends to inverting high-dimensional random matrices

In the "proportional regime"

- inversion bias is large, e.g., as large as the approximation error
- "averaging" (and other statistical/ML methods) becomes ineffective ...

Why focus on the inverse?

- Consider $S \in \mathbb{R}^{l \times n}$ having i.i.d. zero-mean rows statistically.
- $A^\top S^\top S A$ is a *sample covariance estimator* of the "population covariance matrix" $A^\top A \in \mathbb{R}^{d \times d}$.
- How does the spectrum differ between *sample* and *population* covariance?
- RMT answers this by looking at the *resolvent matrix*:

$$(A^\top S^\top S A - zI)^{-1} \quad \text{for} \quad z \in \mathbb{C} \setminus \mathbb{R}_+.$$

- The Stieltjes transform (normalized trace of the resolvent) exhibits *inversion bias*, leading to discrepancy between sample and population.
- Traditional RMT studies limiting eigenvalue distribution as $l, n, d \to \infty$.
- Our goal: *precise* and *non-asymptotic* results on resolvent matrices for sketching, e.g., $(A^\top S^\top S A)^{-1}$, leading to RMT analysis for RandNLA.

Correcting the bias (for Gaussian sketching matrices)

Consider $\hat{H} = \tilde{A}^\top \tilde{A} \approx A^\top A = H$,

(where $\tilde{A} = SA$ is an $l \times d$ sketch of an $n \times d$ matrix A)

Simple correction for a Gaussian sketching matrix S:

- Rescale by a dimensional factor: $\mathbb{E}\big[(\gamma \hat{H})^{-1}\big] = H^{-1}$ for $\gamma = \frac{l}{l-d-1}$

This is **not** true for other sketching methods. Other sketches:

- are *not perfectly rotationally symmetric*, etc.
- could *lose rank*, with very small probability
- suffer from *"coupon collector" problems*

In general, the *bias occurs differently in each direction*,

(so you cannot correct it with a single rescaling)

Q: Can we quickly correct the inversion bias, exactly or approximately?

Near-unbiasedness: an (ϵ, δ)-unbiased estimator

This motivates the following definition.

Definition

A random p.s.d. matrix $\tilde{C}$ is an (ϵ, δ)-unbiased estimator of C if there is an event $\mathcal{E}$ that holds with probability $1 - \delta$ such that

$$\mathbb{E}_{\mathcal{E}}[\tilde{C}] \approx_{1+\epsilon} C, \quad \text{and} \quad \tilde{C} \preceq O(1) \cdot C \quad \text{when conditioned on } \mathcal{E}.$$

Sub-gaussian sketches have small inversion bias

Consider a full rank $n \times d$ matrix A with $n \gg d$.

Proposition (Near-unbiasedness of sub-gaussian sketches)

Let S be an $m \times n$ random matrix such that $\sqrt{m}\,S$ has i.i.d. $O(1)$-sub-gaussian entries with mean zero and unit variance.

$$\textit{If} \qquad m \geq C\big(d + \sqrt{d}/\epsilon + \log(1/\delta)\big), \qquad \textit{then}$$

$(\frac{m}{m-d} A^\top S^\top S A)^{-1}$ is an (ϵ, δ)-unbiased estimator of $(A^\top A)^{-1}$.

So, there is an event $\mathcal{E}$ that holds with probability $1 - \mathrm{e}^{-cm}$, s.t.

$$\mathbb{E}_{\mathcal{E}}\left[(\tfrac{m}{m-d} A^\top S^\top S A)^{-1}\right] \approx_\epsilon (A^\top A)^{-1}, \quad \text{for} \quad \epsilon = O\left(\frac{\sqrt{d}}{m}\right).$$

[DLDM21]

Comparison with JL / subspace embeddings

Condition: Subspace embedding

Sketching matrix S with probability $1 - \delta$ satisfies

$$A^\top S^\top S A \approx_\eta A^\top A \qquad \text{for} \quad \eta = O(1).$$

$$\begin{aligned} \text{Subspace embedding:} \quad & \text{w.h.p.} \quad (A^\top S^\top S A)^{-1} \approx_\eta (A^\top A)^{-1} \\ \text{Near-unbiasedness:} \quad & \mathbb{E}_{\mathcal{E}}\left[(\tfrac{m}{m-d} A^\top S^\top S A)^{-1}\right] \approx_\epsilon (A^\top A)^{-1} \end{aligned}$$

For sub-gaussian sketches, we have:

$$\eta = \Theta\left(\sqrt{\frac{d}{m}}\right) \quad \text{and} \quad \epsilon = O\left(\frac{\sqrt{d}}{m}\right)$$

Subspace embedding is not enough to show near-unbiasedness!

Corollary for model averaging

Effectively, we showed that for sub-gaussian sketches:

$$\text{Bias}^2 \quad \ll \quad \text{Variance}$$

Corollary (Model averaging)

For $q = \tilde{O}(m)$ sub-gaussian sketches of size $m = O(d + \sqrt{d}/\epsilon)$,

$$\frac{1}{q}\sum_{i=1}^{q}\left(\tfrac{m}{m-d}A^\top S_i^\top S_i A\right)^{-1} \approx_\epsilon (A^\top A)^{-1}.$$

Applies to distributed averaging of linear functionals, e.g.:

$$\operatorname{tr} C\left(\tfrac{m}{m-d}A^\top S_i^\top S_i A\right)^{-1}.$$

Extending RMT-style analysis to fast sketching

- Most RMT for sketching requires:
  - different "gaussianization" assumptions
  - and different parameter regimes (e.g., proportional regime)

  compared to classical JL or subspace embedding approaches.
- Most out-of-the-box theory applies only to expensive dense Gaussian or sub-gaussian sketching matrices.
- Question: Can we extend this line of work to fast sketches, e.g., sparse or structured?
- Answer: **Yes!**

Recent developments in fast RMT-style sketches

Consider $l \times d$ sketch SA, where $S \in \mathbb{R}^{l \times n}$ and $A \in \mathbb{R}^{n \times d}$.

- Asymptotic RMT-style analysis for structured sketches [DL19, LLDP20]
  - Free probability theory via asymptotically liberating sequences applied to SRHT sketches when $l/d \to$ const and $n/d \to$ const.
- Sparse sketches close to sub-gaussian in TV distance [Der23]
  - LESS embeddings with $\tilde{O}(d)$ non-zeros per row.
  - Implies small inversion bias and non-asymptotic RMT results.
- Sparse sketches with the same spectrum as Gaussian sketches [CDDR23]
  - LESS embeddings with polylog(d) non-zeros per row
  - CountSketch/OSNAP with polylog(d) non-zeros per column
  - Implies subspace embedding for sketch size $l \geq (1+\theta)d$ with any $\theta > 0$.

Dereziński, Dobriban, et al. [DL19, LLDP20, DLDM21, Der23, CDDR23]

Analysis: Two structural conditions

Conditions for S to ensure small inversion bias:

1. Subspace embedding

   Standard approximation guarantee for sketching methods

2. Restricted Bai-Silverstein ← key novelty

   A variance bound for random quadratic forms
   Inspired by an inequality of Bai and Silverstein [BS10]
   Related to the Hanson-Wright inequality [RV13]

Analysis: Second structural condition

Consider a specific matrix $A \in \mathbb{R}^{n \times d}$, with $n \gg d$

Idea: Restrict matrices B to quadratic forms that act only on the column span of A

Condition: Restricted Bai-Silverstein

Any row vector $\mathbf{s}_i \in \mathbb{R}^n$ of the random matrix $\sqrt{m}\, S$ satisfies:

$$\mathrm{Var}[\mathbf{s}_i^\top B \mathbf{s}_i] \leq O(1) \cdot \mathrm{tr}(B^2),$$

for all $n \times n$ p.s.d. matrices B such that:

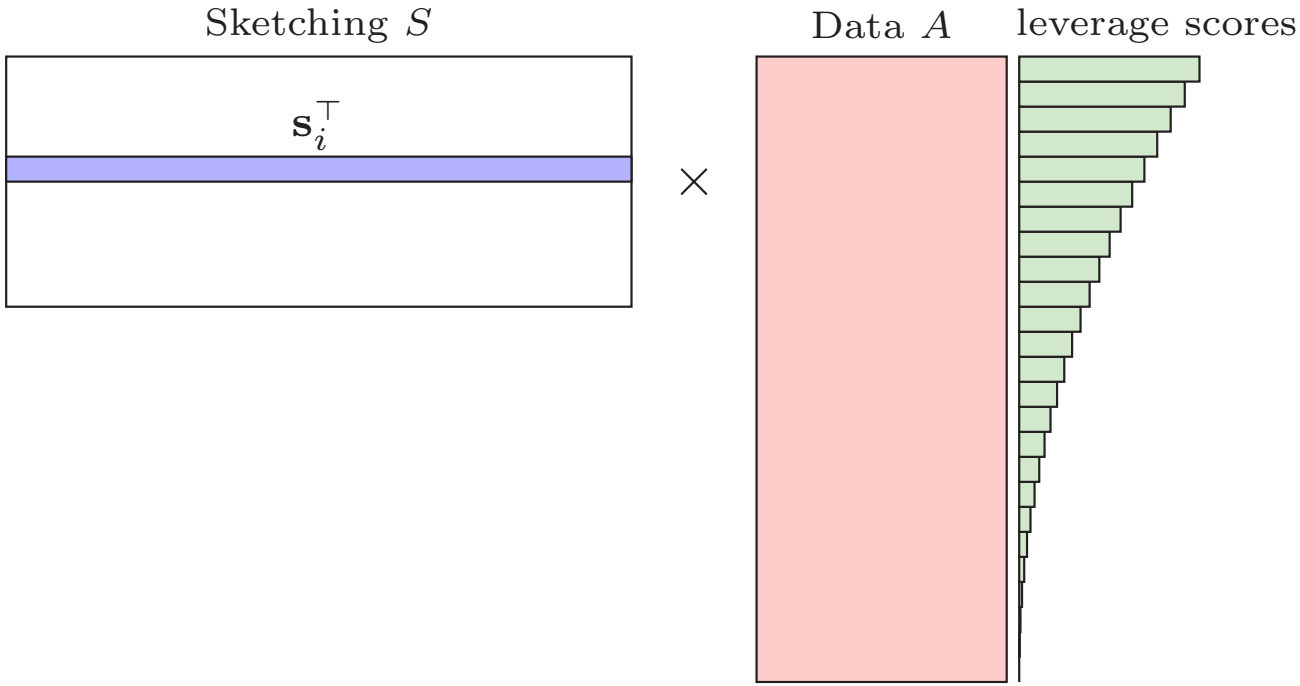
$$PBP = B, \quad \text{where} \quad P = \mathrm{proj}(\mathrm{span}(A)).$$

"Spectral analysis of large dimensional random matrices" [BS10]

Reducing the cost of sub-gaussian sketches

1. Satisfies Subspace Embedding for S
2. Satisfies Restricted Bai-Silverstein for each $\mathbf{s}_i$

Sub-gaussian Sketch

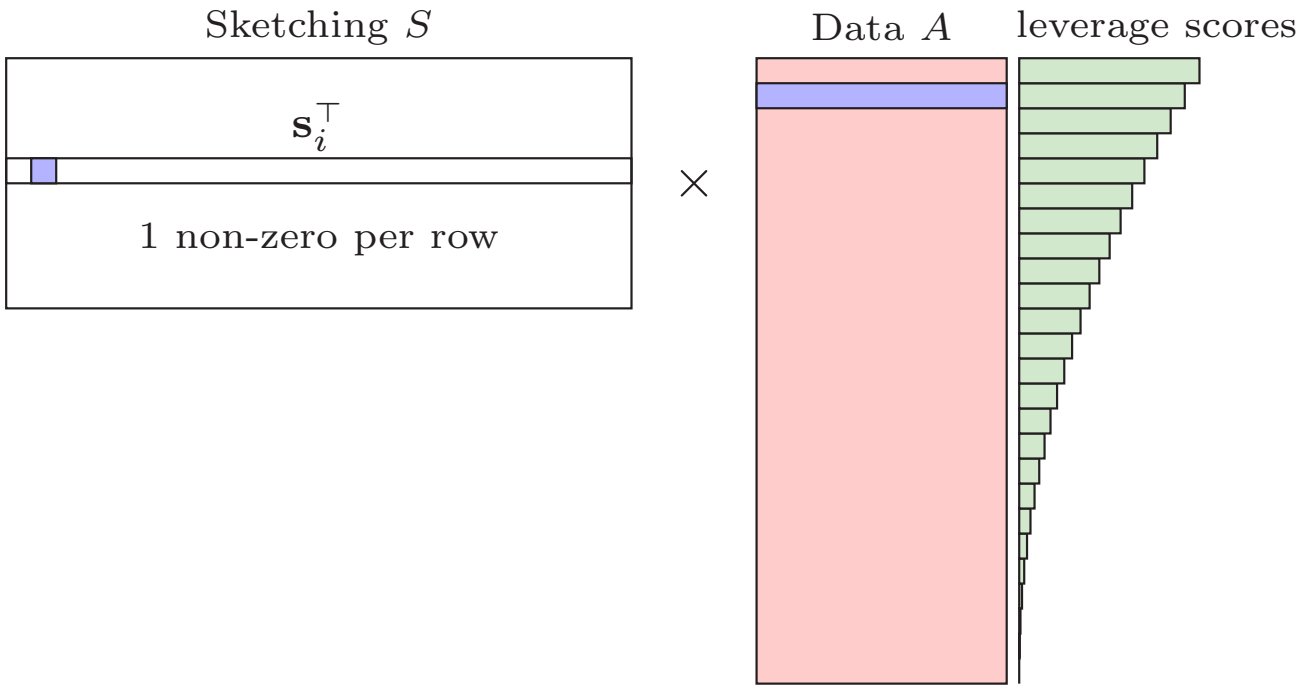


i-th leverage score $=$ i-th diagonal entry of P

Reducing the cost of sub-gaussian sketches

1. Satisfies Subspace Embedding for S ✓
2. Satisfies Restricted Bai-Silverstein for each $\mathbf{s}_i$ X

Leverage Score Sampling [DMM06]

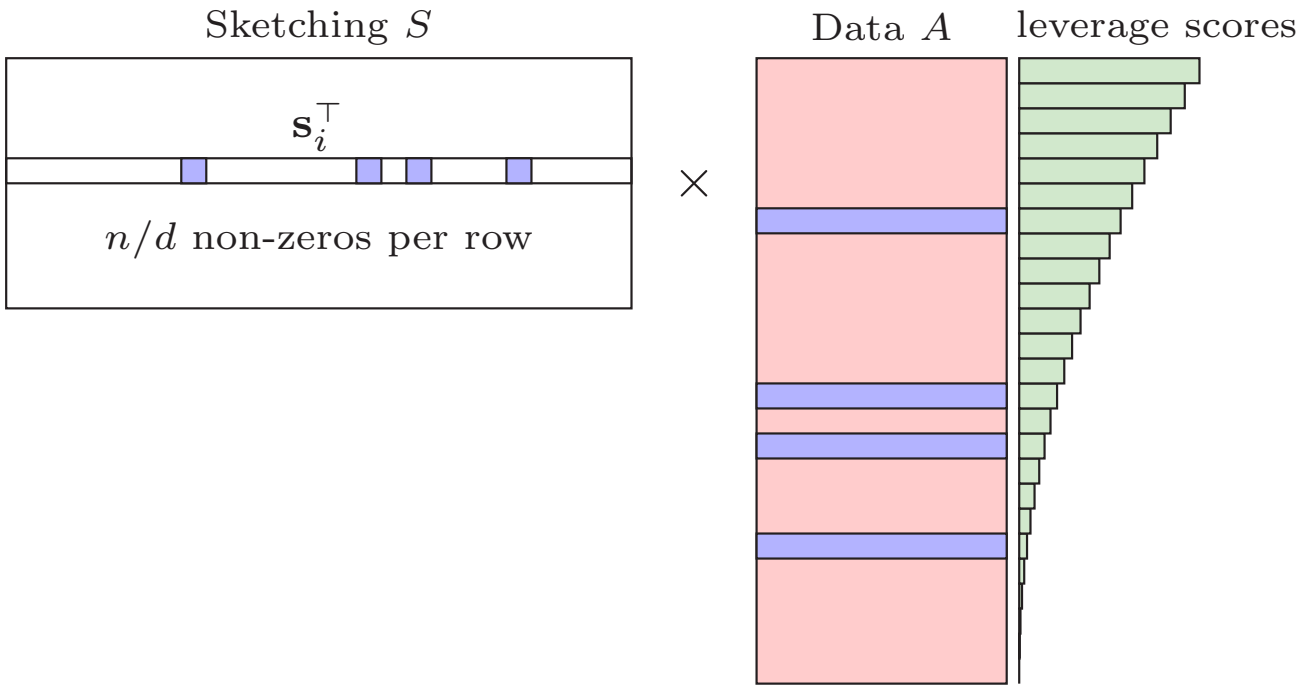


i-th leverage score = i-th diagonal entry of P

Reducing the cost of sub-gaussian sketches

1. Satisfies Subspace Embedding for S ✓
2. Satisfies Restricted Bai-Silverstein for each $\mathbf{s}_i$ ✗

Uniform Sparsification [CW13]

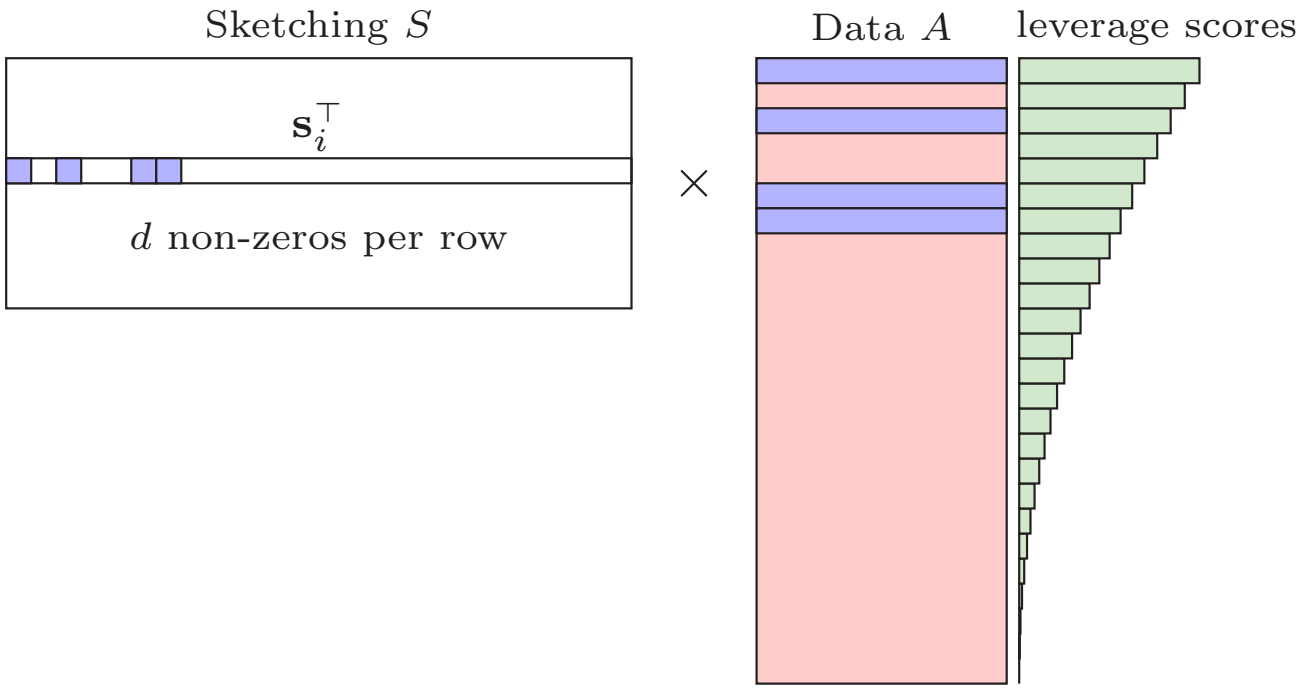


i-th leverage score $=$ i-th diagonal entry of P

Reducing the cost of sub-gaussian sketches

1. Satisfies Subspace Embedding for S
2. Satisfies Restricted Bai-Silverstein for each $\mathbf{s}_i$

Leverage Score Sparsification (LESS) [DLDM21]



i-th leverage score $=$ i-th diagonal entry of P

Correcting inversion bias for LESS embeddings

Theorem (Near-unbiasedness for LESS [DLDM21])

If S is a LESS embedding of size $m \geq C(d\log(d/\delta) + \sqrt{d}/\epsilon)$,

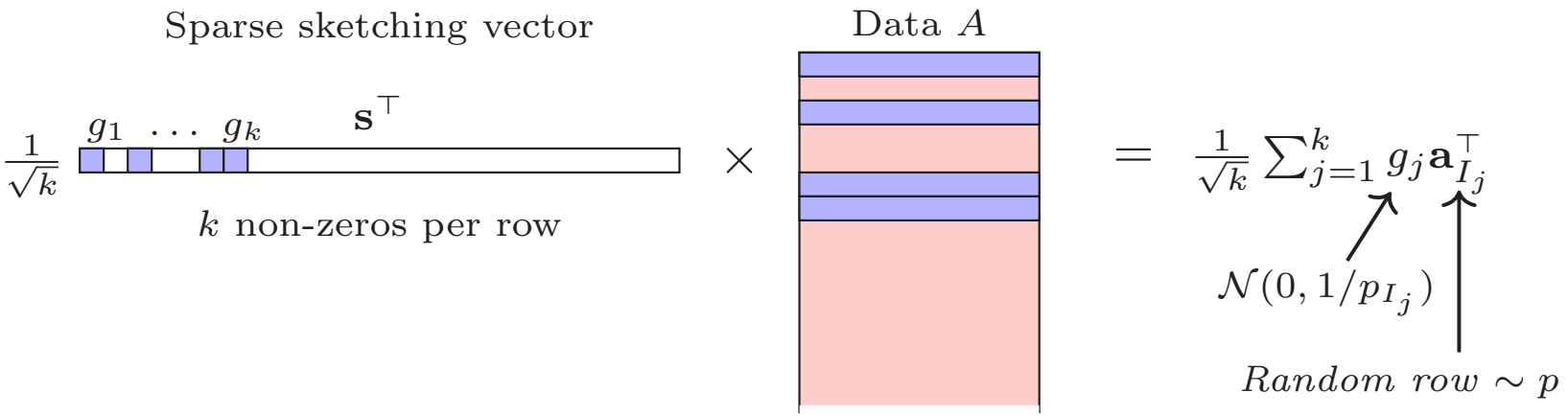
$$\mathbb{E}_{\mathcal{E}}\left[\left(\tfrac{m}{m-d}A^\top S^\top SA\right)^{-1}\right] \approx_\epsilon (A^\top A)^{-1}.$$

- Preprocessing cost: $O(\mathrm{nnz}(A)\log n + d^3\log d)$
  Approximating leverage scores [DMIMW12]
- Sketching cost: $O(md^2)$
  Sparse matrix multiplication

Note: One can show a lower bound for leverage score sampling

$\mathrm{nnz}(A)$ = number of non-zeros in matrix A.

