Lecture Notes

STAT 260 with Michael Mahoney

Spring 2025

Contents

0 Overview of these Notes

These are notes from "Stat260: Randomized Linear Algebra, Optimization, and Large-Scale Learning" taught during the Spring 2025 semester. Michael Mahoney taught the class; individual students put together individual sections, one for each lecture.

Here is the link to assignments:
https://docs.google.com/spreadsheets/d/1ayjpaJjkxesmoLxeg29HL2aW15JvhwDYj4ZICXh7xnE/edit?usp=sharing

Here is the link to the class web page:
https://www.stat.berkeley.edu/~mmahoney/s25-stat260/

1 Tuesday, January 21

1.1 Topics: Course Information, Introduction, and Overview

Main References: Chapter 1 of LN-RLA

2 Thursday, January 23

2.1 Topics: Approximating Matrix Multiplication

Main References: Section 4 of: "Lectures on Randomized Numerical Linear Algebra," Drineas and Mahoney (PCMI 2017) (arxiv)

Backup: Chapter 2 of LN-RLA; Drineas, Kannan, and Mahoney, "Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication"

3 Tuesday, January 28

3.1 Topics: Scalar and Matrix Concentration

Main References: Chapter 3-5 of LN-RLA

Backup: Section 4 of: "Lectures on Randomized Numerical Linear Algebra," Drineas and Mahoney (PCMI 2017)

4 Thursday, January 30

4.1 Topics: Scalar and Matrix Concentration, Cont.

Main References: Same as last class. We will cover Thm 6, Thm 8, and (hopefully) Lemma 15 of Chapter 5 of LN-RLA.

5 Tuesday, February 4th (Scribe: Rahul Shah)

5.1 Logistics

5.1.1 Gradescope

We will use Gradescope for this class. The course entry code is

`BKV62P`.

5.1.2 Office Hours

Office Hours (OH) will be:

- Wednesday 4–5pm (via Zoom)
- Thursday 5–6pm (Evans 307)

5.1.3 Homework Typos

In the homework, there is a typo related to the Sherman–Woodbury formula: the "+" that appears in the statement should actually be a "−". Please keep that correction in mind when completing your assignment.

The homework is due next Tuesday. Note that we do *not* have class then because I will be flying out to the Middle East. In my absence, my postdoc will discuss how these ideas are used in RandLAPACK.

5.1.4 Scribe Logistics

Please sign up to be a scribe through Gradescope! Scribing will help ensure we have a complete set of notes and that everyone can benefit from them.

5.2 Recap from Last Time

We discussed how to *reduce* dimension or sample data in a way that still preserves important structures. One way is to use the *Frobenius* or *spectral* norms to guide sampling probabilities.

We have n points (endpoints) in $\mathbb{R}^d$, denoted $\{u_i\}_{i=1}^n$. We wish to embed (or map) these points into a much lower-dimensional space $\mathbb{R}^k$ (with $k \ll d$), resulting in $\{v_i\}_{i=1}^n \subset \mathbb{R}^k$. Typically, one wants to preserve pairwise distances:

$$\|u_i - u_{i'}\|_2 \quad \text{and} \quad \|v_i - v_{i'}\|_2$$

so that

$$\|u_i - u_{i'}\|_2 \;\approx\; \|v_i - v_{i'}\|_2 \quad \text{(up to some factor } (1 \pm \varepsilon)\text{)}.$$

Even if such a low-dimensional representation $\{v_i\}$ exists, one also wants an *efficient* way to find or construct it.

5.3 Johnson–Lindenstrauss (JL) Lemma

A classic result giving a simple, randomized approach to do this embedding is the Johnson–Lindenstrauss Lemma. Define a random linear map $P \in \mathbb{R}^{d\times k}$ by sampling each entry

$$P_{ij} \;\sim\; \frac{1}{\sqrt{k}}\,\mathcal{N}(0,1),$$

i.e., each P_{ij} is an independent Gaussian with mean 0 and variance $1/k$. Then we map each u_i to

$$v_i \;=\; u_i\,P \;\in\; \mathbb{R}^k.$$

This preserves distances with high probability when k is on the order of $\frac{1}{\varepsilon^2}\log(n)$.

Theorem (Johnson–Lindenstrauss Lemma). Let $\varepsilon \in (0,1)$. Suppose we have n points $\{u_i\}_{i=1}^n \subset \mathbb{R}^d$. Choose a random matrix $P \in \mathbb{R}^{d\times k}$ with i.i.d. entries $P_{ij} \sim \frac{1}{\sqrt{k}}\mathcal{N}(0,1)$. If $k \geq C\,\varepsilon^{-2}\log(n)$ for some universal constant C, then with probability at least $\frac{1}{2}$ (or arbitrarily close to 1 if you increase the constant), we have

$$(1-\varepsilon)\,\|\,u_i - u_{i'}\|_2 \;\leq\; \|\,v_i - v_{i'}\|_2 \;\leq\; (1+\varepsilon)\,\|\,u_i - u_{i'}\|_2 \quad \text{for all } 1 \leq i < i' \leq n.$$

Hence, all pairwise distances among the n points are preserved up to a $(1\pm\varepsilon)$ factor. Equivalently, norms are preserved up to $(1\pm\varepsilon)$, since $\|u_i - u_{i'}\|_2 \approx \|v_i - v_{i'}\|_2$.

Proof Sketch.

1. **Expected norm preservation.** Fix a single vector $u \in \mathbb{R}^d$. Define $v = uP \in \mathbb{R}^k$. We want to show that the expected squared norm of v matches $\|u\|_2^2$. Indeed,

$$\mathbb{E}\big[\|v\|_2^2\big] \;=\; \mathbb{E}\Big[\sum_{j=1}^{k}\Big(\sum_{i=1}^{d} u_i P_{ij}\Big)^2\Big].$$

   Because P_{ij} are Gaussians with variance $1/k$, this expectation works out to

$$\mathbb{E}[\|v\|_2^2] \;=\; \|u\|_2^2.$$

   (Each coordinate in v is an average of Gaussians scaled by u_i, and the cross-terms vanish in expectation.)

2. **Concentration around the mean.** Although $\mathbb{E}[\|v\|_2^2] = \|u\|_2^2$, we need the actual value $\|v\|_2^2$ to stay close to $\|u\|_2^2$ *most of the time*. One uses tail bounds (e.g. via moment generating functions of χ^2-type sums) to show

$$\mathbb{P}\big[\,(1-\varepsilon)\,\|u\|_2^2 \;\leq\; \|v\|_2^2 \;\leq\; (1+\varepsilon)\,\|u\|_2^2\big] \;\geq\; 1 - 2\,\exp\!\big(-c\,k\,\varepsilon^2\big)$$

   for some constant $c > 0$.

3. **Union bound over n points.** We need this concentration to hold for $\binom{n}{2}$ differences of the points $u_i - u_{i'}$. Provided $k \geq C\,\varepsilon^{-2}\log(n)$ for a large enough constant C, a union bound ensures *all* pairwise distances are preserved with high probability.
4. **Conclusion.** Thus, with probability at least $1/2$ (and typically boosted to near 1 in many applications), the linear map P preserves all pairwise distances up to a $(1 \pm \varepsilon)$ factor.

□

5.4 Random Projection Variants and Remarks

- *Gaussian* random matrices are a common choice for P, but there are also faster or more structured choices:
  - ± 1 (Rademacher) entries, scaled appropriately.
  - Hadamard-based transforms.
  - *Sparse* embeddings: e.g., each column has only a few nonzero entries, which makes the transform cheaper to apply.
- In machine learning, techniques like "dropout" can be viewed as random masks (though not exactly the same as an embedding guarantee).
- The key point is that random low-dimensional subspaces typically do *not* align with any special direction among the data. It is therefore unlikely that many points collapse onto the same direction.

5.5 Matrix Sketching for Fast Multiplication

A related application is to approximate matrix products AB by something like $C\,R$, where C and R are typically smaller than A or B. For instance:

$$AB \;\approx\; \underbrace{A\,\Pi}_{=C}\;\underbrace{\Pi^T B}_{=R},$$

where Π is a $\big(d \times k\big)$ random projection matrix. If $k \ll d$, then C and R have dimension $(n \times k)$ and $(k \times m)$ respectively, which can be much smaller to store or handle than $(n \times d)$ or $(d \times m)$.

5.6 A Word of Caution: Pseudoinverses and Products

Warning. A known identity for Moore–Penrose pseudoinverses is

$$(AB)^{+} \;=\; B^{+}\,A^{+}$$

only when A and B have compatible dimensions and the product AB is (full-rank) invertible.

Do *not* use this identity if the matrices are not square/invertible or if the ranks do not match up properly.

5.7 Least-Squares Review

We often encounter least-squares problems:

$$\min_x \ \|Ax - b\|_2^2 \quad \Longrightarrow \quad f(x) = (Ax - b)^T (Ax - b).$$

From basic calculus or linear algebra, the solution is

$$\nabla f(x) = 0 \quad \Longrightarrow \quad A^T A\, x \ = \ A^T b \quad \Longrightarrow \quad x_{\text{opt}} \ = \ (A^T A)^{-1}\, A^T\, b.$$

Using the SVD $A = U\,\Sigma\,V^T$, we get

$$x_{\text{opt}} \ = \ (A^T A)^{-1}\, A^T\, b \ = \ V\,\Sigma^{-1}\, U^T\, b.$$

We can also solve such systems using direct decompositions:

- **Cholesky**: factor $A^T A = R^T R$.
- **QR**: factor $A = Q\,R$. Then $Ax \approx b$ becomes $Q\,R\,x \approx b \implies R\,x \approx Q^T b$.
- **SVD**: factor $A = U\,\Sigma\,V^T$, then $Ax \approx b \implies U\,\Sigma\,V^T\,x \approx b \implies x_{\text{opt}} = V\,\Sigma^{-1}\,U^T\,b$.