Generally: Central Limit Theorem for Sparse Sketches



Central Limit Theorem:

$$\frac{1}{\sqrt{k}} \sum_{j=1}^{k} g_j \mathbf{a}_{I_j}^\top \quad \overset{k\to\infty}{\longrightarrow} \quad \mathcal{N}(\mathbf{0}_d, A^\top A)$$

Questions:

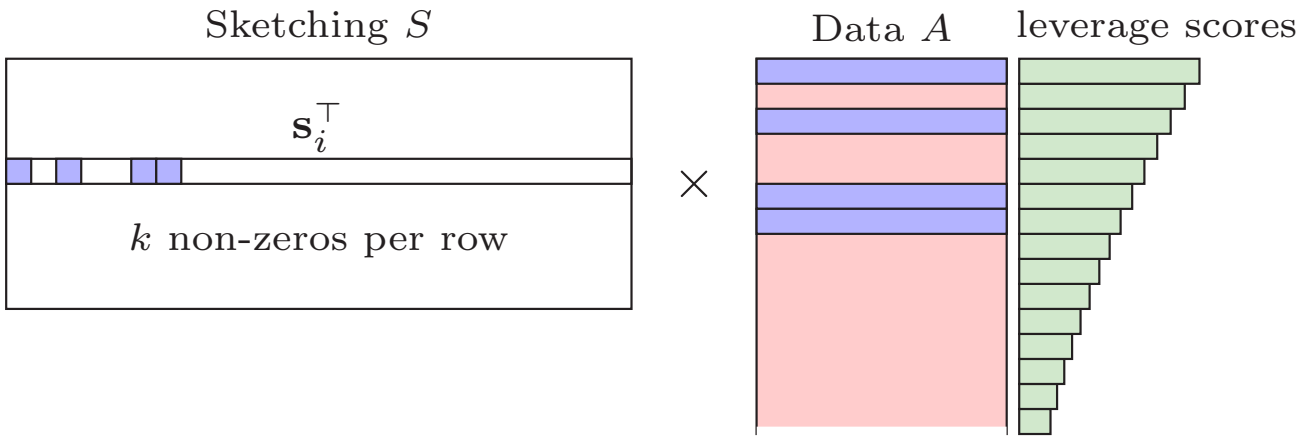
- How many samples/non-zeros do we need, and how do we select them?
- How to measure the convergence, what Gaussian concentration to use?
  e.g. Wasserstein distance, total variation (TV) distance

Generally: Central Limit Theorem for Sparse Sketches

Optimal CLT rates are when sampling non-zeros according to leverage scores:

$$i\text{-th leverage score} \quad = \quad a_i^\top (A^\top A)^{-1} a_i$$

Leverage Score Sparsification

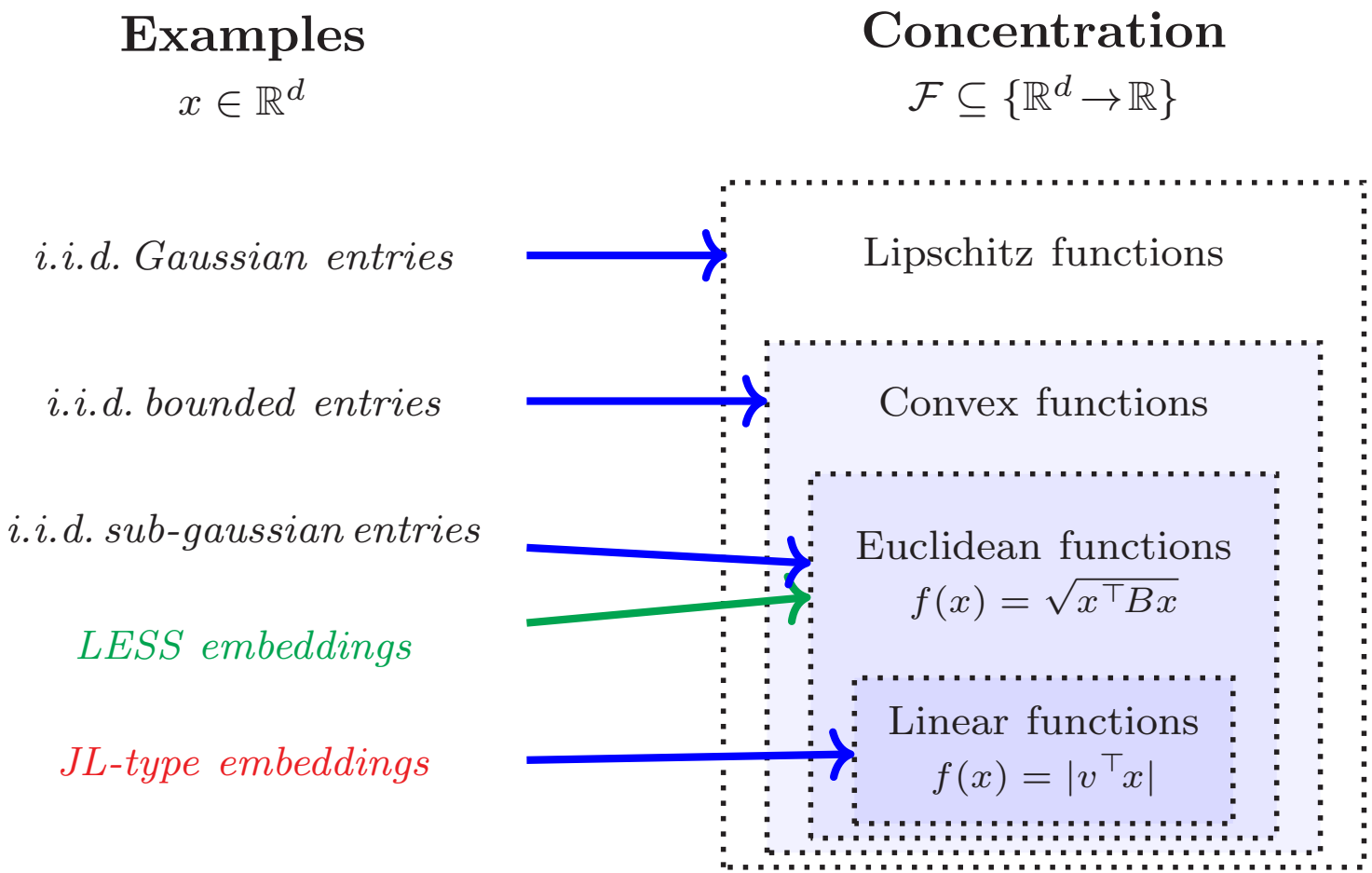


- For $k = \tilde{O}(\log d)$ nnz per row, close to Gaussian in *spectral distribution*.
- For $k = \tilde{O}(d)$ nnz per row, close to sub-gaussian in *TV distance*.

Landscape of Algorithmic Gaussianization

Sub-gaussian concentration of $x \in \mathbb{R}^d$ w.r.t. a set of functions $\mathcal{F}$

$$\forall f \in \mathcal{F}: \qquad X = f(x) - \mathbb{E}\, f(x) \quad \text{is} \quad \underbrace{O(\|f\|_{\text{Lip}})\text{-sub-gaussian}}_{\mathbb{E}\, \exp(cX^2/\|f\|_{\text{Lip}}) \leq 2}$$



Part I: Foundations of RandNLA

What about random sampling?

- RMT improves random sketching in the proportional limit.
- What about random sub-sampling (e.g., leverage scores)?
- Sampling is inherently "non-RMT"
  - involves coordinate axes / coordinate subspaces
  - lower bounds due to the Coupon Collector problem
- Answer: Determinantal Point Processes

Determinantal Point Processes (DPPs) [DM21]

A family of non-i.i.d. sampling distributions

1. Applications in RandNLA
   - Least squares regression [DW17, DWH18]
   - Low-rank approximation [DRVW06, GS12, DKM20]
   - Iterative optimization [DM19, MDK20, DY23]
2. Connections to i.i.d. sampling methods
   - Row norm scores
   - Leverage scores
   - Ridge leverage scores
3. (Theoretically) fast DPP sampling algorithms
   - Exact sampling via eigendecomposition [HKPV06, KT11, GKMV19]
   - Intermediate sampling via leverage scores [Der19, DCV19, CDV20]
   - Markov chain Monte Carlo sampling [AGR16, AD20, ALV22]

Determinantal Point Processes (DPPs) [DM21]

Given a psd $n \times n$ matrix $\mathbf{L}$, sample subset $S \subseteq \{1..n\}$:

$$\text{(L-ensemble)} \quad \text{DPP}(\mathbf{L}): \quad \Pr(S) = \frac{\det(\mathbf{L}_{S,S})}{\det(I + \mathbf{L})} \quad \text{over all subsets.}$$

closed form normalization!

$$\text{(k-DPP)} \quad k\text{-DPP}(\mathbf{L}): \quad \text{DPP}(\mathbf{L}) \text{ conditioned on } |S| = k.$$

DPPs appear everywhere!

- Physics (fermions)
- Random matrix theory (eigenvalue distribution)
- Graph theory (random spanning trees)
- Optimization (variance reduction)
- Machine learning (diverse sets)

WAIT!!! Determinants???

$$\det(A) = \prod_i \lambda_i(A)$$

Some popular wisdom about determinants:

- Expensive to compute
- Numerically unstable
- Exponentially large... or exponentially small

Down With Determinants!

Sheldon Axler

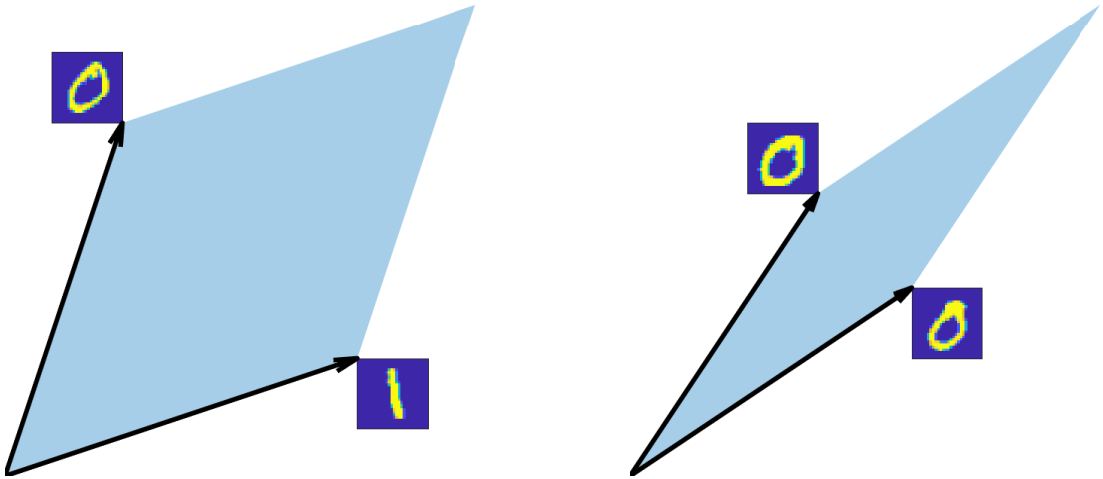
---

1. INTRODUCTION. Ask anyone why a square matrix of complex numbers has an eigenvalue, and you'll probably get the wrong answer, which goes something like this: The characteristic polynomial of the matrix—which is defined via

Volume (determinant) as a measure of diversity

Consider a subset $S \subseteq \{1, .., n\}$ of feature vectors $x_1, \ldots, x_n \in \mathbb{R}^d$

$$\underbrace{\det\left([x_i^\top x_j]_{ij \in S}\right)}_{\text{Determinant}} \quad = \quad \underbrace{\text{Vol}^2\left(\{x_i\}_{i \in S}\right)}_{\text{Volume squared}}$$

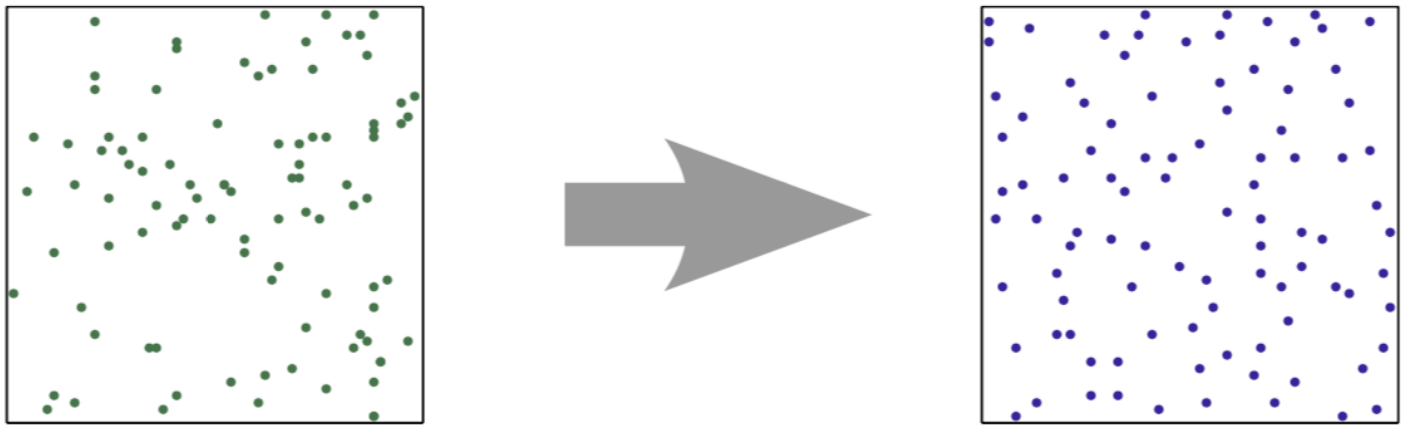


If we let $\mathbf{L} = \left[x_i^\top x_j\right]_{ij}$ then $\text{Vol}^2\left(\{x_i\}_{i \in S}\right) = \det(\mathbf{L}_{S,S})$.

Related to "volume sampling" from "classical" RandNLA [DRVW06].

Example: DPP vs i.i.d.

Negative correlation: $\Pr(i \in S \mid j \in S) < \Pr(i \in S)$



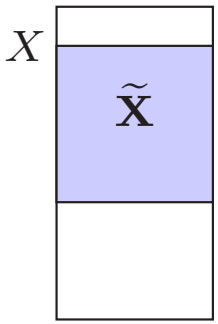
i.i.d. (left) versus DPP (right)

Image from Kulesza and Taskar "DPPs for ML" [KT12]

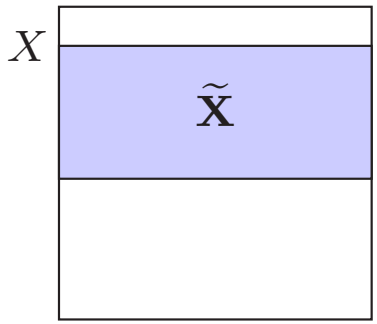
DPPs for two types of RandNLA sketches

Given: data matrix X

Goal: efficiently construct a small sketch $\widetilde{\mathbf{X}}$ (e.g., with row sampling from X)



Least-squares sketch



Low-rank approximation

i.i.d. sampling:	*Leverage scores*	*Ridge leverage scores*
DPP sampling:	*Projection DPPs*	*L-ensembles*

Connections to i.i.d. sampling

Given: full rank $n \times d$ matrix X

Methods based on i.i.d. row sampling:

1. Row norm scores: $p_i = \frac{\|x_i\|^2}{\|X\|_F^2}$

$$\frac{\|x_i\|^2}{\|X\|_F^2} = \Pr\big(i \in S\big) \quad \text{for} \quad S \sim \text{1-DPP}(XX^\top)$$

2. Leverage scores: $p_i = \frac{1}{d} x_i^\top (X^\top X)^{-1} x_i$

$$x_i^\top (X^\top X)^{-1} x_i = \Pr\big(i \in S\big) \quad \text{for} \quad S \sim d\text{-DPP}(XX^\top)$$

3. Ridge leverage scores: $p_i = \frac{1}{d_\lambda} x_i^\top (X^\top X + \lambda I)^{-1} x_i$

$$x_i^\top (X^\top X + \lambda I)^{-1} x_i = \Pr\big(i \in S\big) \quad \text{for} \quad S \sim \text{DPP}(\tfrac{1}{\lambda} XX^\top)$$

Marginals correspond to well-known RandNLA sampling methods!

Correcting inversion bias for DPP sampling

Theorem

Let A be an $n \times d$ full-rank matrix. If we sample subset $S \subseteq \{1, ..., n\}$ of size l so that $\Pr(S) \propto \det(A_S^\top A_S)$, then the row-sampled matrix A_S satisfies:

$$\mathbb{E}\big[(A_S^\top A_S)^{-1}\big] = \frac{n-d+1}{l-d+1}(A^\top A)^{-1}.$$

Consider $\hat{H} = A_S^\top A_S$ and $H = A^\top A$, Then we can correct inversion bias with:

$$\mathbb{E}\big[(\gamma \hat{H})^{-1}\big] = H^{-1} \quad \text{for} \quad \gamma = \tfrac{n-d+1}{l-d+1}$$

Recall: We could do the same for a Gaussian sketching matrix S

Conclusion: DPPs have a "Gaussianizing" effect on row sampling.
Many examples of this phenomenon, e.g., low-rank approximation [DKM20], regression [DLM20b], optimization [DBPM20].

Dereziński/Warmuth, *Unbiased estimates for linear regression via volume sampling* [DW17]

DPPs, Core RandNLA, and Implicit Regularization

	Least-squares sketch		*Low-rank approximation*	
	Projection DPP	$S \sim d\text{-DPP}_{\mathrm{L}}(XX^\top)$	L-ensemble	$S \sim \mathrm{DPP}_{\mathrm{L}}(\frac{1}{\lambda}XX^\top)$
subset size $\mathbb{E}\,\lvert S\rvert \;=$	dimension	d	effective dim.	$\mathrm{tr}(X(X^\top X+\lambda I)^{-1}X^\top)$
marginal $\Pr\{i \in S\} =$	leverage score	$x_i^\top (X^\top X)^{-1} x_i$	ridge lev. score	$x_i^\top (X^\top X+\lambda I)^{-1} x_i$
expectation $\mathbb{E}\, X_S^\dagger y_S \;=$	least squares	$\operatorname{argmin}_w \lVert Xw-y\rVert^2$	ridge regression	$\operatorname{argmin}_w \lVert Xw-y\rVert^2+\lambda\lVert w\rVert^2$

Table: Key properties of the DPPs, as they relate to: RandNLA tasks of least squares and ridge regression; RandNLA methods of leverage score sampling and ridge leverage score sampling.