5.7.1 Iterative Methods

In large-scale or sparse problems, *iterative* methods are often more practical than directly computing a factorization (QR, SVD, etc.). One can project onto a certain subspace, update in a direction, and so on, until convergence.

Geometrically:

$$b = b_{\|} + b_{\perp}, \quad \text{where } b_{\|} \in \text{span}(A),\ b_{\perp} \perp \text{span}(A).$$

Then

$$x_{\text{opt}} \ = \ A^{+}\, b.$$

Sometimes, one can *sketch* the system $A\,x \approx b$ by randomly sampling or projecting rows:

$$\min_x \ \|\, A\,x - b\,\| \qquad \longrightarrow \ \min_x \ \|\, S\,A\,x - S\,b\,\|,$$

where S is a (much smaller) random matrix or sampling operator. Provided $S\,U$ (where $A = U\Sigma V^T$) behaves "like" an isometry, the solution $(SA)^{+}\,S\,b$ is close to the true solution $A^{+}\,b$. Thus:

$$(SU)^T (SU) \ \approx \ I \quad \Longrightarrow \quad \|\, I - (SU)^T (SU)\|_2 \text{ small}.$$

If $\|I - (SU)^T(SU)\|_2 \ \leq \ \frac{1}{2}$, for instance, then with high probability the geometry of U is well preserved in the sketched space, yielding a good approximate solution.

These randomized methods and sketches (via Johnson–Lindenstrauss-like constructions or other fast transforms) play a key role in *randomized numerical linear algebra* (RandNLA), such as RandLAPACK.

Sometimes dropping or sampling part of your data in the right (rigorous) way can lead to big computational gains without losing significant accuracy!

We'll see more on specialized (sparse/structured) transformations next week — Stay tuned for more details!

6 Thursday, February 6th (Scribe: Aneesh Durai)

6.1 Least Squares

We begin by recalling the standard least-squares solution and its two natural "sketchable" forms.

$$\min_{x\in\mathbb{R}^d} \|Ax-b\|_2 \implies x = A^\dagger b \big(A^\dagger = V\Sigma^{-1}U^\top\big).$$

Equivalently, one may view the same problem as

$$\min_{x\in\mathbb{R}^d} \|xA^\top - b\|_2 \implies x = (AA^\top)^{-1}A\,b,$$

which suggests two different sketching strategies (sample rows vs. project onto a subspace).

To make either sketch provably accurate, we impose:

6.1.0.1 Condition I (Subspace Embedding):

For all $y \in \mathrm{col}(A)$,

$$\|S\,y\|_2 = (1\pm\varepsilon)\,\|y\|_2.$$

6.1.0.2 Condition II (Approximate Normal-Equation Feasibility):

The sketched right-hand side remains nearly orthogonal to $\ker(SA)$:

$$\|(SA)^\top Sb - (SA)^\top(SA)\,x_{\mathrm{opt}}\| \;\le\; \varepsilon\,\|b\|_2.$$

The table below summarizes how the choice of $X = S$ affects both the sketching cost and the final solve cost.

	Cond I	Cond II	Sketch	Solve
$X = I$	✓	✓	Fast	Slow (no reduction)
$X = U_A^\top$	✓	✓	Slow ($O(nd^2)$)	Fast ($O(d^3)$)
Basic LS (sampling)	✓	✓	Slow	Fast
Basic LS (projection)	✓	✓	Slow	Fast

In both sampling and projection variants, we replace (A, b) by the much smaller system (SA, Sb), solve that exactly, and then lift back to $\mathbb{R}^d$. The two conditions above are precisely what ensure the final answer still approximates the true minimizer.

6.2 Claim and Reformulation

Claim. If S satisfies Conditions I and II, then the sketched solution $\tilde{x}_{\text{opt}} = \arg\min_x \|SA\,x - Sb\|_2$ obeys

$$\|A\tilde{x}_{\text{opt}} - b\|_2 \;\le\; (1+\varepsilon)\,\min_x \|Ax - b\|_2.$$

To prove this, one shows equivalently that

$$\min_{x\in\mathbb{R}^d} \|S\,b^{\perp} - S\,A\,x\|_2 \;=\; \min_{z\in\mathbb{R}^r} \|S\,b^{\perp} - S\,U_A\,z\|_2,$$

where $b^{\perp} = b - A\,x_{\text{opt}}$ lies in the nullspace of $A^{\top}$, and U_A spans $\text{col}(A)$.

By writing $A(\tilde{x}_{\text{opt}} - x_{\text{opt}}) = U_A\,z_{\text{opt}}$ one checks

$$\|S\,U_A\,z_{\text{opt}} - S\,b^{\perp}\|_2 = \|A\,\tilde{x}_{\text{opt}} - b\|_2,$$

and then invokes the two conditions to finish the $(1+\varepsilon)$–bound.

6.3 Approximation Guarantee via Normal Equations

We now turn to the normal-equation proof: solving the sketched least-squares is equivalent to solving

$$\min_z \|S\,b^{\perp} - S\,U_A\,z\|_2,$$

whose optimality condition is

$$(U_A^{\top} S^{\top} S\,U_A)\,z_{\text{opt}} \;=\; U_A^{\top} S^{\top} S\,b^{\perp}.$$

Condition I ensures $U_A^{\top} S^{\top} S\,U_A \succeq (1-\varepsilon)I$, so one shows

$$\|z_{\text{opt}}\|_2^2 \;\le\; \frac{1}{1-\varepsilon}\,\|U_A^{\top} S^{\top} S\,b^{\perp}\|_2^2 \;\le\; \frac{\varepsilon}{1-\varepsilon}\,\|b^{\perp}\|_2^2.$$

which yields

$$\|b - A\tilde{x}_{\text{opt}}\|_2^2 = \|b^{\perp}\|_2^2 + \|A\,z_{\text{opt}}\|_2^2 \le (1+\varepsilon)\,\|b^{\perp}\|_2^2,$$

completing the proof.

6.4 Basic LS Sampling Algorithm

Input: $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$ **Output:** $\tilde{x}_{\text{opt}}$

1. Compute leverage scores $p_i \propto \|U_{A_{i,:}}\|_2^2$.
2. Sample $r = O(d \log d/\varepsilon^2)$ rows of A, b with probabilities $\{p_i\}$ to form (SA, Sb).
3. Solve $\tilde{x}_{\text{opt}} = (SA)^+ (Sb)$.

With probability $1-\delta$, this returns an ε–approximate solution in total time $O(nd \log d + d^3 \log(1/\delta)/\varepsilon^2)$.

6.5 Basic LS Projection Algorithm

Input: A, b **Output:** $\tilde{x}_{\text{opt}}$

1. Draw $S \in \mathbb{R}^{r \times n}$ with $S_{ij} \sim \frac{1}{\sqrt{n}} \mathcal{N}(0,1)$, $r = O(d/\varepsilon^2)$.
2. Form (SA, Sb) and solve $\tilde{x}_{\text{opt}} = (SA)^+ (Sb)$.

Again, with high probability this yields $\|A\tilde{x}_{\text{opt}} - b\| \leq (1+\varepsilon) \min_x \|Ax - b\|$ in $O(ndr + d^3)$ time.

7 Tuesday, February 11th (Scribe: Divit Rawal)

7.1 Least Squares

We consider the problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{V}\mathbf{x}-\mathbf{b}\|, \quad \text{where } \mathbf{V}\in\mathbb{R}^{n\times d},\ d\ll n.$$

A direct method is to use a QR factorization of $\mathbf{V}$, i.e. $\mathbf{V}=\mathbf{Q}\mathbf{R}$, leading to:

$$\mathbf{x}^* \ = \ \arg\min_{\mathbf{x}} \|\mathbf{V}\mathbf{x}-\mathbf{b}\| \ = \ \arg\min_{\mathbf{x}} \|\mathbf{R}\mathbf{x}-\mathbf{Q}^{\mathsf{T}}\mathbf{b}\|.$$

This approach is $\mathcal{O}(nd^2)$ operations, it is costly for very large n (multiple passes over $\mathbf{V}$, high communication overhead, etc.).

Instead of solving with the full matrix $\mathbf{V}$, we can construct a "sketch" of the system:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \big\|\Theta(\mathbf{V}\mathbf{x}-\mathbf{b})\big\|,$$

where Θ is an $\ell\times n$ random matrix ($\ell=\mathcal{O}(d)$) chosen so that it preserves the geometry of $\mathbf{V}$ and $\mathbf{b}$. In practice:

- We compute $\Theta\mathbf{V}$ and $\Theta\mathbf{b}$ efficiently (ideally in one pass over $\mathbf{V}$).
- Solve the smaller least squares system $\min_{\mathbf{x}} \|\Theta\mathbf{V}\mathbf{x}-\Theta\mathbf{b}\|$.