An aside on Implicit Regularization

- Explicit: Replace $\min f$ with $\min f + \lambda g$:
  interpret heuristically or i.t.o. a Bayesian prior.
- Implicit 1: $\min f$ is intractable $\rightarrow$ so approximate it:
  Thm 1: $f_{approx} \approx f_{opt}$
  Thm 2: f_{approx} exactly solves $\min f + \lambda g$, for some λ, g.
  "Approximate Computation and Implicit Regularization ..." [Mah12].
- Implicit 2: Do SGD for NN training and fiddle with knobs:
  Every training knob de facto is a regularization knob.
  "Regularization for Deep Learning: A Taxonomy," [KGC17].
- Implicit Self-Regularization: The training process itself regularizes, depending on (correlated) properties of the data.
  "Implicit Self-Regularization in Deep Neural Networks ...," [MM21].
- Implicit 3: With DPPs: precise control for "noise" due to sampling/sketching; quantitatively useful for statistics, optimization, etc.
  Dereziński et al. [WGM18, DM21, DLM20b, DBPM20, DLPM21].

Part II

Recent and Upcoming Advances

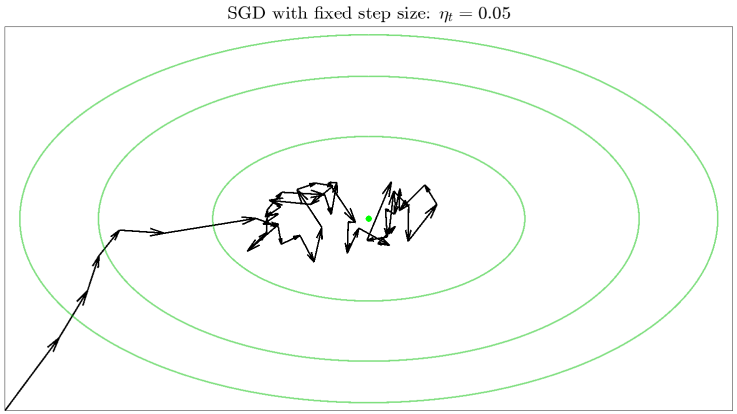
Part II: Recent and Upcoming Advances

Optimization in ML

Training ML model with parameter vector x on large dataset of size n:

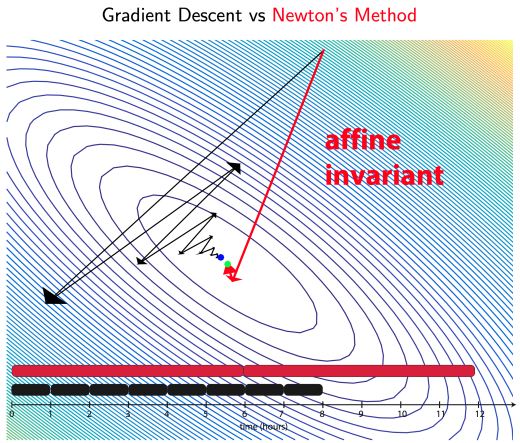
$$\text{Minimize} \qquad f(x) = \frac{1}{n} \sum_{i=1}^{n} \psi_i(x)$$

- Each $\psi_i(x)$ corresponds to the loss for the ith data point.
- Stochastic methods: Converge to the optimum with fast noisy updates by sampling the gradient (SGD, AdaGrad, SVRG, etc)
- RandNLA: helps improve convergence, stability, communication cost, and injecting curvature information into stochastic methods.



Optimization in ML

Types of optimization methods:



1. First-order: Use the gradient of f at x_t.
   - Example: Gradient descent (GD)

     $x_{t+1} = x_t - \eta_t g_t, \quad \text{where} \quad g_t := \nabla f(x_t) \in \mathbb{R}^d.$

2. Second-order: Also use the Hessian matrix
   - Example: Newton's method

     $x_{t+1} = x_t - \eta_t H_t^{-1} g_t, \quad \text{where} \quad H_t := \nabla^2 f(x_t) \in \mathbb{R}^{d \times d}.$

Iterative sketching: Instead of the exact gradient/Hessian, use a sketched/sub-sampled estimate $\hat{g}_t$ or $\hat{H}_t$.

Paradigms in iterative sketching for optimization

1. Gradient Sketch:
   - Compute a stochastic gradient estimate $\tilde{g}_t \approx g_t$ using RandNLA.
   - Application: Variance reduction and compression.
2. Hessian Sketch:
   - Compute a stochastic Hessian estimate $\tilde{H}_t \approx H_t$ using RandNLA.
   - Application: Injecting curvature information into optimization.
3. Sketch-and-Project:
   - Iteratively project onto a sketched/subsampled linear system.
   - Application: Solving the Newton system $H_t(x_t - x) = g_t$.

Part II: Recent and Upcoming Advances

Starting point: SGD

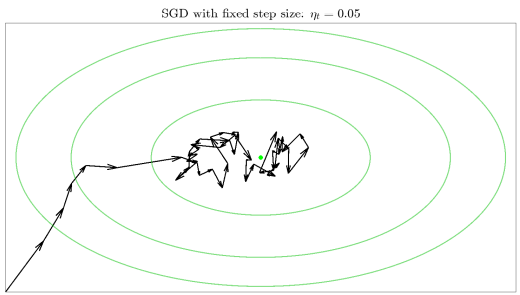
$$\text{Minimize} \qquad f(x) = \frac{1}{n}\sum_{i=1}^{n}\psi_i(x)$$

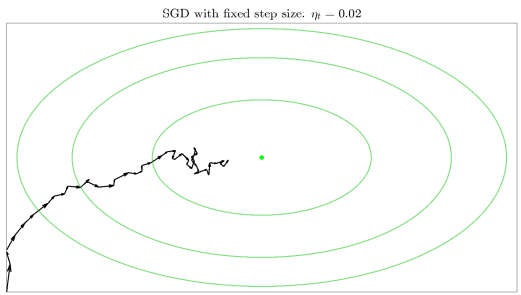
Gradient descent using a stochastic gradient estimate, e.g., $\hat{g}_t = \nabla\psi_{i_t}(x_t)$

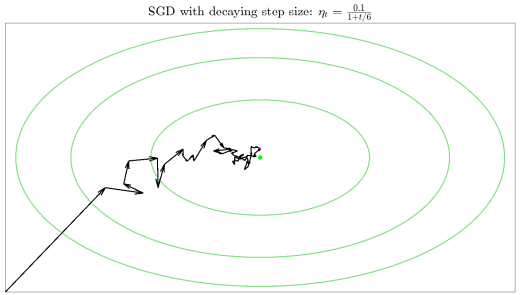
$$x_{t+1} = x_t - \eta_t\hat{g}_t, \qquad \text{where} \quad \mathbb{E}[\hat{g}_t] = \nabla f(x_t).$$

Limitations of SGD:

- Large variance $\mathbb{E}\left[\|\hat{g}_t - g_t\|^2\right]$ slows the convergence near the optimum.
- Sensitive to hyper-parameters (step size, mini-batch size, etc.)
- Lots of variants for different areas (so, more hyper-parameters to tune)







RandNLA perspective on SGD

RandNLA techniques for addressing limitations of SGD:

- Use weighted SGD with better gradient sampling (e.g., leverage scores):

$$\hat{g}_t = \frac{1}{b}\sum_{i=1}^{b}\frac{1}{w_{I_i}}\nabla\psi_{I_i}(x_t), \quad \Pr(I_i) \propto w_{I_i}.$$

- Use a sketching-based preconditioner (more on that shortly...):

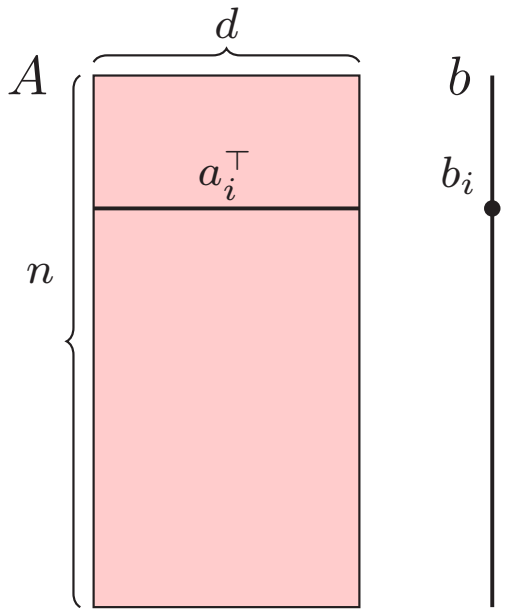
$$x_{t+1} = x_t - \eta_t M\hat{g}_t, \qquad M \approx H_t^{-1}.$$

- Or extend it to Sketch-and-Project (more on that later...)

Example: Preconditioned Weighted SGD for regression

Minimize $\quad f(x) = \frac{1}{n}\|Ax - b\|^2 = \frac{1}{n}\sum_{i=1}^{n} \psi_i(x), \qquad \psi_i(x) = (a_i^\top x - b_i)^2.$

1: Compute SA with some sketching operator S
2: Compute R such that $SA = QR^{-1}$ for orthogonal Q
3: Compute leverage score estimates $\tilde{l}_i$ for A
4: **for** $t = 0$ to $T - 1$ **do**
5: Compute
$g_t \leftarrow \frac{1}{b}\sum_{i=1}^{b} \frac{Z}{\tilde{l}_{I_i}} \nabla \psi_{I_i}(x_t), \quad \Pr(I_i) \propto \tilde{l}_{I_i}.$
6: Compute $x_{t+1} \leftarrow x_t - \eta_t RR^\top g_t$
7: **end for**



- Matrix $RR^\top$ is the preconditioner, i.e., $\approx (A^\top A)^{-1}$
- Leverage score estimates of A are the weights: $\tilde{l}_i \approx a_i^\top (A^\top A)^{-1} a_i$
- Often initialized by a Sketch-and-Solve estimate x_0.

[YCRM17]

Example: Preconditioned Weighted SGD for regression

Convergence of PW-SGD, e.g., Lemma 6.1 in [CDDR23]

Suppose that *leverage score* estimates satisfy: $\tilde{l}_i \geq l_i(A)/\alpha$ for all i. Then, letting $\eta_t := \frac{\beta}{1+\beta t/8}$ for $\beta = \frac{k/8}{k+4\alpha d}$, PW-SGD satisfies the following:

$$\mathbb{E}[f(x_t) - f(x^*)] \leq \frac{f(x_0)}{1 + bt/(c\alpha d)} \qquad \forall_{t \geq 1}.$$

- Convergence: $t = O(d/b\epsilon)$, independent of n or the condition number κ.
- Better than Sketch-and-Solve or Sketch-and-Precondition for reaching *moderate precision* in least squares.
- Extensions to l_p, Lasso, Ridge, GLMs, etc, but with different sampling (Lewis weights, ridge leverage, sensitivity scores, ...) [DLS18, ABH17]

[YCRM17, CDDR23]

Further approaches

- Gradient sketching for non-finite-sum settings:
  $g_t \in \mathbb{R}^d \to Sg_t \in \mathbb{R}^l$ for $l \ll d$, e.g., Sega [HMR18]
- Gradient compression via CountSketch for distributed/federated learning, e.g., FetchSGD [RPU$^+$20]
- Randomized Kaczmarz-inspired methods (more on that in the Sketch-and-Project section...) e.g., [NWS14]
- Randomized coordinate descent-inspired methods, e.g., [LS13]
- Randomized preconditioning for other stochastic gradient methods, e.g., Preconditioned SVRG [GOSS16], and others [GLRB18, LFY19]

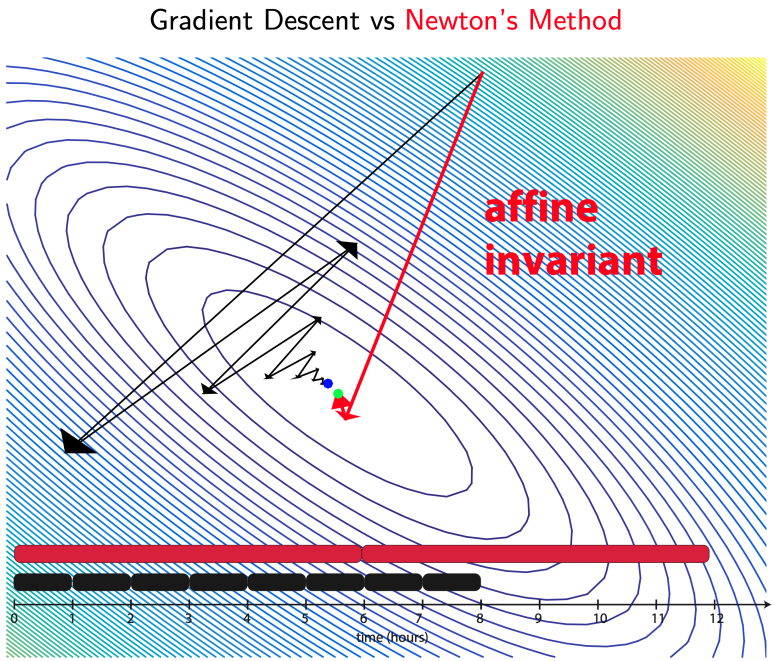
Part II: Recent and Upcoming Advances

Second-order optimization

Newton's method: Minimize a quadratic approximation of the objective using gradient g_t and Hessian H_t

$$x_{t+1} = x_t + \underset{v}{\operatorname{argmin}} \left\{ \underbrace{g_t^\top v + \frac{\eta_t}{2} v^\top H_t v}_{\text{quadratic approximation}} \right\} = x_t - \eta_t H_t^{-1} g_t$$

In other words, solve the linear system $H_t(x_t - x) = g_t$ in each iteration.



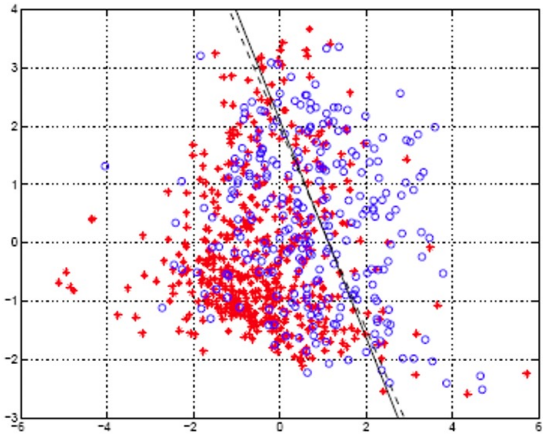
Example: Generalized linear models (GLMs)

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} l_i(a_i^\top x) + \frac{\gamma}{2}\|x\|^2$$

- Prediction $a_i^\top x$ is a linear function of a data point $a_i \in \mathbb{R}^d$.
- Loss l_i encodes prediction error, dependent on a label y_i.

Examples:

- Ridge regression: $b_i \in \mathbb{R}$, and $l_i(a_i^\top x) = \frac{1}{2}(a_i^\top x - y_i)^2$.
- Logistic regression: $b_i = \pm 1$, and $l_i(a_i^\top x) = \log(1 + e^{-y_i a_i^\top x})$.
- ...



Computing gradients and Hessians of GLMs

Computing gradients and Hessians of GLMs as matrix operations:

1. Full gradient: Matrix-vector products

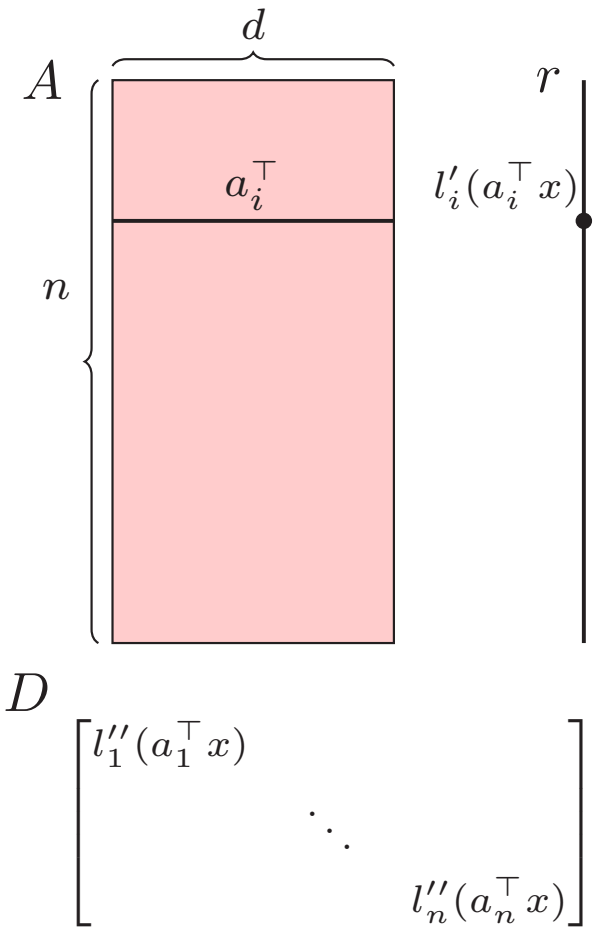
$$\nabla f(x) = \frac{1}{n} A^\top r + \gamma x$$

The cost is $O(nd)$, i.e., linear in data size.

2. Full Hessian: matrix-matrix products

$$\nabla^2 f(x) = \frac{1}{n} A^\top D A + \gamma I,$$

The cost is $O(nd^2)$, i.e., worse than linear.



Idea: Reduce the cost of computing the Hessian with sketching/sampling

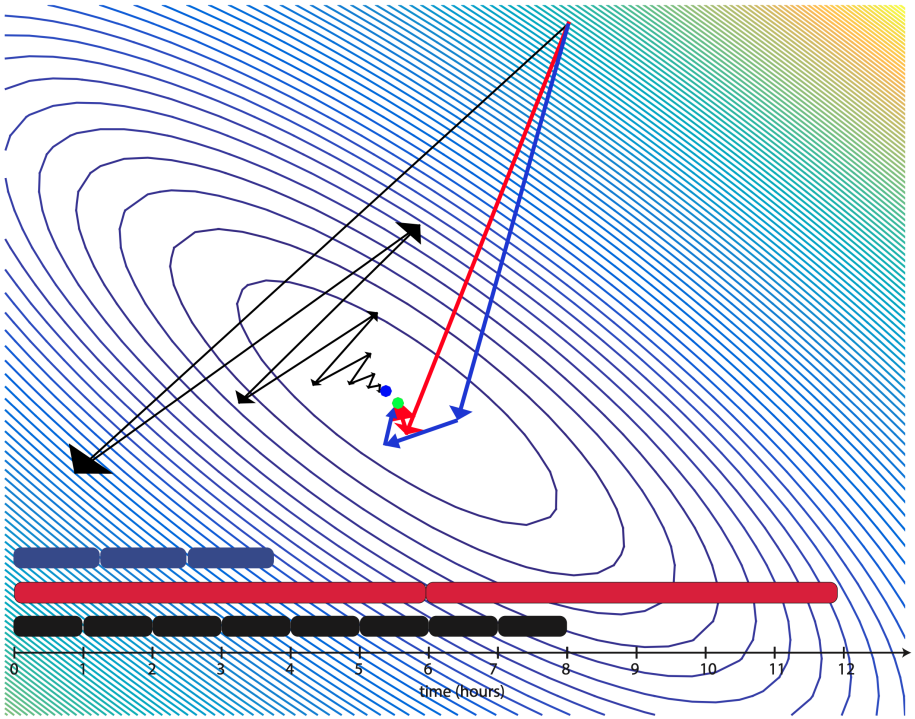
Example: Newton Sketch

For GLMs, we can sketch with $S_t \in \mathbb{R}^{l \times n}$ to approximate the Hessian:

$$\hat{H}_t = \frac{1}{n}\tilde{A}^\top \tilde{A} + \gamma I, \quad \text{where} \quad \tilde{A} = S_t D_t^{1/2} A.$$

The Newton Sketch update with step size η_t:

$$x_{t+1} = x_t + \operatorname*{argmin}_v \left\{ g_t^\top v + \frac{\eta_t}{2} v^\top \hat{H}_t v \right\} = x_t - \eta_t \hat{H}_t^{-1} g_t$$



[PW16, PW17]

Example: Newton Sketch

The Newton Sketch update with step size η_t:

$$x_{t+1} = x_t + \operatorname*{argmin}_v \left\{ g_t^\top v + \frac{\eta_t}{2} v^\top \hat{H}_t v \right\} = x_t - \eta_t \hat{H}_t^{-1} g_t$$

Theory: Ensure that $\hat{H}_t$ is a spectral approximation of H_t:

$$\frac{1}{1+\epsilon_H} H_t \preceq \hat{H}_t \preceq (1+\epsilon_H) H_t$$

E.g., when $\tilde{A}$ is a subspace embedding of $D_t^{1/2} A$, but this is often overkill.

$$\text{sketch size:} \quad l = \tilde{O}(d/\epsilon_H^2).$$

Reduces the cost of a Newton step from $O(nd^2)$ to roughly $\tilde{O}(nd + d^3/\epsilon_H^2)$.