A theoretical guarantee is given by a subspace embedding for $\mathbf{V}$ and $\mathbf{b}$. If Θ is an ϵ-embedding for the space $\text{range}(\mathbf{V})+\text{span}(\mathbf{b})$, then the solution $\mathbf{x}_{\text{sketch}}$ to the sketched system is quasi-optimal:

$$\|\mathbf{V}\mathbf{x}_{\text{sketch}}-\mathbf{b}\| \ \le \ (1+\epsilon)\min_{\mathbf{x}} \|\mathbf{V}\mathbf{x}-\mathbf{b}\|.$$

7.2 Oblivious Subspace Embeddings (OSE)

A random matrix $\Theta\in\mathbb{R}^{\ell\times n}$ is called an oblivious subspace embedding (OSE) for dimension d if, for any d-dimensional subspace $V\subset\mathbb{R}^n$,

$$\Pr\big[\Theta \text{ is an } \epsilon\text{-embedding for } V\big] \ \ge \ 1-\delta.$$

This means that, with high probability, Θ preserves the pairwise inner products of any vectors in V up to a factor of $(1\pm\epsilon)$.

Examples of OSEs

- **Gaussian matrices:** A matrix Θ whose entries are sampled i.i.d. from a suitable normal distribution can be shown (via concentration inequalities) to embed all d-dimensional subspaces with high probability if $\ell\gtrsim\frac{d}{\epsilon^2}$ (plus logarithmic factors).

- **Subsampled Random Hadamard Transform (SRHT):**

$$\Theta \;=\; \sqrt{\frac{1}{\ell}}\,(SH_nD),$$

where D is a random diagonal sign-flip (i.e. ± 1 on the diagonal), H_n is the $n \times n$ Hadamard matrix, and S is a random sampling (subselection) of rows. SRHT also achieves $\ell = \mathcal{O}(d \operatorname{polylog}(n))$.

If Θ satisfies the Johnson–Lindenstrauss property for all vectors in $\mathbb{R}^n$ with distortion ϵ^*, then for any d-dimensional subspace V, Θ also serves as a $(c_1\epsilon^*,\, c_2^d\delta)$ OSE. The main takeaway is that many JL constructions imply OSEs for low-dimensional subspaces.

7.3 Restricted Isometry Property (RIP)

A matrix $\Theta \in \mathbb{R}^{\ell \times n}$ satisfies the (t, ϵ)-RIP if for every t-sparse vector $x \in \mathbb{R}^n$,

$$(1-\epsilon)\,\|x\|_2^2 \;\le\; \|\Theta x\|_2^2 \;\le\; (1+\epsilon)\,\|x\|_2^2.$$

RIP is central in compressed sensing, ensuring recovery of sparse signals from a small number of measurements.

An (ϵ, δ, t)-OSE satisfies the (t, ϵ)-RIP with probability at least $1 - \binom{n}{t}\,\delta$. This implies that sketching matrices preserving entire t-dimensional subspaces also preserve norms of t-sparse vectors.

Consider the ℓ_0-constrained least squares:

$$\min_{\|x\|_0 \le t} \|Vx - b\|, \quad t \le d \ll n.$$

Using a Gaussian embedding of size $l = \mathcal{O}\big(t \log(n) + \log(1/\delta)\big)$, we can obtain a quasi-optimal solution via sketching (since RIP ensures stable embeddings of t-sparse vectors).

7.4 Illustrations

Orthogonalization:
When we have a tall matrix $V \in \mathbb{R}^{n \times d}$, we can orthogonalize its columns via **QR** decomposition. Two classical approaches:

- *Classical (or Modified) Gram–Schmidt*:

$$\begin{aligned} q_1 \;&=\; \frac{v_1}{\|v_1\|}, \\ r_{ij} \;&=\; q_i^{\mathsf{T}} v_j, \quad v_j \;:=\; v_j - r_{ij}\, q_i \quad (\text{for } i = 1, \ldots, j-1), \\ q_j \;&=\; \frac{v_j}{\|v_j\|}. \end{aligned}$$

This method is robust but not always the fastest on modern architectures.

- *Cholesky QR*:
$$R \;=\; \mathrm{chol}(V^{\mathsf{T}}V), \qquad Q \;=\; V\,R^{-1}.$$
Faster in practice (fewer memory references, better cache efficiency) but somewhat less numerically stable for extremely ill-conditioned matrices.

Subspace Approximation:
We want to approximate the column space of a matrix $A \in \mathbb{R}^{n\times n}$ using a low-dimensional subspace V. A common approach is:
$$V \;=\; \mathrm{range}(A\,\Omega),$$
for a random matrix Ω. By multiplying A with Ω (and then orthogonalizing), we obtain a basis for an approximate range of A. This is often more efficient than forming V via repeated applications of A (depending on hardware and memory patterns).

7.5 RandNLA in Practice

Modern computing environments have:

- Multi-core CPUs, GPUs, distributed clusters.
- Costs dominated by communication (memory bandwidth, latency, message passing), not just raw floating-point operations.

Hence, an efficient RandNLA algorithm should:

- Minimize passes over the large matrix.
- Exploit parallelism (e.g. using OpenMP, MPI, GPU acceleration).
- Use optimized BLAS/LAPACK kernels for linear algebra operations.

Streaming model:
Data matrices may be updated incrementally:
$$V_{t+1} \;=\; \begin{bmatrix} V_t \\ \\ \Delta V \end{bmatrix} \quad \text{or} \quad V_{t+1} \;=\; V_t + \Delta V, \quad \ldots$$

We want to maintain a sketch $S_t = \Theta V_t$ without recomputing from scratch. For example,
$$S_{t+1} \;=\; \Theta\,V_{t+1} \;=\; \Theta\,\big(V_t + \Delta V\big) \;=\; S_t \;+\; \Theta\,\Delta V.$$

This approach avoids storing the entire matrix.

7.5.1 BLAS & LAPACK

- *BLAS (Basic Linear Algebra Subprograms):* standardized low-level routines.
- *LAPACK:* higher-level algorithms for linear systems, eigenvalue problems, SVD, built atop BLAS.

RandBLAS aims to:

- Provide high-performance kernels for random embedding and subspace sampling.
- Integrate smoothly with existing BLAS/LAPACK pipelines.
- Support both dense and sparse embeddings.

RandLAPACK builds on RandBLAS to offer end-to-end RandNLA routines:

- Sketched least squares and optimization.
- Low-rank approximation and matrix factorizations.
- Eigenvalue and singular value computations.
- Iterative methods and PDE solvers with random preconditioning.

8 Thursday, February 13th (Scribe: Yuxi Liu)

Recall that we want to solve

$$Ax \approx b$$

where $A \in \mathbb{R}^{n \times d}$ with $n \gg d$. Because the system is overconstrained, an exact solution generally does not exist; instead, we compute an approximate solution that minimizes the residual.

When we sketch the problem using a matrix (denoted by S or Π), we wish the sketch to preserve key properties of A and the LS problem. Two conditions are crucial:

If we let U_A be an orthonormal basis for the columns of A (e.g., from an SVD or QR decomposition), then a sufficient condition is

$$\sigma_{\min}(SA)^2 \geq \frac{1}{2} \quad \Longleftrightarrow \quad \sigma_{\min}(SU_A) \geq \frac{1}{\sqrt{2}},$$

which guarantees that the sketch does not distort the geometry of $\text{span}(A)$ too much.

Denote by

$$b^{\perp} = b - Ax_{opt}$$

the component of b orthogonal to $\text{span}(A)$. We require that

$$\|U_A S^T S b^{\perp}\| \leq \epsilon \|b\|$$

(in some versions the bound is expressed relative to $\|b^{\perp}\|$). This condition ensures that the sketch does not amplify the part of b that cannot be represented in $\text{span}(A)$.

Under these conditions, one can show that the LS objective is approximated up to a factor of $(1 \pm \epsilon)$:

$$\|A\hat{x} - b\| \leq (1 + \epsilon)\|b^{\perp}\|.$$

The error in the solution vector satisfies (up to constants that may involve the condition number $\kappa(A)$ and a parameter γ measuring how much of b lies in $\text{span}(A)$):

$$\|\tilde{x}_{opt} - x_{opt}\| \leq \sqrt{\epsilon}\, \kappa(A) \sqrt{\gamma^{-2} - 1}\, \|x_{opt}\|.$$

Before discussing fast implementations, let's outline two baseline (or "slow") randomized algorithms. Both yield a good approximation but do not yet improve the asymptotic running time over traditional $O(nd^2)$ methods. We will take both and make both fast.

We are given an overconstrained system $Ax \approx b$ with $A \in \mathbb{R}^{n \times d}$ and $n > d$. We let $U \in \mathbb{R}^{n \times d}$ be an orthonormal basis for the columns of A (so $U^T U = I_d$). We want to construct a "good" projection $\Pi \in \mathbb{R}^{r \times n}$ such that

$$\|I_d - U^T \Pi^T \Pi U\|_2 < \epsilon,$$

and such that computing Πx takes time $O\left(nd \ln r\right)$.