[PW16, PW17]

Newton Sketch: Linear-quadratic convergence

- $\epsilon_H \sim \|\hat{H}_t - H_t\|$ - Hessian sketch approximation error
- $\epsilon_t \sim \|x_t - x^*\|$ - iterate optimization error

Two phases of local convergence:

1. Quadratic convergence. When $\epsilon_H \ll \epsilon_t$, then we get quadratic (Newton-like) convergence until $\epsilon_t \to \epsilon_H$:

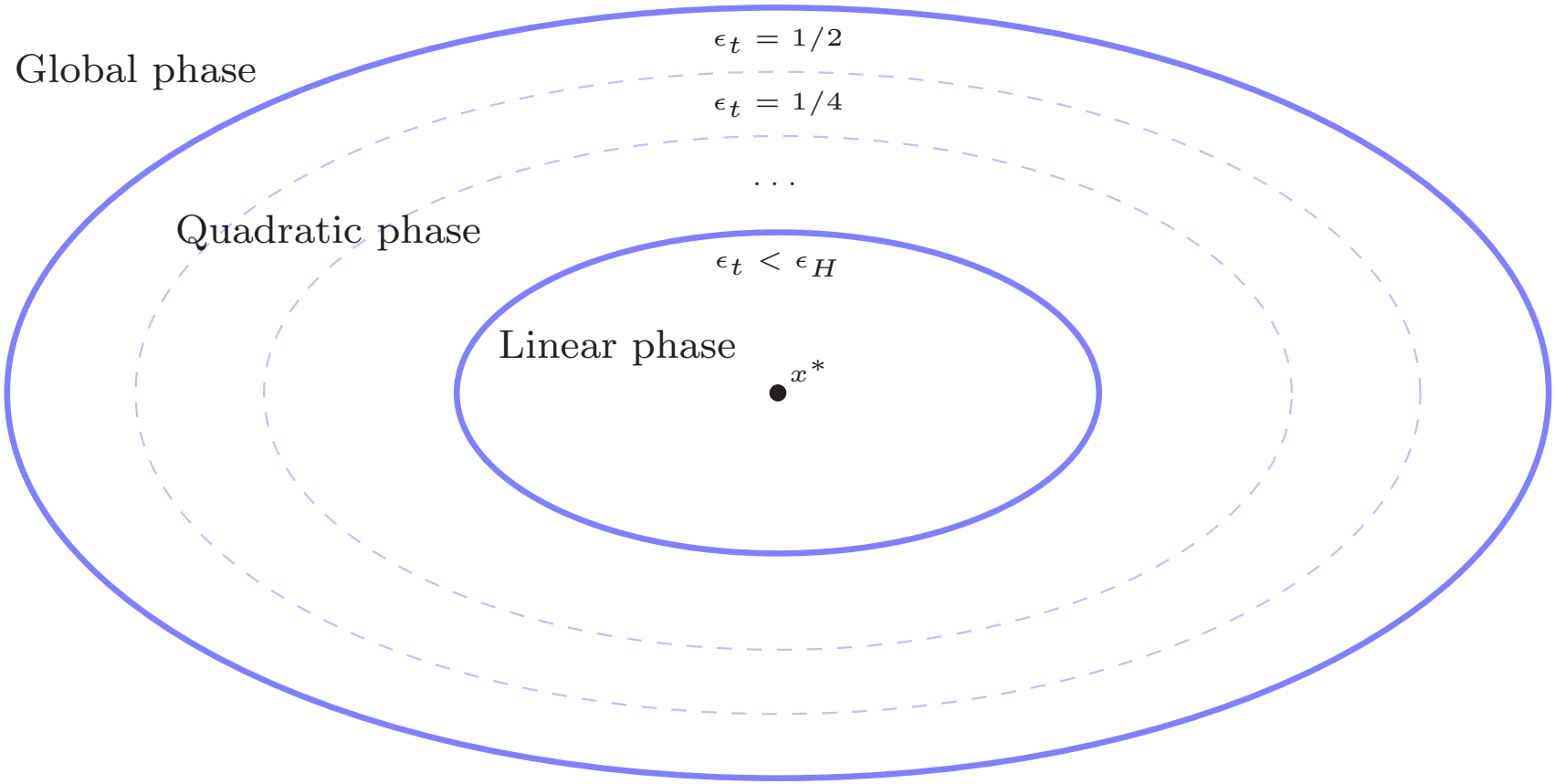
$$\|x_{t+1} - x^*\| \lesssim \|x_t - x^*\|^2,$$

2. Fast linear convergence. When $\epsilon_t \ll \epsilon_H$, then we get linear (preconditioned GD-like) convergence:

$$\|x_{t+1} - x^*\| \lesssim \underbrace{\left(1 - \frac{1}{(1+\epsilon_H)^2}\right)}_{\lesssim \epsilon_H} \|x_t - x^*\|$$

Newton Sketch: Linear-quadratic convergence

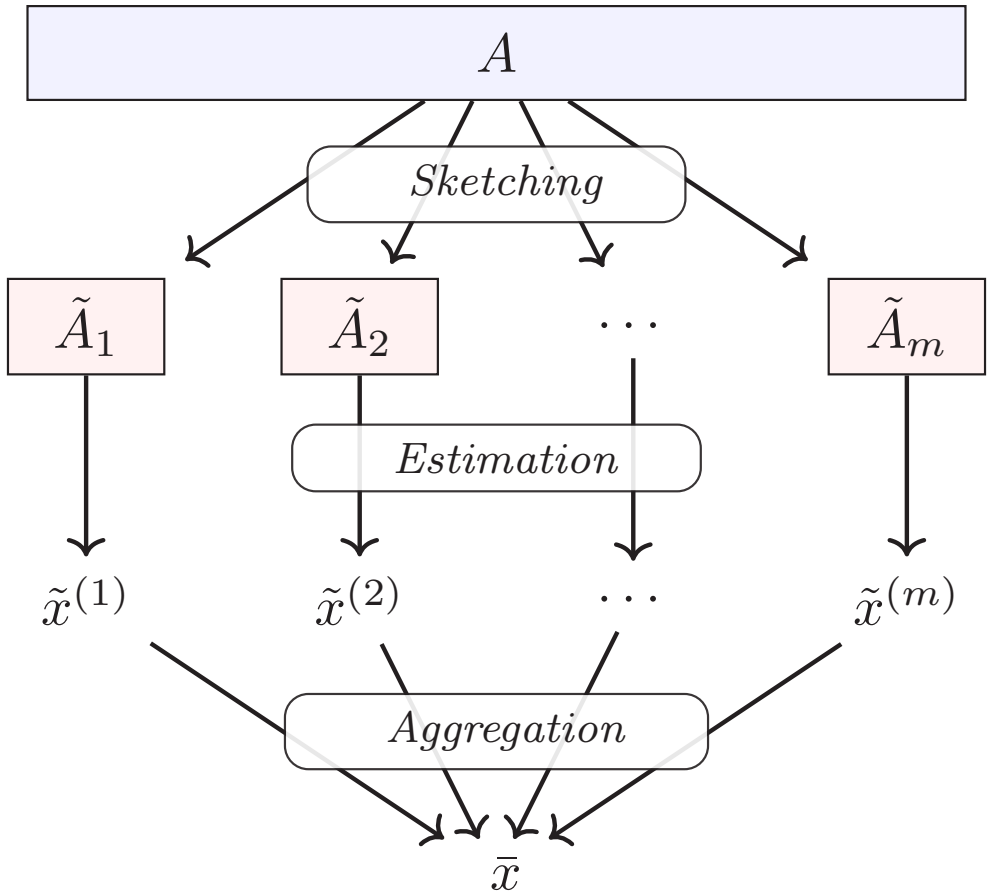
- $\epsilon_H \sim \|\hat{H}_t - H_t\|$ - Hessian sketch approximation error
- $\epsilon_t \sim \|x_t - x^*\|$ - iterate optimization error



Extending to distributed settings

1. Ensemble methods
2. Distributed optimization
3. Federated learning

$$\bar{x} = \frac{1}{m} \sum_i \tilde{x}^{(i)}$$



Example: Distributed Newton sketch

The Newton Sketch update with step size η_t and Hessian sketch $\hat{H}_t^{(i)}$:

$$x_{t+1}^{(i)} = x_t + \operatorname*{argmin}_v \left\{ g_t^\top v + \frac{\eta_t}{2} v^\top \hat{H}_t^{(i)} v \right\}$$

Distributed Newton Sketch based on m Hessian sketches $\hat{H}_t^{(1)}, ..., \hat{H}_t^{(m)}$:

$$x_{t+1} = \frac{1}{m} \sum_{i=1}^{m} x_{t+1}^{(i)}$$

- We would like x_{t+1} to get closer to exact Newton as we increase m.
- This may not occur because of *inversion bias*:

$$\mathbb{E}\big[\hat{H}_t^{-1}\big] \neq H_t^{-1}, \quad \text{even though} \quad \mathbb{E}\big[\hat{H}_t\big] = H_t.$$

- We saw how to correct this using RMT-style analysis for DPP, LESS.

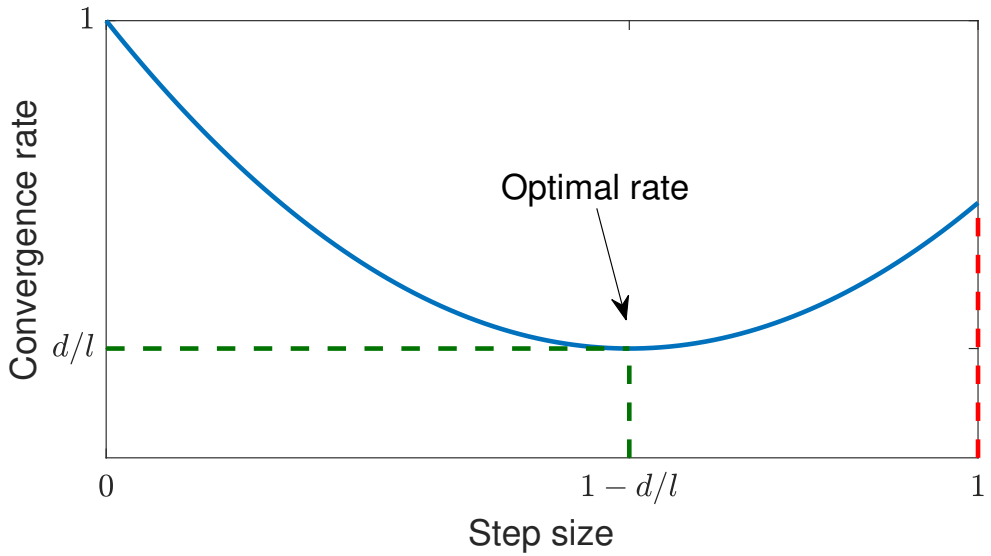
[WRXM18, DBPM20, DLPM21]

Parameter tuning in sketched Newton-type methods

Question: How to choose parameters such as sketch size l, step size η, etc?

- Could find the right step size η using line search.
- Or use RMT to derive the optimal η for a given sketch size l:

$$\mathbb{E}\,\frac{\|x_{t+1}-x^*\|^2}{\|x_t-x^*\|^2} \asymp (1-\eta)^2 + \frac{d}{l-d}\eta^2.$$



[LLDP20, DLPM21]

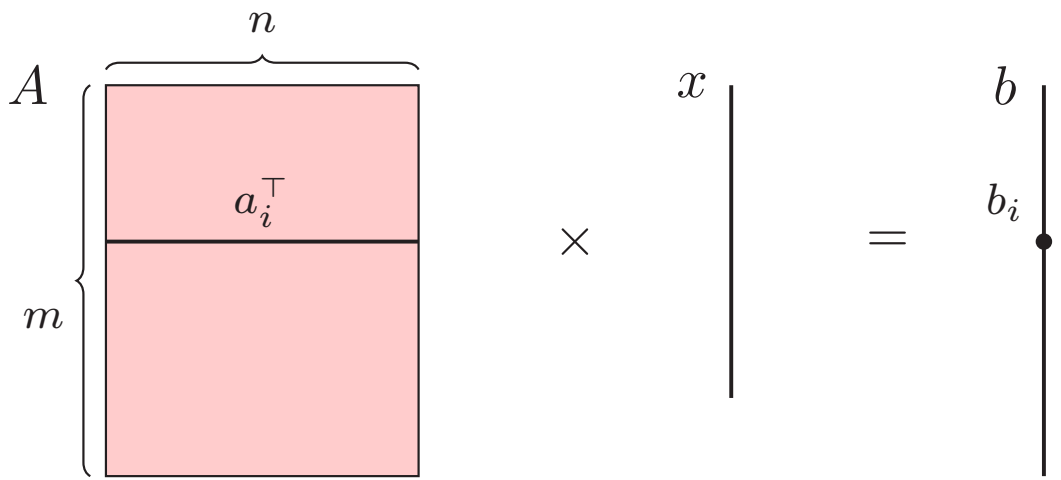
Further approaches

- Newton Sketch-type methods and RMT-style analysis, e.g., [LP22, LLDP20, NDM22, DLPM21]
- Subsampled Newton-type methods [EM15, RKM19, BBN18, BBN20]
- Subsampled Hessian approximations via Taylor expansion, e.g., LiSSA [ABH17]
- Approximating the Hessian diagonal/trace via Hutchinson's method [MMMW21] and Stochastic Lanczos Quadrature, particularly for non-convex problems, e.g., PyHessian [YGKM20], AdaHessian [YGS$^+$21]
- Stochastic Quasi-Newton type methods [KMR19, MER18].

Part II: Recent and Upcoming Advances

Key primitive: Solving linear systems

Task: Given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, find $x \in \mathbb{R}^n$ such that $Ax = b$, i.e., system of m linear equations with n unknowns.



Examples:

- Newton system $H_t(x_t - x) = g_t$ in second-order methods.
- Kernel ridge regression, e.g., $(K + \lambda I)x = y$ (more on that later).
- Many applications in scientific computing and engineering.

"Classical" RandNLA perspective

Task: Given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, find $x \in \mathbb{R}^n$ such that $Ax = b$, i.e., system of m linear equations with n unknowns.

Sketch-and-Precondition: Construct a preconditioner by sketching matrix A, and then run your favorite iterative solver (GD, CG, etc.)

- *"Least squares" approach:* Use a subspace embedding of A

  Applicable only when the system is very "tall" (over-determined) or very "wide" (under-determined). [AMT10, MSM14]

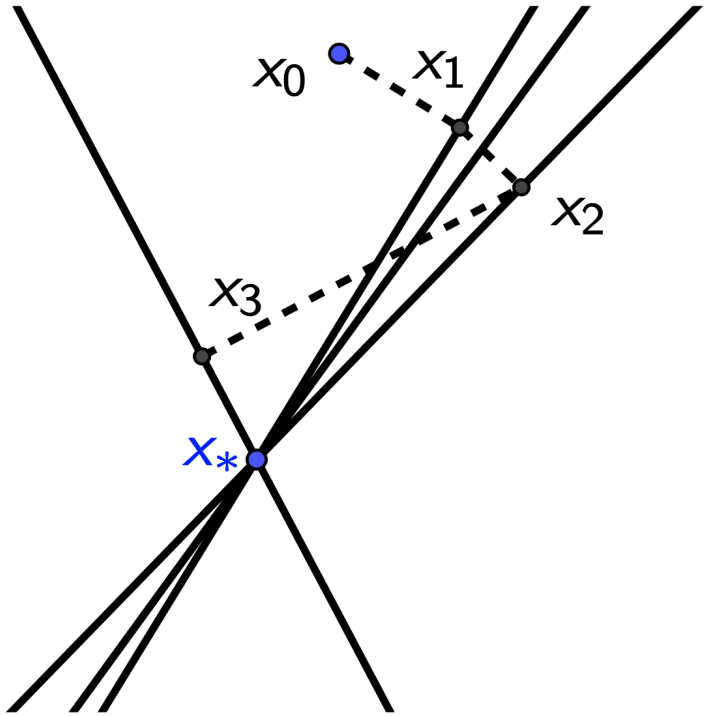
- *"Low-rank" approach:* Use a low-rank approximation of A

  Applicable for "low-rank+noise" matrices A or with regularized linear systems, i.e., where $A = K + \lambda I$. [ACW17, FTU21]

Alternative: Iteratively solve parts of the linear system at a time.

Kaczmarz algorithm

Iteratively project onto the solutions of individual equations.



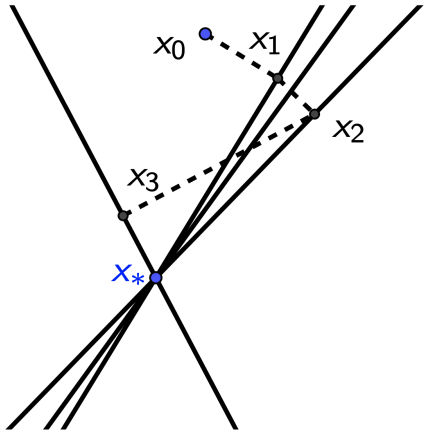
[Kac37]

Randomized Kaczmarz

Starting at $x_0 \in \mathbb{R}^n$:

1. Sample equation I_t with prob. $\propto \|a_{I_t}\|^2$
2. Project current iterate x_t onto the solutions of equation I_t:

$$x_{t+1} = x_t - \frac{a_{I_t}^\top x_t - b_{I_t}}{\|a_{I_t}\|^2} a_{I_t}$$



Theorem (Linear convergence of Randomized Kaczmarz)

If the linear system has a solution $Ax^* = b$, then:

$$\mathbb{E}\,\|x_t - x^*\|^2 \leq \left(1 - \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right)^t \|x_0 - x^*\|^2.$$

[SV09]

Randomized Kaczmarz as Weighted SGD

We can interpret Randomized Kaczmarz as solving the problem:

$$\text{Minimize} \qquad f(x) = \frac{1}{m}\sum_{i=1}^{m} \psi_i(x), \qquad \psi_i(x) = (a_i^\top x - b_i)^2.$$

Claim

Randomized Kaczmarz is Weighted SGD with importance sampling:

$$x_{t+1} = x_t - \frac{\eta_t}{w_i}\nabla\psi_i(x_t), \qquad \Pr(i) \;\propto\; w_i = \text{Lipschitz}(\nabla\psi_i).$$

- Achieves linear convergence iff x^* minimizes all components ψ_i.
- Otherwise, we get a sublinear SGD rate by decaying step sizes η_t.