Algorithm 1 Slow Sampling

procedure SLOWSAMPLING(A, b)
    Obtain an orthonormal basis U_A for span(A) (via QR or SVD).
    For each row i, compute its leverage score:

$$p_i = \frac{1}{d}\|U_{A,(i)}\|_2^2.$$

    Randomly sample

$$r \gtrsim O\left(\frac{d \ln d}{\epsilon}\right)$$

rows of A (and corresponding entries of b) according to the probabilities $\{p_i\}$.
    Rescale each sampled row by an appropriate factor (typically $1/\sqrt{rp_i}$).
    Form the sketched problem $SA\,x \approx Sb$
    return $\hat{x}_{opt} = (SA)^+ Sb$
end procedure

Algorithm 2 Slow Projection

procedure SLOWPROJECTION(A, b)
    Let $\Pi \in \mathbb{R}^{r \times n}$ be a dense random matrix with i.i.d. entries (e.g., drawn from $\mathcal{N}(0,1)$ or $\{\pm 1\}$).
    Compute the projected matrix ΠA and vector Πb.
    return $\hat{x}_{opt} = (\Pi A)^+ \Pi b$
end procedure

8.1 Fast Johnson–Lindenstrauss Embedding

Definition 1 (ϵ-FJLT Embedding)**.** A matrix $\Pi : \mathbb{R}^n \to \mathbb{R}^r$ is an ϵ-FJLT embedding (Fast Johnson–Lindenstrauss Transform) if:

1. It is a usual Johnson–Lindenstrauss embedding (i.e., it approximately preserves norms of vectors in a certain subspace up to $(1 \pm \epsilon)$ distortions).
2. It can be applied quickly in $O(n \log n)$ or related sublinear time via structured transforms (e.g., Hadamard-based).

Definition 2 (Hadamard Matrices)**.** Let $\tilde{H}_{2^m}$ be the $2^m \times 2^m$ unnormalized Hadamard matrix defined recursively by

$$\tilde{H}_{2^m} = \begin{bmatrix} & \\ 1 & 1 \\ & \\ 1 & -1 \end{bmatrix}^{\otimes m}$$

The normalized Hadamard matrix H_n is given by $H_n := \frac{1}{\sqrt{n}} \tilde{H}_n$, where $n = 2^m$. By construction, H_n is orthogonal: $H_n H_n^T = I_n$.

Example 1 (Fast Embedding $\Pi = PHD$)**.** A typical structured construction of a fast JL transform is:

$$\Pi = P\,H\,D,$$

where

- $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose diagonal entries are random ± 1 .
- $H \in \mathbb{R}^{n \times n}$ is a (normalized) Hadamard matrix.
- $P \in \mathbb{R}^{r \times n}$ is a sparse random projection or sampling matrix. Concretely, each entry P_{ij} is 0 with probability $1 - q$ and a Gaussian random variable $\mathcal{N}(0, 1/q)$ with probability q . (Other structures for P are also common.)

We have the following useful properties of fast embedding

1. Multiplying a vector x by HD tends to "spread" the coordinates of x uniformly, i.e., for a unit vector x, most entries of HDx become $O\left(1/\sqrt{n}\right)$ with high probability.
2. We can multiply HDx in $O(n \ln n)$ time using the FFT-like fast Hadamard transform (since H is a special case of the Fourier transform on a group of size $n = 2^m$).

Hence, $\Pi x = P(HD)x$ can be computed quickly. This construction is an ϵ-FJLT if r is chosen appropriately in terms of d, ϵ, and $\ln n$.

Lemma 1 (Bound on $\|HDU\|_2$)**.** Let $U \in \mathbb{R}^{n \times d}$ have orthonormal columns ($U^T U = I_d$), and let

- $H \in \mathbb{R}^{n \times n}$ be a normalized Hadamard matrix,
- $D \in \mathbb{R}^{n \times n}$ be a diagonal $\{\pm 1\}$ random matrix.

Then, with probability at least $1 - \frac{1}{20}$,

$$\|HDU\|_2^2 \leq \frac{2\,d}{n} \ln(40\,n).$$

Proof. 1. **Setup random variables.** Fix indices i and j. Note that

$$(HDU)_{ij} = \sum_{\ell=1}^{n} H_{i\ell}\, D_{\ell\ell}\, U_{\ell j} = \sum_{\ell=1}^{n} X_\ell,$$

where we define

$$X_\ell := H_{i\ell}\, D_{\ell\ell}\, U_{\ell j}.$$

Each $D_{\ell\ell}$ is a random sign (± 1) and is independent of $H_{i\ell}$, $U_{\ell j}$. By symmetry, $\mathbb{E}[X_\ell] = 0$.

2. **Bound each term ($|X_\ell|$).** Since H is a normalized Hadamard matrix, each $|H_{i\ell}| = \frac{1}{\sqrt{n}}$. Thus,

$$|X_\ell| = |\, H_{i\ell}\, D_{\ell\ell}\, U_{\ell j}| \le \frac{1}{\sqrt{n}}\ |\, U_{\ell j}|\,.$$

In particular, $\sum_{\ell=1}^{n} X_\ell$ has bounded summands, which is key for a Chernoff- or Hoeffding-type inequality.