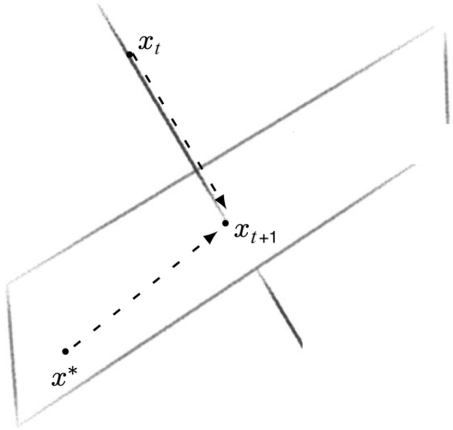
[NWS14]

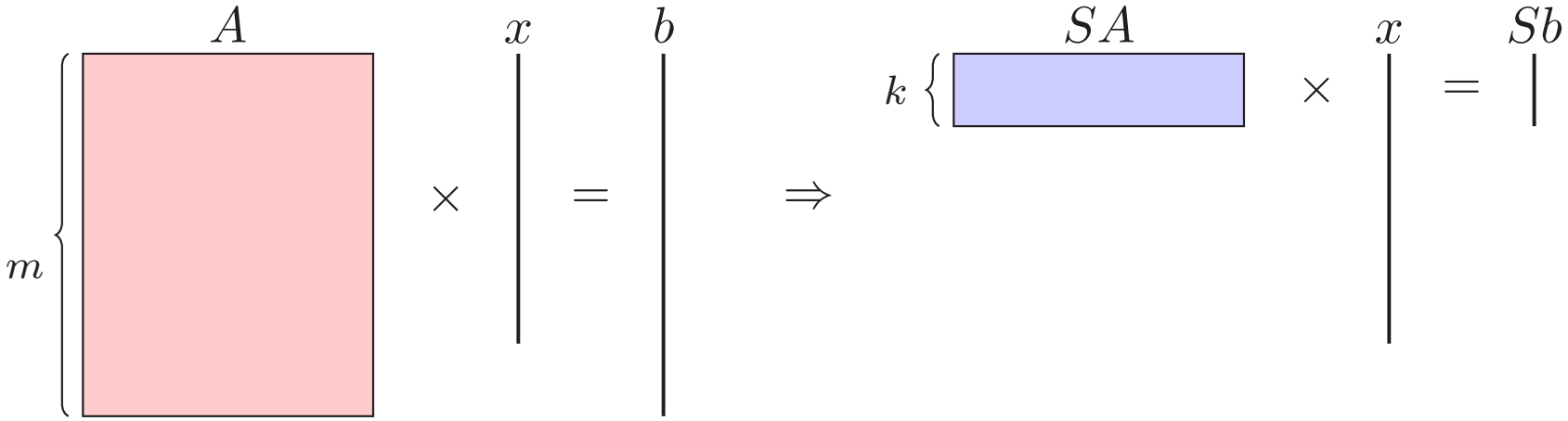
Randomized Kaczmarz as Sketch-and-Project

Sketch-and-Project:

1. Sample random $k \times m$ sketching matrix $S = S(t)$.
2. Project iterate x_t onto the solutions of $SAx = Sb$:

$$x_{t+1} = \underset{x}{\operatorname{argmin}} \|x_t - x\|^2 \quad \text{subject to} \quad SAx = Sb.$$



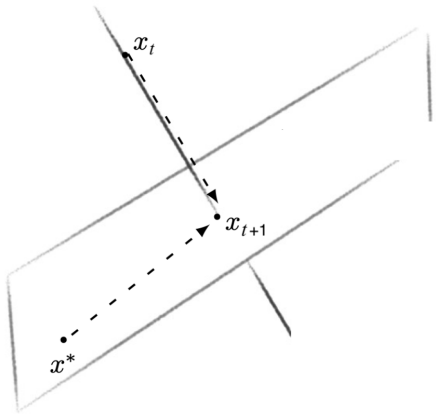


Note: Projection step is solving a "wide" linear system, so it can be done fast using the "least squares" approach of Sketch-and-Precondition.

[GR15]

Sketch-and-Project

$$x_{t+1} = \operatorname*{argmin}_x \|x_t - x\|^2$$
$$\text{subject to} \quad SAx = Sb.$$



Extends to many stochastic optimization methods:

- Block Kaczmarz, e.g., [NW13, NT14, RN21],
- Coordinate Descent, e.g., [LL10, RK20, MDK20],
- Randomized Newton, e.g., [GKLR19, KMR19, HDNR20, YLG22],
- Accelerated variants, e.g., [LS13, GHRS18, RT20]
- ...

Convergence analysis: very tricky – complex dependence on A's spectrum

RMT analysis of Sketch-and-Project

Convergence rate of Sketch-and-Project, e.g., [DR22]

For Gaussianized sketching/sampling, Sketch-and-Project satisfies:

$$\mathbb{E}\,\|x_t - x^*\|^2 \;\lesssim\; \left(1 - \frac{k\sigma_{\min}^2(A)}{\|A - \mathrm{Proj}_{SA}(A)\|_F^2}\right)^t \|x_0 - x^*\|^2.$$

- Recall convergence rate of Randomized Kaczmarz: $\left(1 - \frac{\sigma_{\min}^2}{\|A\|_F^2}\right)^t$.
- Requires RMT-style analysis and Gaussianized sketches (DPP, LESS)
- Implicit preconditioning:
  - $\|A - \mathrm{Proj}_{SA}(A)\|_F^2$ is the error of low-rank projection on the span of SA
  - Algorithm converges as if it was preconditioned with a low-rank sketch (no real preconditioning required).

[RK20, MDK20, DR22]

Making it practical: Fast Kaczmarz algorithm

Input: matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$, block size k, iterate x_0
Transform A, b using Randomized FFT/FHT; ▷ Gaussianize the data.
for $t = 0, 1, \ldots, t_{\max} - 1$ **do**
    Uniformly sample k equations $\tilde{A} = SA$, $\tilde{b} = Sb$; ▷ Sketching step.
    Solve $x_{t+1} \approx \operatorname{argmin}_x \|x_t - x\|^2$ s.t. $\tilde{A}x = \tilde{b}$; ▷ Projection step.
end for
return $\tilde{x} = x_{t_{\max}}$; ▷ Solves $Ax = b$.

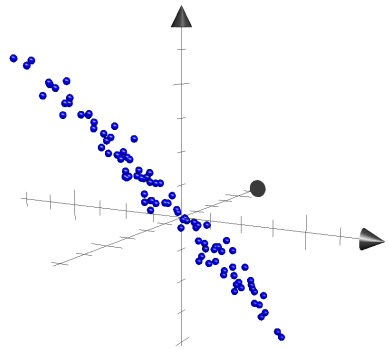
- Randomized Fourier/Hadamard transform Gaussianizes the data.
- Analysis goes via a reduction to DPP-based Sketch-and-Project.
- Recall: Projection step uses a black-box RandNLA least squares solver.

Question: How does this compare to "Classical" RandNLA-based solvers?

[DY23]

Example: Linear systems with low-rank structure

Consider $m \times n$ linear system $Ax = b$ with "rank-k + noise"



- "Classical" RandNLA: Precondition a gradient-based solver (GD, CG) with randomized rank k approximation

$$\text{Cost (via Power Iteration):} \qquad \underbrace{\tilde{O}(mnk)}_{\text{precondition}} + \underbrace{\tilde{O}(mn)}_{\text{solve}}$$

- "Modern" RandNLA: Rely on *implicit preconditioning* by using Gaussianized Sketch-and-Project with sample size k

$$\text{Cost (via Fast Kaczmarz):} \qquad \tilde{O}(mn + nk^2)$$

Various trade-offs once we look past $\tilde{O}()$ complexity, which need to be taken into account when integrating into RandLAPACK.

[DY23]

Part II: Recent and Upcoming Advances

ML perspective on RandNLA

TCS / NLA perspective:

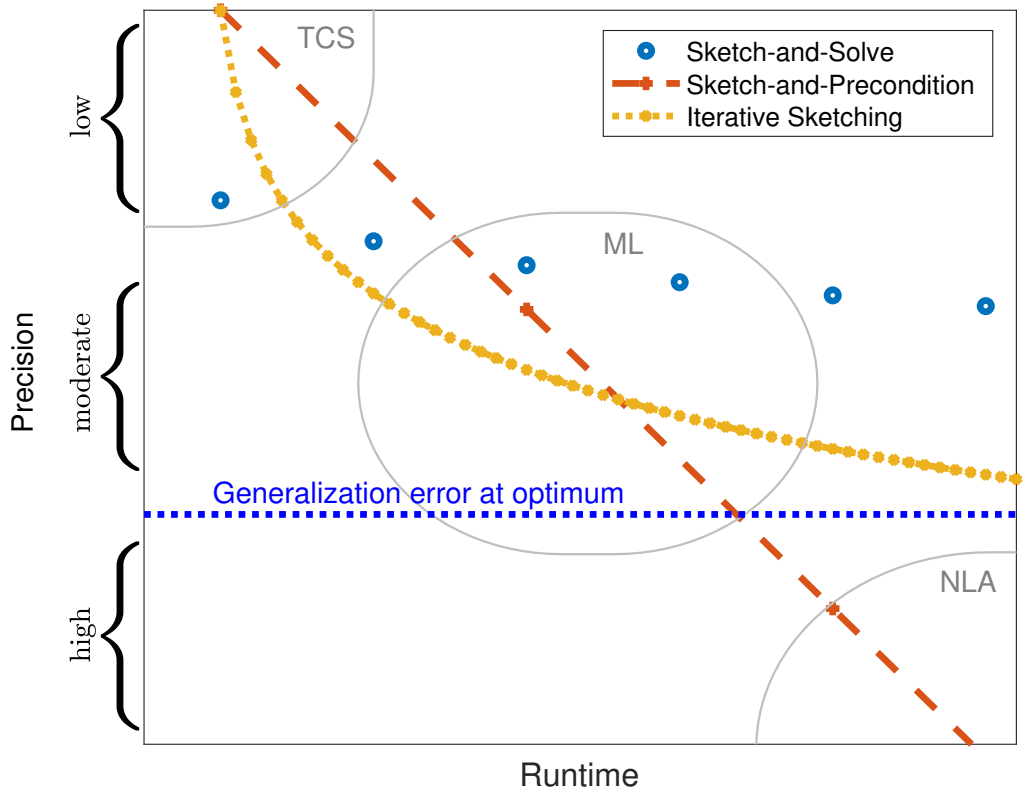
$$\text{Given } A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n \quad \text{find} \quad x_{\text{LS}} = \operatorname*{argmin}_x \sum_{i=1}^{n} |\sigma(a_i^\top x) - b_i|^2$$

- Data A and labels b are both deterministic and given.
- Goal: directly compute or estimate x_{LS} – *approximation error*

ML perspective:

- Labels are noisy, e.g., $b_i = \sigma(a_i^\top x^*) + \xi_i$ (and may be missing)
- Data may be coming from a larger population $a_i \sim \mathcal{D}$
- Goal: estimate the underlying model x^* – *generalization error*

ML perspective on RandNLA



Key question: How does RandNLA affect generalization in ML?

Advances in RandNLA for ML

1. Statistical learning approaches:
   - RandNLA sampling for robust learning
   - Kernel-based learning via low-rank approximation
2. Statistical inference approaches:
   - Incorporating statistical inference tools
   - Asymptotic bias-variance analysis
3. Random matrix theory approaches:
   - Multiple-descent in low-rank approximation
   - Implicit regularization induced by sketching

Part II: Recent and Upcoming Advances

Motivating example: Robust active learning

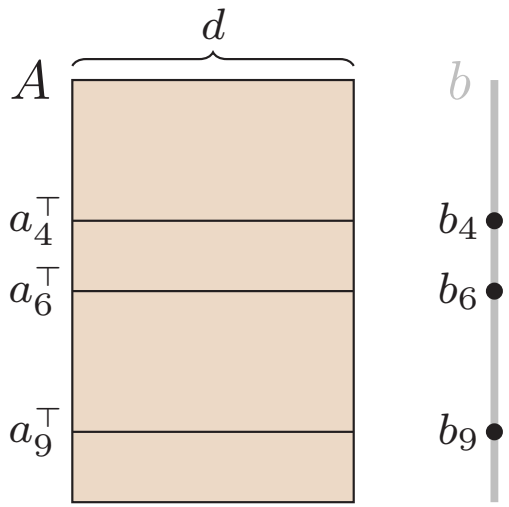
You are given a dataset with n unlabeled data points: $a_1, \ldots, a_n \in \mathbb{R}^d$.
For each point a_i, you can decide to query the random label $b_i \in \mathbb{R}$.

Goal: Query subset S of $k \ll n$ labels to output $\hat{x}$ that has small

$$\text{(Generalization Error)} \quad \mathbb{E}\,\mathrm{Err}(\hat{x}) := \mathbb{E}\,\frac{1}{n}\sum_{i=1}^{n}(a_i^\top \hat{x} - b_i)^2.$$

Select $S = \{4, 6, 9\}$

Receive b_4, b_6, b_9



Motivating example: Robust active learning

Question: How should we choose the data points to label?

Answer: It depends!

- Suppose that we know the label noise distribution, e.g.:

$$b_i = a_i^\top x^* + \xi_i, \qquad \xi_i \sim \mathcal{N}(0, \sigma^2).$$

  - This is the setting of classical experimental design. [Fed72, Puk06]
  - For each subset S we can simply compute expected generalization error $= \frac{1}{n}\mathrm{tr}(A(A_S^\top A_S)^{-1}A^\top)$, so just pick the best subset. [WYS17, AZLSW17, DLM20a]

- What if we don't know anything about the label noise?
  - This is the setting of worst-case (adversarial) active learning. [SN09, CP19]
  - Any deterministic choice of subset S could be very bad, so we must randomize.

RandNLA sampling for robust learning

Strategy: Treat this as a RandNLA Sketch-and-Solve problem where we are not allowed to look at vector b

Theorem (e.g., [Mah11])

For any fixed A and random b, if we query $O(d \log d + d/\epsilon)$ entries of b, sampled according to A's leverage scores, and solve (A_S, b_S), then:

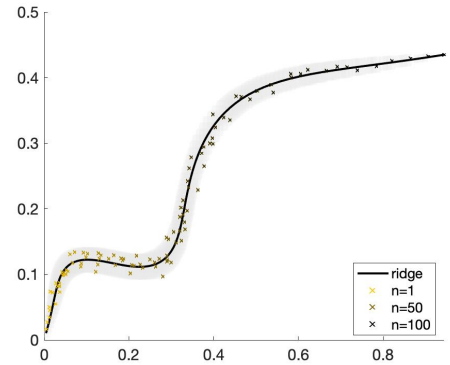
$$\mathbb{E}\ \mathrm{Err}(\hat{x}) \leq (1+\epsilon) \min_x \mathbb{E}\ \mathrm{Err}(x).$$

- Improves to $O(d/\epsilon)$ using Gaussianized RandNLA sampling (DPP, BSS). [DW17, DWH18, CP19]
- Other regression losses lead to other RandNLA sampling schemes (inverse scores, Lewis weights, ridge leverage, sensitivity scores etc.) [DCMW19, CD21, PPP21, MMWY22, CLS22]

Extensions to kernel-based learning

Kernel-based learning methods extract "non-linear" structure:

- Define features in an expanded space $\mathcal{F}$.
- Map the data space, $\mathcal{X}$, to $\mathcal{F}$ using $\phi : \mathcal{X} \to \mathcal{F}$.
- Do classification/regression with linear methods.



Basic idea:

- Use dot products for information about mutual positions.
- Define the $n \times n$ kernel matrix: $K = \left[\phi(x^{(i)}), \phi(x^{(j)})\right]_{ij} = \Phi\Phi^\top$, where Φ is the kernel representation matrix.

RandNLA + Nyström: Improves the complexity of kernel-based learning from $O(n^3)$ to $O(n)$ while retaining the same generalization guarantees.

Kernel-based learning model

Generative model: We receive a dataset $(x_i, y_i)_{i=1}^n \sim \mathcal{D}$ such that

$$y_i = f^*(x_i) + \xi_i, \qquad f^* \in \mathcal{F},$$

where $\mathcal{F}$ is a reproducing kernel Hilbert space defined by $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

Label noise ξ_i: Statistical [Bac13, AM15] or worst-case [RCR17, RCCR18]

Learning algorithm: We can construct an estimator $\tilde{f}$ for f^* by minimizing a regularized objective over the kernel space $\mathcal{F}$:

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \frac{\lambda}{2} \|f\|_{\mathcal{F}}^2$$

Example: Kernel Ridge Regression (KRR)

Using square loss and representer theorem, this problem can be cast as kernel ridge regression, with respect to the $n \times n$ kernel matrix $K = [k(x_i, x_j)]_{ij}$.

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} ([K\alpha]_i - y_i)^2 + \frac{\lambda}{2} \alpha^\top K \alpha,$$

where $[K\alpha]_i$ corresponds to our trained prediction $f(x_i)$.

This is equivalent to solving the following regularized linear system:

$$(K + n\lambda I)\alpha = y.$$

Conclusion: Ultimately, we're back to a matrix problem, which takes $O(n^3)$ time to solve exactly. What does RandNLA have to say about it?

Kernel Ridge Regression as a RandNLA problem

Problem: Given an $n \times n$ positive semidefinite (psd) matrix K and a vector $y \in \mathbb{R}^n$, solve the linear system

$$(K + n\lambda I)\alpha = y.$$

"Classical" RandNLA approaches to this problem:

- Low/Moderate precision: E.g., construct a low-rank approximation $\tilde{K}$ of K and solve the resulting smaller problem – *Sketch-and-Solve*
- High precision: E.g., construct a low-rank approximation $\tilde{K}$ and use it to precondition an iterative solver – *Sketch-and-Precondition*

Note: Either way, we need a low-rank approximation of K.

Taking into account the learning model

Recall: We receive a dataset $(x_i, y_i)_{i=1}^n \sim \mathcal{D}$ such that

$$y_i = f^*(x_i) + \xi_i, \qquad f^* \in \mathcal{F},$$

where $\mathcal{F}$ is a reproducing kernel Hilbert space defined by $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Here, the kernel matrix arises as $K_{ij} = k(x_i, x_j)$, not as a "given psd matrix".

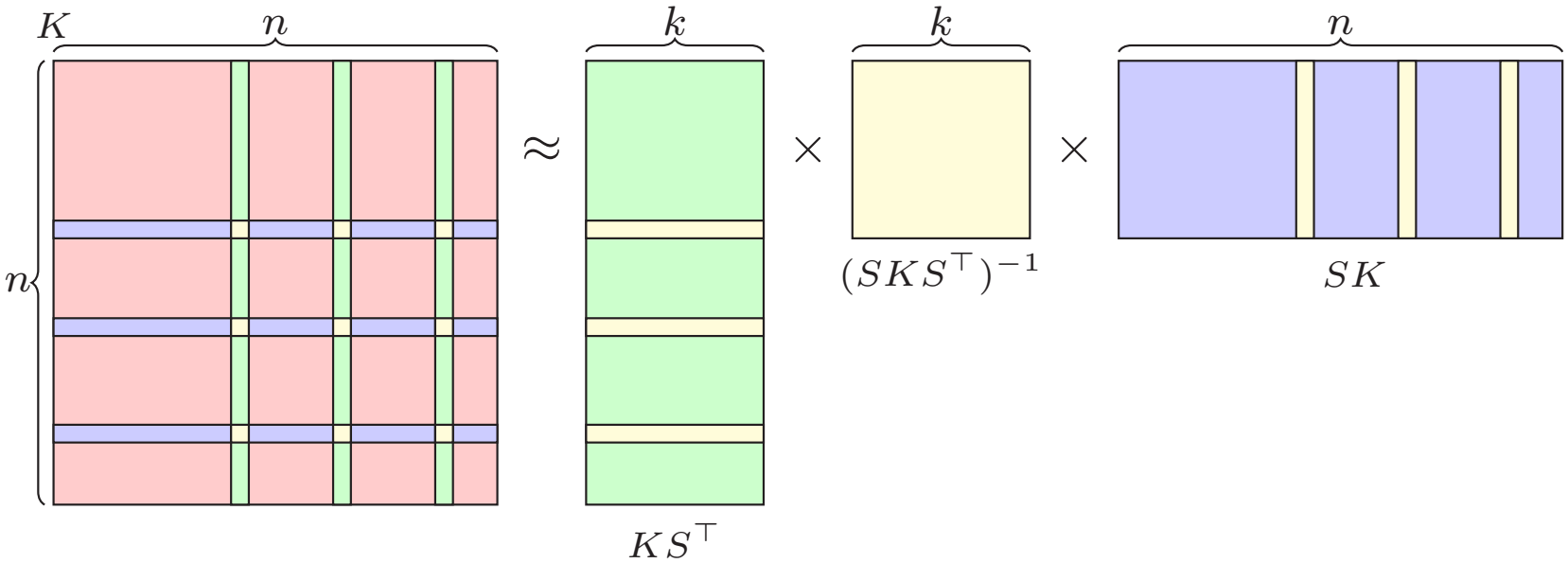
- Even an exact KRR solution only gives us an estimate $\tilde{f}$ with generalization error $\|\tilde{f} - f^*\|$.
  $\Rightarrow$ It is typically enough to seek "moderate precision".

- Even just computing the matrix K takes $O(n^2)$ time, which may already be prohibitively expensive.
  $\Rightarrow$ We need to reformulate our low-rank approximation task.

Low-rank approximation via the Nyström method

Goal: Find a low-rank approximation $\tilde{K}$ of an $n \times n$ psd matrix K, using only a subset of the entries of K.

Idea: Approximate K using a subset of rows and columns.



Let $S \in \mathbb{R}^{k \times n}$ have a single 1 in every row. Then $KS^\top$ selects k columns from K and SK selects k rows from K

$$\tilde{K} = KS^\top (SKS^\top)^{-1} SK$$

Nyström method with RandNLA sampling

$$\tilde{K} = KS^\top (SKS^\top)^{-1} SK$$

- Requires computing only nk entries of K
- Interpretation: Projection onto the k anchor points selected by S in the *kernel representation* space described by matrix Φ

$$\text{If} \quad K = \Phi\Phi^\top \quad \text{then} \quad \tilde{K} = \Phi \cdot \text{Proj}_{S\Phi} \cdot \Phi^\top$$

- How should we choose the anchors?
  - Originally: Uniform sampling, e.g., [WS01, Bac13, RCR17]
  - More robust: *ridge* leverage score sampling (as in, kernel *ridge* regression), e.g., [AM15, MM17, RCCR18]
  - RMT approach: DPP sampling to further improve the anchors, e.g., [DCV19, BRVDW19, DKM20]

Sublinear time Nyström approximation

Question: How can we construct a worst-case robust low-rank approximation of kernel matrix K, without computing K?

- Naive anwer: Uniform sampling $\Rightarrow$ not worst-case robust!
- "Classical" RandNLA: Use leverage score sampling, but to estimate the leverage scores you need to look at the whole matrix

Idea: Use the positive semidefinite (psd) structure of a kernel matrix K.

Theorem (Sublinear time leverage score sampling)

Given query access to an $n \times n$ psd matrix K, we can estimate its "rank k" ridge leverage scores using $\tilde{O}(nk)$ entries and $\tilde{O}(nk^2)$ time.

[MM17, RCCR18]

Further related advances

- Statistical learning analysis of Nyström KRR with RandNLA, e.g., [AM15, RCR17, RCCR18]
- Nyström for Gaussian Process regression, e.g., [BRVDW19, CCL+19]
- Sketching-based Nyström methods, e.g., [GM16, BBGL22]
- High-precision solvers for KRR, e.g., [ACW17, FTU21]
- Sublinear time low-rank approximation of structured matrices, e.g., [MW17, BW18]
- Sublinear time DPP sampling, e.g., [DCV19, CDV20, ALV22]

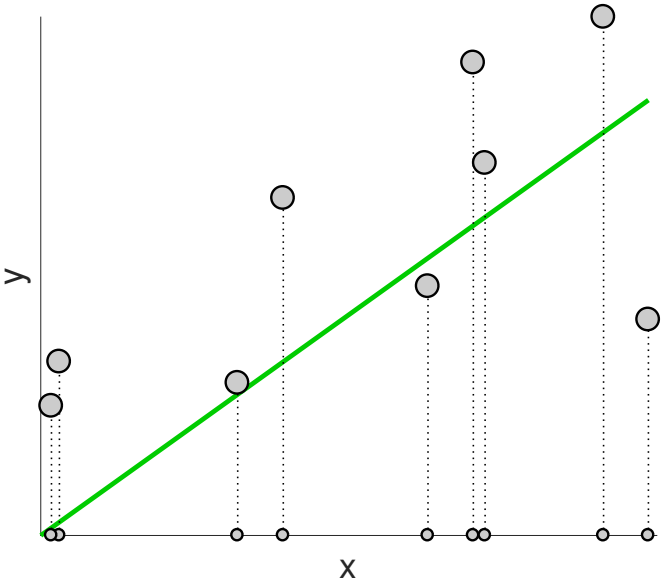
Part II: Recent and Upcoming Advances

Modeling data in statistical inference

$$z_1, \ldots, z_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0_d, \Sigma_{d\times d}), \qquad y_i = z_i^\top \beta + \xi_i, \quad \xi_i \sim \mathcal{N}(0, \sigma^2)$$

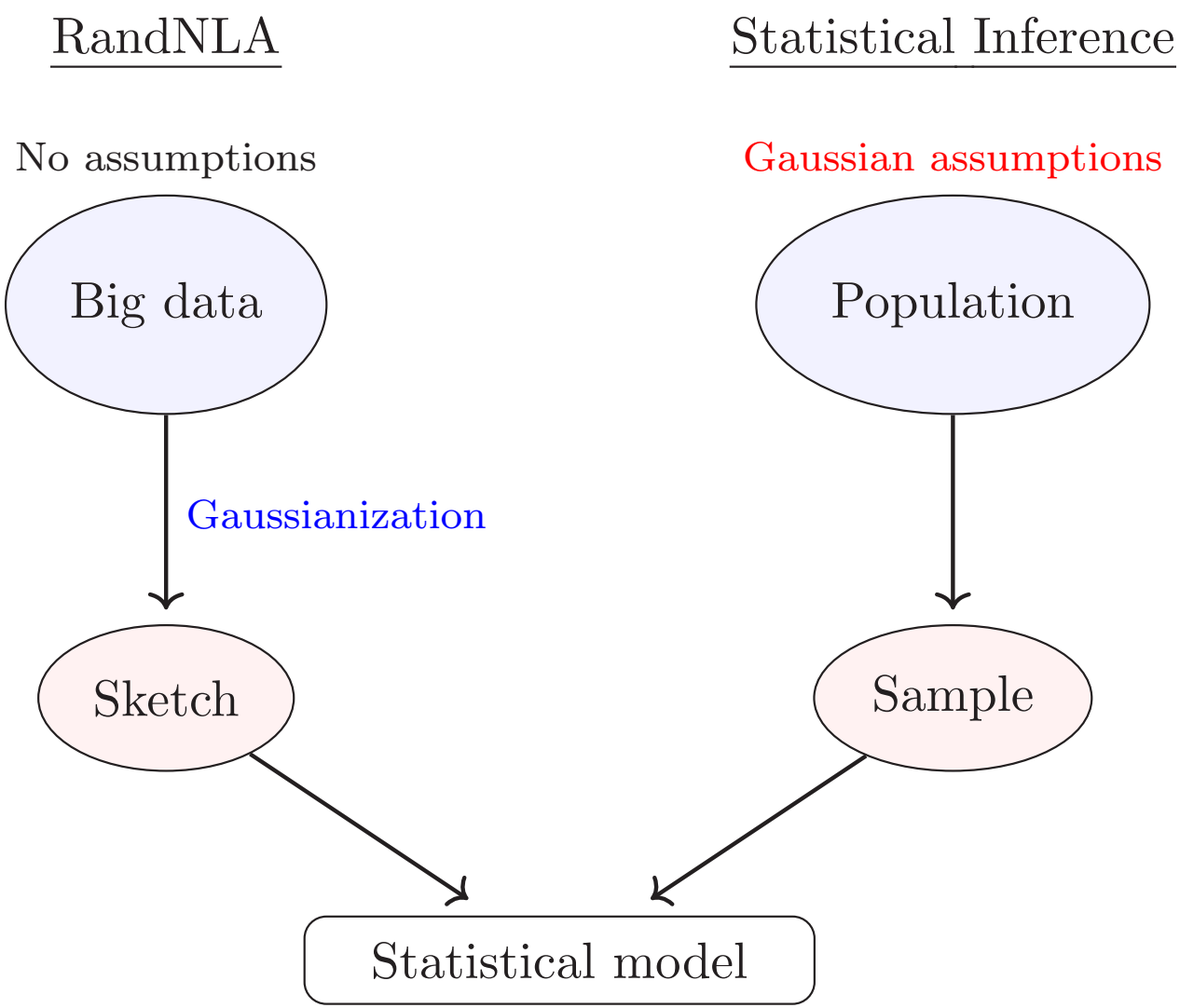
Enables a wide range of inferential tools for optimizing and evaluating estimators:

- Cross-validation provably works
- Feature selection provably works
- Confidence intervals are reliable
- Bootstrap, jackknife, ...



Note: This is completely different than in RandNLA, where we typically assume arbitrary worst-case data. And yet...

Gaussianization in RandNLA vs Statistical Inference



Incorporating statistical inference tools

RandNLA sketches follow statistical data model assumptions
$\Rightarrow$ We can use statistical inference tools on them.

- Error estimation and uncertainty quantification
  - Example: Compute a numerical estimate $\tilde{q} \approx \|\tilde{x} - x^*\|$ without knowing x^*
  - Standard approaches: Bootstrap and cross-validation

- Bagging (boostrap aggregating), i.e., averaging sketching-based estimators in distributed settings or ensemble methods.

Example: Boostrap error estimation

Consider a deterministic matrix $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, with $n \gg d$.
We reduce the problem with a random sketching matrix $S \in \mathbb{R}^{l \times n}$:
Define $\tilde{A} := SA$ and $\tilde{b} := Sb$ and compute

$$\tilde{x} := \operatorname*{argmin}_x \left\| \tilde{A}x - \tilde{b} \right\| \approx x^* := \operatorname*{argmin}_x \left\| Ax - b \right\|.$$

Goal: Efficiently compute a *numerical estimate* $\tilde{q}(\alpha)$ such that

$$\|\tilde{x} - x^*\| \leq \tilde{q}(\alpha) \qquad \text{with probability } 1 - \alpha.$$

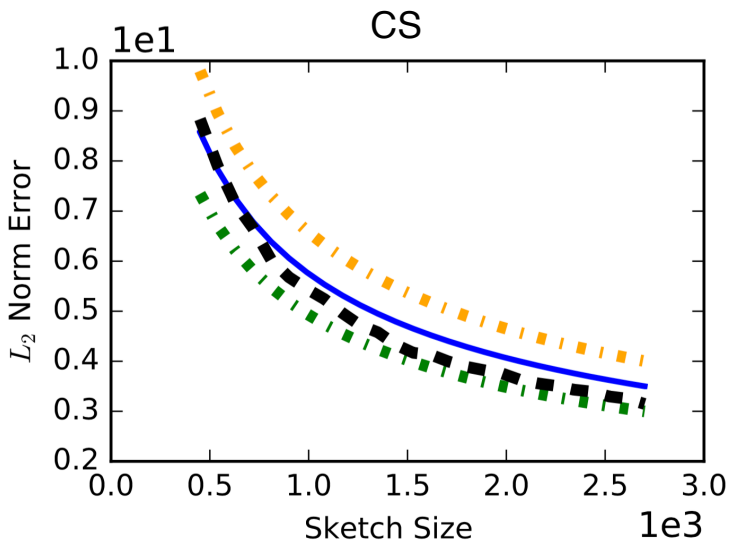
Idea: Artificially generate a bootstrapped solution $\hat{x}$ such that the fluctuations of $\hat{x} - \tilde{x}$ are similar to the fluctuations of $\tilde{x} - x^*$.

- The bootstrap sample $\hat{x}$ is the LS solution obtained by "perturbing" $\tilde{A}$ and $\tilde{b}$ (sub-sampling with replacement).
- In the "bootstrap world", $\tilde{x}$ plays the role of x^*, and $\hat{x}$ plays the role of $\tilde{x}$.

Example: Bootstrap error estimation

Theory: Guarantees are available for dense sub-gaussian sketches.

Experiment: 'YearPredictionMSD' data from LIBSVM: $n \sim 4.6 \times 10^5$, $d = 90$



[LWM18, LWM19]

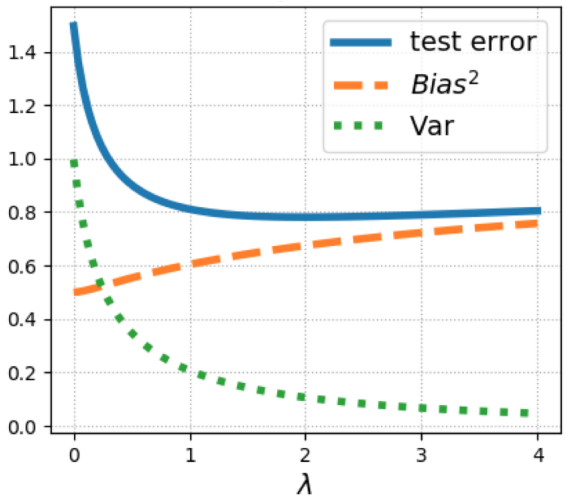
Statistical bias-variance analysis

Example: Bias-variance decomposition for a statistical estimator $\tilde{x}$:

$$\mathbb{E}\,\|\tilde{x} - x^*\|^2 = \mathrm{Var}(\tilde{x}) + \mathrm{Bias}^2(\tilde{x}).$$

Key observation: We can do this for sketching-based estimators $\tilde{x} = \tilde{x}(SA)$ because RandNLA sketches follow statistical data assumptions

- Sharp estimates for bias and variance via asymptotic RMT analysis.
- Showing separation between different sketching methods in certain parameter regimes.



Example: Separation between sketching methods

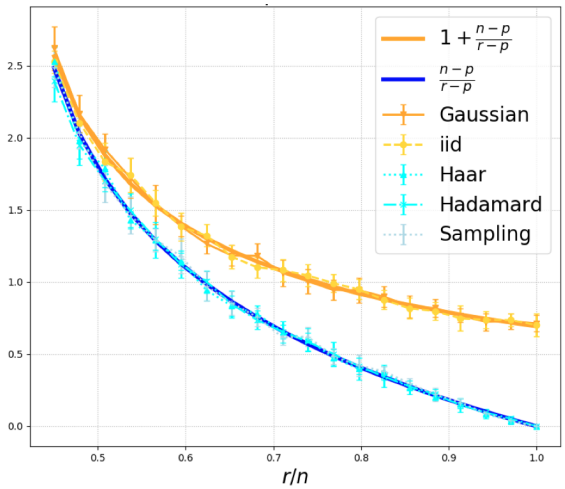
Consider the $n \times p$ least squares task $x^* = \text{argmin}_x \|Ax - b\|$

The corresponding sketched least squares estimator is

$$\tilde{x} = \min_{x \in \mathbb{R}^p} \|Ax - b\|^2 = \tilde{A}^\dagger \tilde{b},$$

where $\tilde{A} := SA$ and $\tilde{b} := Sb$ for sketching matrix $S \in \mathbb{R}^{r \times n}$.

Claim: With a fixed sketch-to-data aspect ratio r/n, asymptotic error for orthogonal (Hadamard, Haar) sketches is less than for i.i.d/Gaussian sketches.



[DL19, LLDP20]

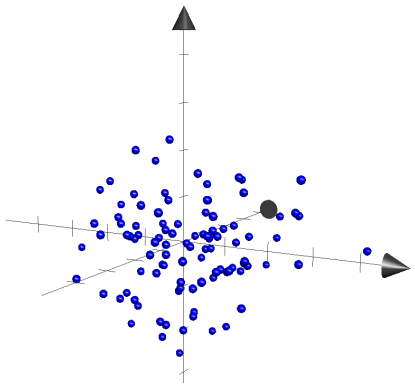
Further related advances

- Statistical perspective on RandNLA importance sampling, e.g., [MMY15, RM16, MCZ+22]
- Statistical perspective on randomized low-rank approximation [YLDW20]
- Bootstrapping sketched covariance estimation [LEM23]
- Uncertainty quantification for randomized linear system solvers [BCIH19, CIOR21]

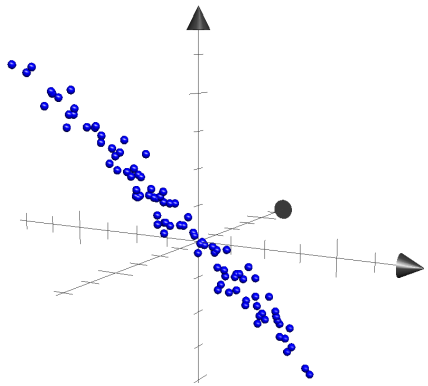
Part II: Recent and Upcoming Advances

Random matrix theory approaches

Goal: Beyond worst-case analysis of RandNLA algorithms via RMT

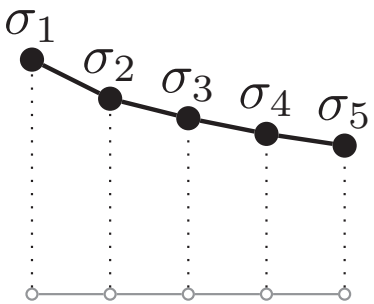


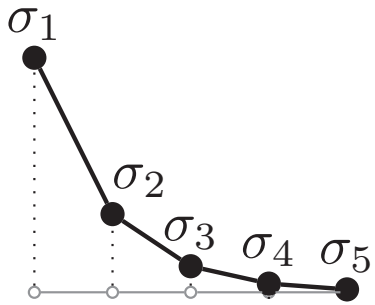
Input 1



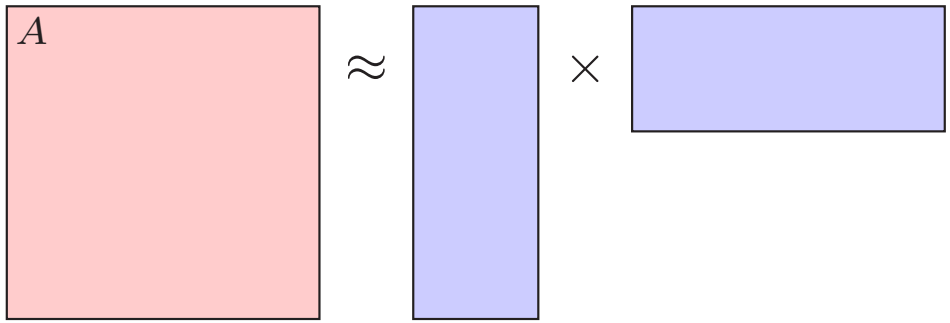
Input 2

Key idea: Performance of sketching is determined by the *spectral decay profile* (decreasing singular values of the data matrix)





Example: Low-rank approximation



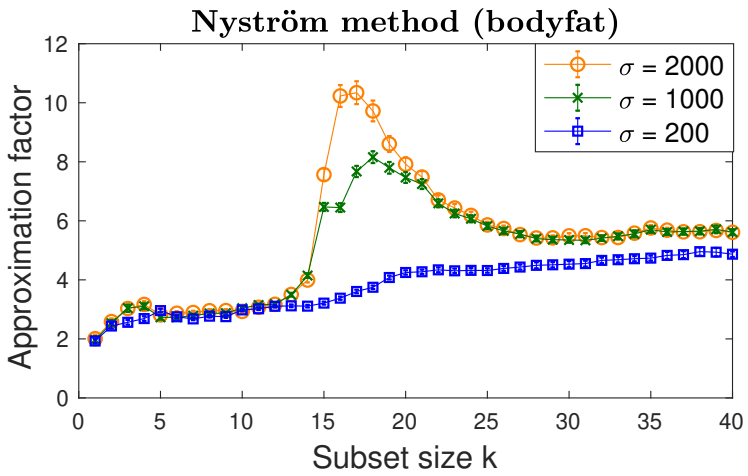
Optimal choice: Top k Truncated SVD of matrix A

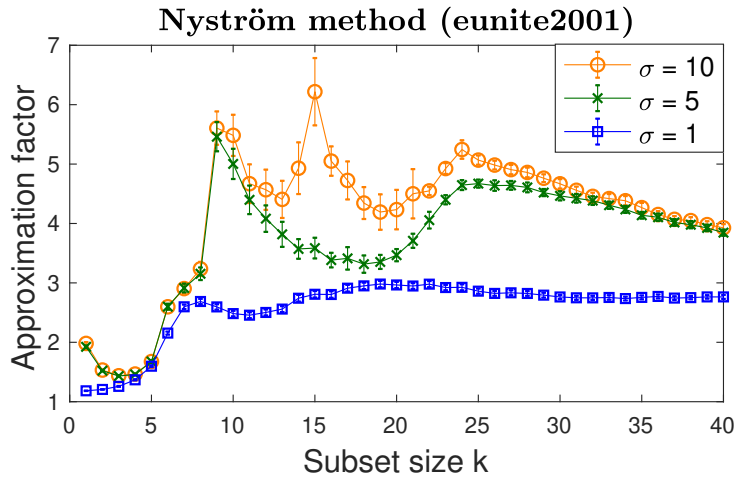
$$\text{Sketching approximation factor} \quad = \quad \frac{\text{Er}(\text{rank } k \text{ sketch } SA)}{\text{Er}(\text{top } k \text{ SVD of } A)},$$

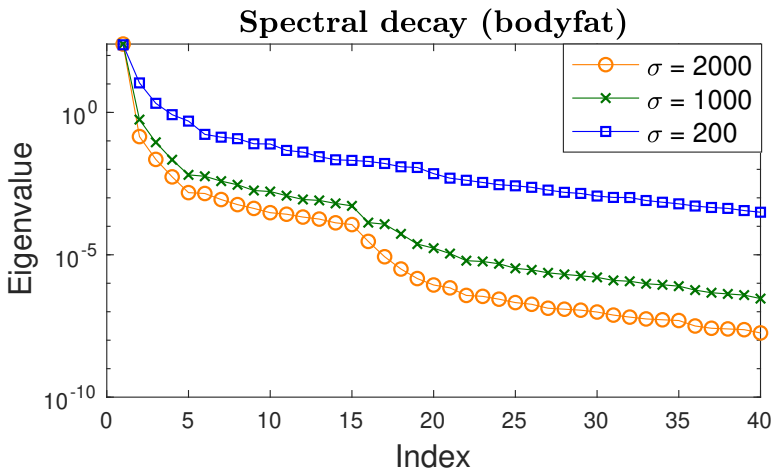
where $\text{Er}(SA) = \|A - \text{Proj}_{SA}(A)\|_F^2$ is the low-rank projection error.

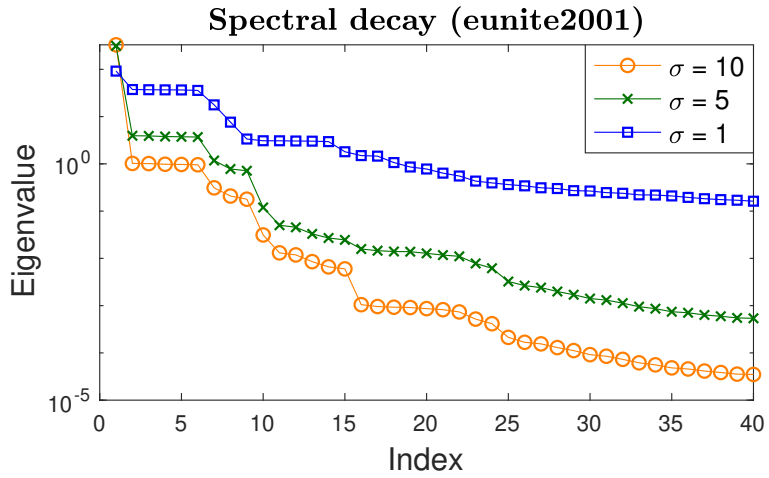
Example: Low-rank approximation

Experiment: LIBSVM data with Gaussian RBF kernel parameterized by σ





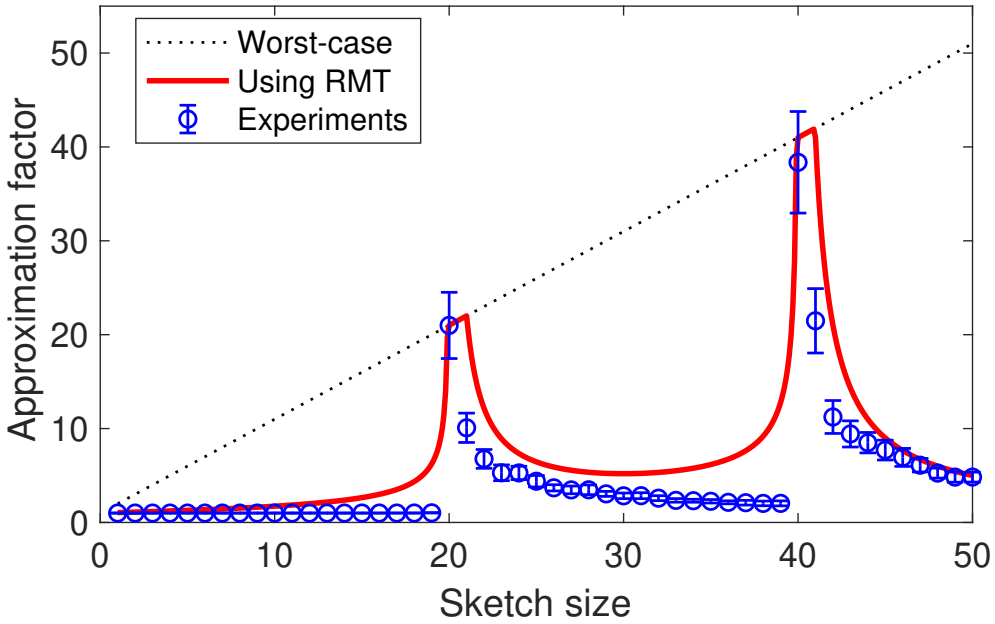




[DKM20]

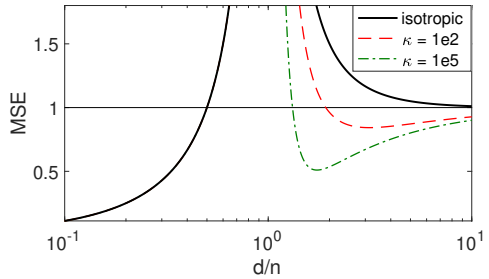
Multiple-descent in low-rank approximation

Theory: Characterizing the approximation factor using RMT [DKM20]



Connection: double descent in over-parameterized ML models [DLM20b]

"Classical" ML:	$parameters \ll data$
"Modern" ML:	$parameters \gg data$
Phase transition:	$parameters \sim data$



Example: Implicit regularization in sketching

Sketched least squares estimator: $\tilde{x} = \min_{x \in \mathbb{R}^d} \|SAx - Sb\|^2 = \tilde{A}^\dagger \tilde{b} \in \mathbb{R}^d$

"Classical" RandNLA: Guarantees for sketch sizes $l \geq \tilde{O}(d)$.

Question: What happens when the sketch size is less than the dimension ($l < d$, over-parameterized setting)?

Answer: In the over-parameterized setting, the estimator $\tilde{x}$ becomes biased (implicit regularization).

RMT analysis shows that this bias is precisely implicit ℓ_2 regularization

$$\mathbb{E}[\tilde{x}] \asymp \min \|Ax - b\|^2 + \gamma \|x\|^2.$$

Connection: implicit regularization in training NNs with SGD [Ney17, MM18]

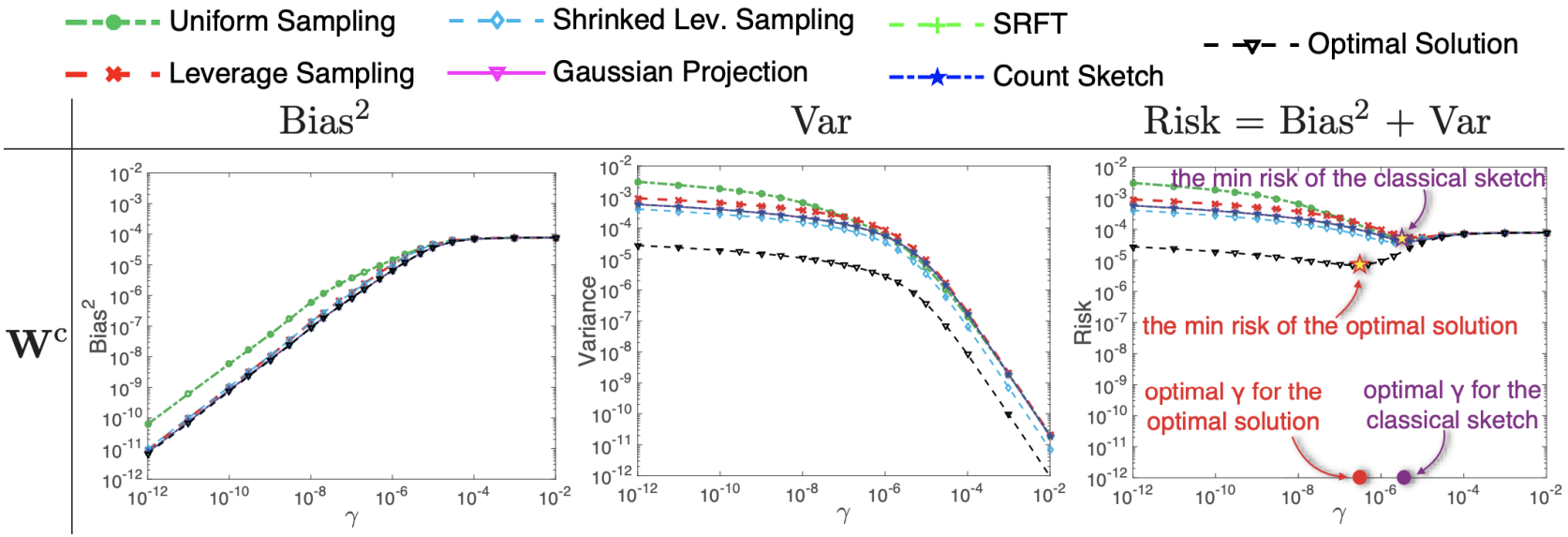
[FSS20, LJB20, DLM20b, DBPM20, LPJ+22]

Finding optimal regularization parameters

Consider Sketch-and-Solve for ridge regression with data A and labels b:

$$x^* = \min_x \frac{1}{n} \sum_{i=1}^{n} (a_i^\top x - b_i)^2 + \gamma \|x\|^2$$

Question: Suppose we sketch the problem with $\tilde{A} = SA$, $\tilde{b} = Sb$.
What regularization parameter should we use for the sketch?



[WGM18]

Finding optimal regularization parameters

Correcting implicit regularization with RMT

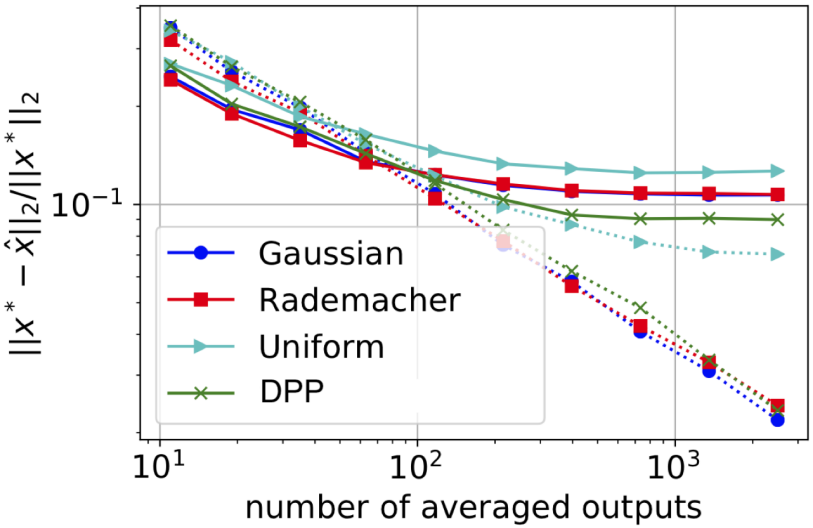
Suppose $\tilde{x} = \operatorname{argmin}_x \|\tilde{A}x - \tilde{b}\|^2 + \gamma'\|x\|^2$ is based on sketch size l. Then, defining $d_\gamma = \operatorname{tr}(A^\top A(A^\top A + \gamma I)^{-1})$ as the γ-effective dimension of A,

$$\mathbb{E}[\tilde{x}] \asymp x^*, \qquad \text{if} \quad \gamma' = \gamma \cdot \left(1 - \frac{d_\gamma}{l}\right),$$

Experiment: Distributed averaging

$$\hat{x} = \frac{1}{m}\sum_{i=1}^{m} \tilde{x}_i$$

Solid lines: estimates using γ
Dotted lines: estimates using γ'



[DBPM20]

Part II: Recent and Upcoming Advances

4. Advances in RandNLA for Optimization
   - Gradient Sketch
   - Hessian Sketch
   - Sketch-and-Project
5. Advances in RandNLA for ML
   - Statistical Learning Approaches
   - Statistical Inference Approaches
   - Random Matrix Theory Approaches
6. Putting Randomness into LAPACK
   - RandBLAS/RandLAPACK
7. Concluding thoughts

Part II: Recent and Upcoming Advances

4. Advances in RandNLA for Optimization
   - Gradient Sketch
   - Hessian Sketch
   - Sketch-and-Project
5. Advances in RandNLA for ML
   - Statistical Learning Approaches
   - Statistical Inference Approaches
   - Random Matrix Theory Approaches
6. Putting Randomness into LAPACK
   - RandBLAS/RandLAPACK
7. Concluding thoughts

Standard libraries for numerical linear algebra

Basic Linear Algebra Subprograms

BLAS

Level 1. E.g.,

$$s = x^\top y$$

Level 2. E.g.,

$$y = \alpha Ax + \beta y$$

Level 3. E.g.,

$$C = \alpha AB + \beta C$$

The Linear Algebra PACKage

LAPACK

Computational routines. E.g.,

$$A = QR$$

$$A = R^\top R$$

Drivers. E.g.,

$$\min \|Ax - b\|_2^2$$

$$x = A^{-1}b$$

$$A = U\Lambda U^\top$$

Developing standard libraries for RandNLA

RandBLAS

- Library that concerns basic sketching for dense data matrices.
- Reference implementation in C++.
- Hope: it grows to become a community standard for RandNLA, in the sense that its API would see wider adoption than any single implementation.

RandLAPACK

- Library that concerns algorithms for solving traditional linear algebra problems and advanced sketching functionality.
- To be written in C++, build on BLAS++/LAPACK++ portability layer
- Main drivers:
  - Least squares and optimization.
  - Low-rank approximation.
  - Full-rank decompositions.

"The RandLAPACK book"

Mathematics > Numerical Analysis

[Submitted on 22 Feb 2023]

Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software

Riley Murray, James Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E. Lopes, Tianyu Liang, Hengrui Luo, Jack Dongarra

Randomized numerical linear algebra – RandNLA, for short – concerns the use of randomization as a resource to develop improved algorithms for large-scale linear algebra computations.

The origins of contemporary RandNLA lay in theoretical computer science, where it blossomed from a simple idea: randomization provides an avenue for computing approximate solutions to linear algebra problems more efficiently than deterministic algorithms. This idea proved fruitful in the development of scalable algorithms for machine learning and statistical data analysis applications. However, RandNLA's true potential only came into focus upon integration with the fields of numerical analysis and "classical" numerical linear algebra. Through the efforts of many individuals, randomized algorithms have been developed that provide full control over the accuracy of their solutions and that can be every bit as reliable as algorithms that might be found in libraries such as LAPACK. Recent years have even seen the incorporation of certain RandNLA methods into MATLAB, the NAG Library, NVIDIA's cuSOLVER, and SciPy.

For all its success, we believe that RandNLA has yet to realize its full potential. In particular, we believe the scientific community stands to benefit significantly from suitably defined "RandBLAS" and "RandLAPACK" libraries, to serve as standards conceptually analogous to BLAS and LAPACK. This 200-page monograph represents a step toward defining such standards. In it, we cover topics spanning basic sketching, least squares and optimization, low-rank approximation, full matrix decompositions, leverage score sampling, and sketching data with tensor product structures (among others). Much of the provided pseudo-code has been tested via publicly available Matlab and Python implementations.

Comments: v1: this is the first arXiv release of LAPACK Working Note 299

Subjects: **Numerical Analysis (math.NA)**; Mathematical Software (cs.MS); Optimization and Control (math.OC)

Cite as: arXiv:2302.11474 **[math.NA]**

(or arXiv:2302.11474v1 **[math.NA]** for this version)

https://doi.org/10.48550/arXiv.2302.11474

"The RandLAPACK book"

Table of Contents

"Randomized Numerical Linear Algebra: A Perspective on the Field with an Eye to Software," arXiv:2302.11474, [MDM+23]

Part II: Recent and Upcoming Advances

Lots of things we did not cover

- Data-driven methods for learning good sketching operators, e.g., [IVY19, IWW21, CDD+23]
- RandNLA methods for tensor decompositions, e.g., [WTSA15, LHW17, EMBK20, JKW21]
- RandNLA methods in streaming/online environments, e.g., [BDM+20]
- Probabilistic numerics, e.g., [BCIH18]
- Hutchinson and spectral function methods
- Randomness deep inside an algorithm, e.g., for pivot rule decisions
- Theory/practice for neural networks, e.g., NTK, Nystromformer, etc.
- Graph-related and other structured matrices, e.g., Laplacians
- ...

"Concluding thoughts"

RandNLA:

- **Major interdisciplinary area**: b/w TCS, NLA, scientific computing, ML
- **Recent years: major developments** in implementation (e.g., GPUs, hardware) and application (ML vs scientific/engineering ML) motivations
- Existing (strong) theoretical foundations: needs revisiting (updating)
- **Recent years: major developments** in theory
- Leads to improvements: in {inferential objectives, iterative algorithms, etc.} for {old, new} ML problems
- **RandBLAS/RandLAPACK**: implement theory "lower in the stack"

Main theme for today:

- Identify core linear algebraic structures and build algorithmic / statistical methods around them (rather than tacking them on later as a "band aid")
- A good way to build robust (theoretical and practical) ML pipelines and avoid lots of problems later ...

References I

Naman Agarwal, Brian Bullins, and Elad Hazan.
Second-order stochastic optimization for machine learning in linear time.
The Journal of Machine Learning Research, 18(1):4148–4187, 2017.

Haim Avron, Kenneth L Clarkson, and David P Woodruff.
Faster kernel ridge regression using sketching and preconditioning.
SIAM Journal on Matrix Analysis and Applications, 38(4):1116–1138, 2017.

Nima Anari and Michał Dereziński.
Isotropy and Log-Concave Polynomials: Accelerated Sampling and High-Precision Counting of Matroid Bases.
Proceedings of the 61st Annual Symposium on Foundations of Computer Science, 2020.

Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei.
Monte carlo markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes.
In 29th Annual Conference on Learning Theory, pages 103–115. PMLR, 23–26 Jun 2016.

Nima Anari, Yang P Liu, and Thuy-Duong Vuong.
Optimal sublinear sampling of spanning trees and determinantal point processes via average-case entropic independence.
In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pages 123–134. IEEE, 2022.

Ahmed El Alaoui and Michael W. Mahoney.
Fast randomized kernel ridge regression with statistical guarantees.
In Proceedings of the 28th International Conference on Neural Information Processing Systems, pages 775–783, 2015.

References II

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

Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang.
Near-optimal design of experiments via regret minimization.
In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 126–135, Sydney, Australia, August 2017.

Francis Bach.
Sharp analysis of low-rank kernel matrix approximations.
In Conference on learning theory, pages 185–209. PMLR, 2013.

Oleg Balabanov, Matthias Beaupère, Laura Grigori, and Victor Lederer.
Block subsampled randomized Hadamard transform for low-rank approximation on distributed architectures.
arXiv preprint arXiv:2210.11295, 2022.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal.
Exact and inexact subsampled newton methods for optimization.
IMA Journal of Numerical Analysis, 39(2), 2018.

Albert S Berahas, Raghu Bollapragada, and Jorge Nocedal.
An investigation of newton-sketch and subsampled newton methods.
Optimization Methods and Software, 35(4):661–680, 2020.

S. Bartels, J. Cockayne, I. C. F. Ipsen, and P. Hennig.
Probabilistic linear solvers: A unifying view.
Technical Report Preprint: arXiv:1810.03398, 2018.

References III

Simon Bartels, Jon Cockayne, Ilse CF Ipsen, and Philipp Hennig.
Probabilistic linear solvers: a unifying view.
Statistics and Computing, 29:1249–1263, 2019.

Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P Woodruff, and Samson Zhou.
Near optimal linear algebra in the online and sliding window models.
In 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pages 517–528. IEEE, 2020.

C. Boutsidis, M. W. Mahoney, and P. Drineas.
An improved approximation algorithm for the column subset selection problem.
In Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 968–977, 2009.

David Burt, Carl Edward Rasmussen, and Mark Van Der Wilk.
Rates of convergence for sparse variational Gaussian process regression.
In Proceedings of the 36th International Conference on Machine Learning, pages 862–871, 2019.

Zhidong Bai and Jack W Silverstein.
Spectral analysis of large dimensional random matrices, volume 20.
Springer, 2010.

Ainesh Bakshi and David Woodruff.
Sublinear time low-rank approximation of distance matrices.
Advances in Neural Information Processing Systems, 31, 2018.

Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco.
Gaussian process optimization with adaptive sketching: Scalable and no regret.
In Conference on Learning Theory, 2019.

References IV


Xue Chen and Michał Dereziński.
Query complexity of least absolute deviation regression via robust uniform convergence.
In Conference on Learning Theory, pages 1144–1179. PMLR, 2021.


Younghyun Cho, James W Demmel, Michał Dereziński, Haoyun Li, Hengrui Luo, Michael W Mahoney, and Riley J Murray.
Surrogate-based autotuning for randomized sketching algorithms in regression problems.
arXiv preprint arXiv:2308.15720, 2023.


Shabarish Chenakkod, Michał Dereziński, Xiaoyu Dong, and Mark Rudelson.
Optimal embedding dimension for sparse subspace embeddings.
arXiv preprint arXiv:2311.10680, 2023.


Daniele Calandriello, Michal Derezinski, and Michal Valko.
Sampling from a k-dpp without looking at all items.
Advances in Neural Information Processing Systems, 33:6889–6899, 2020.


Jocelyn T Chi and Ilse CF Ipsen.
Randomized least squares regression: Combining model-and algorithm-induced uncertainties.
Tehnical Report, 1:34, 2018.


Jon Cockayne, Ilse CF Ipsen, Chris J Oates, and Tim W Reid.
Probabilistic iterative methods for linear systems.
The Journal of Machine Learning Research, 22(1):10505–10538, 2021.


Cheng Chen, Yi Li, and Yiming Sun.
Online active regression.
In International Conference on Machine Learning, pages 3320–3335. PMLR, 2022.

References V

Xue Chen and Eric Price.
Active regression via linear-sample sparsification.
In Proceedings of the 32nd Conference on Learning Theory, pages 663–695, 2019.

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, STOC '13, pages 81–90, New York, NY, USA, 2013. ACM.

Michał Dereziński, Burak Bartan, Mert Pilanci, and Michael W Mahoney.
Debiasing distributed second order optimization with surrogate sketching and scaled regularization.
Advances in Neural Information Processing Systems, 33:6684–6695, 2020.

Michał Dereziński, Kenneth L. Clarkson, Michael W. Mahoney, and Manfred K. Warmuth.
Minimax experimental design: Bridging the gap between statistical and worst-case approaches to least squares regression.
In Proceedings of the Thirty-Second Conference on Learning Theory, pages 1050–1069, 2019.

Michał Dereziński, Daniele Calandriello, and Michal Valko.
Exact sampling of determinantal point processes with sublinear time preprocessing.
In Advances in Neural Information Processing Systems, pages 11542–11554, 2019.

Michał Dereziński.
Fast determinantal point processes via distortion-free intermediate sampling.
In Proceedings of the 32nd Conference on Learning Theory, pages 1029–1049, 2019.

Michał Dereziński.
Algorithmic gaussianization through sketching: Converting data into sub-gaussian random designs.
In The Thirty Sixth Annual Conference on Learning Theory, pages 3137–3172. PMLR, 2023.

References VI

P. Drineas, I. Ipsen, E. Kontopoulou, and M. Magdon-Ismail.
Structural convergence results for approximations of dominant subspaces from block Krylov spaces.
SIAM Journal on Matrix Analysis and Applications, 39:567–586, 2018.

P. Drineas, R. Kannan, and M. W. Mahoney.
Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication.
SIAM Journal on Computing, 36:132–157, 2006.

Michał Dereziński, Rajiv Khanna, and Michael W Mahoney.
Improved guarantees and a multiple-descent curve for the column subset selection problem and the nyström method.
In Advances in Neural Information Processing Systems, volume 33, pages 4953–4964, 2020.

Edgar Dobriban and Sifan Liu.
Asymptotics for sketching in least squares regression.
Advances in Neural Information Processing Systems, 32, 2019.

Michał Dereziński, Zhenyu Liao, Edgar Dobriban, and Michael Mahoney.
Sparse sketches with small inversion bias.
In Conference on Learning Theory, pages 1467–1510. PMLR, 2021.

Michal Derezinski, Feynman Liang, and Michael Mahoney.
Bayesian experimental design using regularized determinantal point processes.
In International Conference on Artificial Intelligence and Statistics, pages 3197–3207, 2020.

Michał Dereziński, Feynman T Liang, and Michael W Mahoney.
Exact expressions for double descent and implicit regularization via surrogate random design.
Advances in neural information processing systems, 33:5152–5164, 2020.

References VII

Michał Dereziński, Jonathan Lacotte, Mert Pilanci, and Michael W Mahoney.
Newton-less: Sparsification without trade-offs for the sketched newton update.
Advances in Neural Information Processing Systems, 34:2835–2847, 2021.

David Durfee, Kevin A Lai, and Saurabh Sawlani.
l1 regression using lewis weights preconditioning and stochastic gradient descent.
In Conference On Learning Theory, pages 1626–1656. PMLR, 2018.

P. Drineas and M. W. Mahoney.
RandNLA: Randomized numerical linear algebra.
Communications of the ACM, 59:80–90, 2016.

P. Drineas and M. W. Mahoney.
Lectures on randomized numerical linear algebra.
In The Mathematics of Data, IAS/Park City Mathematics Series, pages 1–48. AMS/IAS/SIAM, 2018.

Michał Dereziński and Michael W Mahoney.
Distributed estimation of the inverse Hessian by determinantal averaging.
In Advances in Neural Information Processing Systems 32, pages 11401–11411. 2019.

Michał Dereziński and Michael W Mahoney.
Determinantal point processes in randomized numerical linear algebra.
Notices of the American Mathematical Society, 68(1):34–45, 2021.

Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff.
Fast approximation of matrix coherence and statistical leverage.
J. Mach. Learn. Res., 13(1):3475–3506, 2012.

References VIII

Petros Drineas, Michael W Mahoney, and S Muthukrishnan.
Sampling algorithms for ℓ_2 regression and applications.
In Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pages 1127–1136, 2006.

Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan.
Relative-error CUR matrix decompositions.
SIAM J. Matrix Anal. Appl., 30(2):844–881, 2008.

P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós.
Faster least squares approximation.
Numerische Mathematik, 117(2):219–249, 2010.

Michał Dereziński and Elizaveta Rebrova.
Sharp analysis of sketch-and-project methods via a connection to randomized singular value decomposition.
arXiv preprint arXiv:2208.09585, 2022.

Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang.
Matrix approximation and projective clustering via volume sampling.
In Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, pages 1117–1126, January 2006.

Michał Dereziński and Manfred K. Warmuth.
Unbiased estimates for linear regression via volume sampling.
In Advances in Neural Information Processing Systems 30, pages 3087–3096, 2017.

References IX


Michał Dereziński, Manfred K. Warmuth, and Daniel Hsu.
Leveraged volume sampling for linear regression.
In Advances in Neural Information Processing Systems 31, pages 2510–2519. 2018.


Michał Dereziński and Jiaming Yang.
Solving linear systems faster than via preconditioning.
2023.


Murat A Erdogdu and Andrea Montanari.
Convergence rates of sub-sampled Newton methods.
Advances in Neural Information Processing Systems, 28:3052–3060, 2015.


N Benjamin Erichson, Krithika Manohar, Steven L Brunton, and J Nathan Kutz.
Randomized cp tensor decomposition.
Machine Learning: Science and Technology, 1(2):025012, 2020.


Valerii V Fedorov.
Theory of optimal experiments.
Probability and mathematical statistics. Academic Press, New York, NY, USA, 1972.


Michaël Fanuel, Joachim Schreurs, and Johan AK Suykens.
Diversity sampling is an implicit regularization for kernel methods.
arXiv:2002.08616, 2020.


Zachary Frangella, Joel A Tropp, and Madeleine Udell.
Randomized Nystrom preconditioning.
arXiv preprint arXiv:2110.02820, 2021.

References X

Robert Gower, Filip Hanzely, Peter Richtárik, and Sebastian U Stich.
Accelerated stochastic matrix inversion: general theory and speeding up bfgs rules for faster second-order optimization.
Advances in Neural Information Processing Systems, 31, 2018.

Robert Gower, Dmitry Kovalev, Felix Lieder, and Peter Richtárik.
Rsn: Randomized subspace newton.
Advances in Neural Information Processing Systems, 32, 2019.

Jennifer Gillenwater, Alex Kulesza, Zelda Mariet, and Sergei Vassilvtiskii.
A tree-based method for fast repeated sampling of determinantal point processes.
In Proceedings of the 36th International Conference on Machine Learning, pages 2260–2268, 2019.

Robert Gower, Nicolas Le Roux, and Francis Bach.
Tracking the gradients using the hessian: A new look at variance reducing stochastic methods.
In International Conference on Artificial Intelligence and Statistics, pages 707–715. PMLR, 2018.

Alex Gittens and Michael W. Mahoney.
Revisiting the Nyström method for improved large-scale machine learning.
J. Mach. Learn. Res., 17(1):3977–4041, 2016.

Alon Gonen, Francesco Orabona, and Shai Shalev-Shwartz.
Solving ridge regression using sketched preconditioned svrg.
In International conference on machine learning, pages 1397–1405. PMLR, 2016.

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

References XI

Venkatesan Guruswami and Ali K. Sinop.
Optimal column-based low-rank matrix reconstruction.
In Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1207–1214, 2012.

Filip Hanzely, Nikita Doikov, Yurii Nesterov, and Peter Richtarik.
Stochastic subspace cubic newton method.
In International Conference on Machine Learning, pages 4027–4038. PMLR, 2020.

J Ben Hough, Manjunath Krishnapur, Yuval Peres, and Bálint Virág.
Determinantal processes and independence.
Probability surveys, 3:206–229, 2006.

Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik.
Sega: Variance reduction via gradient sketching.
Advances in Neural Information Processing Systems, 31, 2018.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM review, 53(2):217–288, 2011.

Piotr Indyk, Ali Vakilian, and Yang Yuan.
Learning-based low-rank approximations.
Advances in Neural Information Processing Systems, 32, 2019.

Piotr Indyk, Tal Wagner, and David Woodruff.
Few-shot data-driven algorithms for low rank approximation.
Advances in Neural Information Processing Systems, 34:10678–10690, 2021.

References XII

Ruhui Jin, Tamara G Kolda, and Rachel Ward.
Faster johnson–lindenstrauss transforms via kronecker products.
Information and Inference: A Journal of the IMA, 10(4):1533–1562, 2021.

M. S. Kaczmarz.
Angenaherte auflosung von systemen linearer gleichungen.
Bulletin International de l'Academie Polonaise des Sciences et des Lettres, 35:355–357, 1937.

J. Kukacka, V. Golkov, and D. Cremers.
Regularization for deep learning: A taxonomy.
Technical Report Preprint: arXiv:1710.10686, 2017.

Dmitry Kovalev, Konstantin Mishchenko, and Peter Richtárik.
Stochastic newton and cubic newton methods with simple local linear-quadratic rates.
arXiv preprint arXiv:1912.01597, 2019.

Alex Kulesza and Ben Taskar.
k-DPPs: Fixed-Size Determinantal Point Processes.
In Proceedings of the 28th International Conference on Machine Learning, pages 1193–1200, June 2011.

Alex Kulesza and Ben Taskar.
Determinantal Point Processes for Machine Learning.
Now Publishers Inc., Hanover, MA, USA, 2012.

R. Kannan and S. Vempala.
Randomized algorithms in numerical linear algebra.
Acta Mathematica, 26:95–135, 2017.

References XIII


Miles E Lopes, N Benjamin Erichson, and Michael W Mahoney.
Bootstrapping the operator norm in high dimensions: Error estimation for covariance matrices and sketching.
Bernoulli, 29(1):428–450, 2023.


Yanli Liu, Fei Feng, and Wotao Yin.
Acceleration of svrg and katyusha x by inexact preconditioning.
In International Conference on Machine Learning, pages 4003–4012. PMLR, 2019.


Xingguo Li, Jarvis Haupt, and David Woodruff.
Near optimal sketching of low-rank tensor regression.
Advances in Neural Information Processing Systems, 30, 2017.


Daniel LeJeune, Hamid Javadi, and Richard Baraniuk.
The implicit regularization of ordinary least squares ensembles.
In International Conference on Artificial Intelligence and Statistics, pages 3525–3535. PMLR, 2020.


Dennis Leventhal and Adrian S Lewis.
Randomized methods for linear constraints: convergence rates and conditioning.
Mathematics of Operations Research, 35(3):641–654, 2010.


Jonathan Lacotte, Sifan Liu, Edgar Dobriban, and Mert Pilanci.
Optimal iterative sketching methods with the subsampled randomized Hadamard transform.
Advances in Neural Information Processing Systems, 33:9725–9735, 2020.


Jonathan Lacotte and Mert Pilanci.
Adaptive and oblivious randomized subspace methods for high-dimensional optimization: Sharp analysis and lower bounds.
IEEE Transactions on Information Theory, 68(5):3281–3303, 2022.

References XIV

Daniel LeJeune, Pratik Patil, Hamid Javadi, Richard G Baraniuk, and Ryan J Tibshirani.
Asymptotics of the sketched pseudoinverse.
arXiv preprint arXiv:2211.03751, 2022.

Yin Tat Lee and Aaron Sidford.
Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems.
In 2013 ieee 54th annual symposium on foundations of computer science, pages 147–156. IEEE, 2013.

Miles Lopes, Shusen Wang, and Michael Mahoney.
Error estimation for randomized least-squares algorithms via the bootstrap.
In International Conference on Machine Learning, pages 3217–3226. PMLR, 2018.

Miles E Lopes, Shusen Wang, and Michael W Mahoney.
A bootstrap method for error estimation in randomized matrix multiplication.
The Journal of Machine Learning Research, 20(1):1434–1473, 2019.

M. W. Mahoney.
Randomized algorithms for matrices and data.
Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011.

M. W. Mahoney.
Approximate computation and implicit regularization for very large-scale data analysis.
In Proceedings of the 31st ACM Symposium on Principles of Database Systems, pages 143–154, 2012.

M. W. Mahoney.
Lecture notes on randomized linear algebra.
Technical Report Preprint: arXiv:1608.04481, 2016.

References XV

P.-G. Martinsson.
Randomized methods for matrix computations.
In M. W. Mahoney, J. C. Duchi, and A. C. Gilbert, editors, The Mathematics of Data, IAS/Park City Mathematics Series, pages 187–230. AMS/IAS/SIAM, 2018.

Ping Ma, Yongkai Chen, Xinlian Zhang, Xin Xing, Jingyi Ma, and Michael W Mahoney.
Asymptotic analysis of sampling estimators for randomized numerical linear algebra algorithms.
The Journal of Machine Learning Research, 23(1):7970–8014, 2022.

M. W. Mahoney and P. Drineas.
Structural properties underlying high-quality randomized numerical linear algebra algorithms.
In Handbook of Big Data, pages 137–154. CRC Press, 2016.

Mojmir Mutny, Michał Dereziński, and Andreas Krause.
Convergence analysis of block coordinate algorithms with determinantal sampling.
In International Conference on Artificial Intelligence and Statistics, pages 3110–3120, 2020.

Riley Murray, James Demmel, Michael W Mahoney, N Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E Lopes, et al.
Randomized numerical linear algebra: A perspective on the field with an eye to software.
arXiv preprint arXiv:2302.11474, 2023.

Aryan Mokhtari, Mark Eisen, and Alejandro Ribeiro.
Iqn: An incremental quasi-newton method with local superlinear convergence rate.
SIAM Journal on Optimization, 28(2):1670–1698, 2018.

Cameron Musco and Christopher Musco.
Randomized block krylov methods for stronger and faster approximate singular value decomposition.
Advances in neural information processing systems, 28, 2015.

References XVI

Cameron Musco and Christopher Musco.
Recursive sampling for the nystrom method.
Advances in neural information processing systems, 30, 2017.

C. H. Martin and M. W. Mahoney.
Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning.
Technical Report Preprint: arXiv:1810.01075, 2018.

C. H. Martin and M. W. Mahoney.
Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning.
Journal of Machine Learning Research, 22(165):1–73, 2021.

Raphael A Meyer, Cameron Musco, Christopher Musco, and David P Woodruff.
Hutch++: Optimal stochastic trace estimation.
In Symposium on Simplicity in Algorithms (SOSA), pages 142–155. SIAM, 2021.

Cameron Musco, Christopher Musco, David P Woodruff, and Taisuke Yasuda.
Active linear regression for lp norms and beyond.
In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pages 744–753. IEEE, 2022.

P. Ma, M. W. Mahoney, and B. Yu.
A statistical perspective on algorithmic leveraging.
Journal of Machine Learning Research, 16:861–911, 2015.

References XVII

X. Meng, M. A. Saunders, and M. W. Mahoney.
LSRN: A parallel iterative solver for strongly over- or under-determined systems.
SIAM Journal on Scientific Computing, 36(2):C95–C118, 2014.

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

Cameron Musco and David P Woodruff.
Sublinear time low-rank approximation of positive semidefinite matrices.
In 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pages 672–683. IEEE, 2017.

Sen Na, Michał Dereziński, and Michael W Mahoney.
Hessian averaging in stochastic newton methods achieves superlinear convergence.
arXiv preprint arXiv:2204.09266, 2022.
Accepted for publication, Mathematical Programming.

B. Neyshabur.
Implicit regularization in deep learning.
Technical report, 2017.
Preprint: arXiv:1709.01953.

Deanna Needell and Joel A Tropp.
Paved with good intentions: analysis of a randomized block kaczmarz method.
Linear Algebra and its Applications, 441:199–221, 2014.

References XVIII

Deanna Needell and Rachel Ward.
Two-subspace projection method for coherent overdetermined systems.
Journal of Fourier Analysis and Applications, 19(2):256–269, 2013.

Deanna Needell, Rachel Ward, and Nati Srebro.
Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm.
Advances in neural information processing systems, 27, 2014.

Aditya Parulekar, Advait Parulekar, and Eric Price.
L1 regression with lewis weights subsampling.
Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 2021.

Friedrich Pukelsheim.
Optimal Design of Experiments.
Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

M. Pilanci and M. J. Wainwright.
Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares.
Journal of Machine Learning Research, 17(53):1–38, 2016.

Mert Pilanci and Martin J Wainwright.
Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence.
SIAM Journal on Optimization, 27(1):205–245, 2017.

Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco.
On fast leverage score sampling and optimal learning.
In Advances in Neural Information Processing Systems 31, pages 5672–5682. 2018.

References XIX


Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco.
Falkon: An optimal large scale kernel method.
Advances in neural information processing systems, 30, 2017.


Anton Rodomanov and Dmitry Kropotov.
A randomized coordinate descent method with volume sampling.
SIAM Journal on Optimization, 30(3):1878–1904, 2020.


Farbod Roosta-Khorasani and Michael W Mahoney.
Sub-sampled newton methods.
Mathematical Programming, 174(1):293–326, 2019.


G. Raskutti and M. W. Mahoney.
A statistical perspective on randomized sketching for ordinary least-squares.
Journal of Machine Learning Research, 17(214):1–31, 2016.


Elizaveta Rebrova and Deanna Needell.
On block Gaussian sketching for the kaczmarz method.
Numerical Algorithms, 86:443–473, 2021.


Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora.
Fetchsgd: Communication-efficient federated learning with sketching.
In International Conference on Machine Learning, pages 8253–8265. PMLR, 2020.


V. Rokhlin, A. Szlam, and M. Tygert.
A randomized algorithm for principal component analysis.
SIAM Journal on Matrix Analysis and Applications, 31(3):1100–1124, 2009.

References XX

Vladimir Rokhlin and Mark Tygert.
A fast randomized algorithm for overdetermined linear least-squares regression.
Proceedings of the National Academy of Sciences, 105(36):13212–13217, 2008.

Peter Richtárik and Martin Takác.
Stochastic reformulations of linear systems: algorithms and convergence theory.
SIAM Journal on Matrix Analysis and Applications, 41(2):487–524, 2020.

Mark Rudelson and Roman Vershynin.
Hanson-Wright inequality and sub-gaussian concentration.
Electronic Communications in Probability, 18, 2013.

Tamas Sarlos.
Improved approximation algorithms for large matrices via random projections.
In Proceedings of the Symposium on Foundations of Computer Science, FOCS '06, pages 143–152, 2006.

Masashi Sugiyama and Shinichi Nakajima.
Pool-based active learning in approximate linear regression.
Mach. Learn., 75(3):249–274, June 2009.

Thomas Strohmer and Roman Vershynin.
A randomized Kaczmarz algorithm with exponential convergence.
Journal of Fourier Analysis and Applications, 15(2):262–278, 2009.

J. A. Tropp.
An introduction to matrix concentration inequalities.
Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2015.

References XXI

S. Wang, A. Gittens, and M. W. Mahoney.
Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging.
Journal of Machine Learning Research, 18(218):1–50, 2018.

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

Shusen Wang, Fred Roosta, Peng Xu, and Michael W Mahoney.
GIANT: globally improved approximate newton method for distributed optimization.
Advances in Neural Information Processing Systems, 31:2332–2342, 2018.

Christopher K. I. Williams and Matthias Seeger.
Using the Nyström method to speed up kernel machines.
In Advances in Neural Information Processing Systems 13, pages 682–688. 2001.

Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar.
Fast and guaranteed tensor decomposition via sketching.
Advances in neural information processing systems, 28, 2015.

Yining Wang, Adams W. Yu, and Aarti Singh.
On computationally tractable selection of experiments in measurement-constrained regression models.
J. Mach. Learn. Res., 18(1):5238–5278, January 2017.

Jiyan Yang, Yin-Lam Chow, Christopher Ré, and Michael W Mahoney.
Weighted sgd for lp regression with randomized preconditioning.
The Journal of Machine Learning Research, 18(1):7811–7853, 2017.

References XXII

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney.
Pyhessian: Neural networks through the lens of the hessian.
In 2020 IEEE international conference on big data (Big data), pages 581–590. IEEE, 2020.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney.
Adahessian: An adaptive second order optimizer for machine learning.
In proceedings of the AAAI conference on artificial intelligence, volume 35, 2021.

Fan Yang, Sifan Liu, Edgar Dobriban, and David P Woodruff.
How to reduce dimension with pca and random projections?
arXiv preprint arXiv:2005.00511, 2020.

Rui Yuan, Alessandro Lazaric, and Robert M Gower.
Sketched newton–raphson.
SIAM Journal on Optimization, 32(3):1555–1583, 2022.