3. **Apply a Chernoff/Hoeffding bound.** We want:

$$\Pr\left(|\,(HDU)_{ij}| > n\,t\right) = \Pr\left(\left|\sum_{\ell=1}^{n} X_\ell\right| > n\,t\right).$$

A standard bound (Hoeffding) gives, for independent mean-zero X_ℓ with $|X_\ell| \le \alpha_\ell$:

$$\Pr\left(\left|\sum_{\ell=1}^{n} X_\ell\right| > n\,t\right) \le 2\,\exp\!\left(-\frac{2\,n^2\,t^2}{\sum_{\ell=1}^{n}\alpha_\ell^2}\right).$$

Here, $\alpha_\ell = \frac{1}{\sqrt{n}}\,|U_{\ell j}|$. Thus

$$\sum_{\ell=1}^{n}\alpha_\ell^2 = \sum_{\ell=1}^{n}\frac{1}{n}\ |U_{\ell j}|^2 = \frac{1}{n}\sum_{\ell=1}^{n}|U_{\ell j}|^2\,.$$

Since $\sum_{\ell=1}^{n}|U_{\ell j}|^2 = \|U_{(\cdot j)}\|_2^2 \le 1$, we get

$$\frac{1}{n}\sum_{\ell=1}^{n}|U_{\ell j}|^2 \le \frac{1}{n}.$$

Substituting back:

$$\Pr\left(\left|\sum_{\ell=1}^{n} X_\ell\right| > n\,t\right) \le 2\,\exp\!\left(-\frac{2\,n^2\,t^2}{\frac{1}{n}\sum_\ell U_{\ell j}^2}\right) \le 2\,\exp\!\big(-2\,n^3\,t^2\big)\,.$$

Setting $\delta = 2\,e^{-2n^3t^2}$ yields

$$\Pr\left(|\,(HDU)_{ij}| > n\,t\right) \le \delta \implies t = \sqrt{\frac{\ln\!\left(\frac{2}{\delta}\right)}{2\,n^3}}.$$

4. **Rewrite the probability bound in a more direct form.** From $\delta = 2\,e^{-2n^3t^2}$, solve for t to get

$$\Pr\left(|\,(HDU)_{ij}| > \sqrt{\frac{2\,\ln(\frac{2}{\delta})}{n}}\right) \le \delta.$$

5. **Apply the union bound.** We want this to hold for all pairs (i, j) with $i \in [n]$ and $j \in [d]$. There are nd such entries. Set $\delta = \frac{1}{nd}$ times a constant factor to get the desired probability of success. Concretely, if we choose $\delta = \frac{1}{20\,nd}$, then with probability at least $1 - \frac{1}{20}$ we have

$$|\,(HDU)_{ij}| \leq \sqrt{\frac{2\,\ln(40\,nd)}{n}} \quad \forall\, i, j.$$

6. **Bound row norms.** Finally,

$$\|(HDU)_{(i)}\|_2^2 = \sum_{j=1}^{d} |\,(HDU)_{ij}|^2 \leq d \cdot \frac{2\,\ln(40\,nd)}{n} = O\left(\frac{d}{n}\,\ln(n)\right).$$

□

8.2 Fast LS via Random Projection

To approximate least squares solutions, we often need two main conditions (sometimes called "Condition 1" and "Condition 2") to hold when replacing A and b by sketched versions SA and Sb.

Definition 3 (Condition 1)**.** $\sigma_{\min}(SA)$ is close to 1, so that SA preserves the column space geometry.

Definition 4 (Condition 2)**.** $\|U_A\, S^T\, Sb^{\perp}\| \lesssim \epsilon\, \|b^{\perp}\|$ is small, so that the portion of b orthogonal to span(A) does not get amplified.

Using FJLT-based projections (like $\Pi = PHD$), one can prove:

Lemma 2. If $r \gtrsim d\,\ln(nd)\,\ln(d)$, then with constant probability, Condition 1 is satisfied.

Lemma 3. If $r \gtrsim \frac{\ln(nd)}{\epsilon}$, then with constant probability, Condition 2 is satisfied.

Hence, by taking r on the order of $d \ln d \cdot \ln(n)$ (and possibly $\frac{1}{\epsilon}$ factors for Condition 2), we can ensure a relative-error approximation to the LS problem.

Theorem 1 (Fast LS Projection)**.** If r is set to $O\left(d \ln d + \frac{d \ln n}{\epsilon}\right)$, then with high probability,

- $\|A\hat{x}_{\text{opt}} - b\| \;\leq\; (1+\epsilon)\,\|b^{\perp}\|$,
- $\|\hat{x}_{\text{opt}} - x_{\text{opt}}\| \;\leq\; \sqrt{\epsilon}\,\kappa(A)\,\sqrt{\frac{1}{\gamma^2} - 1}\;\|x_{\text{opt}}\|$,

and the total running time is

$$O\left(n\,d\,\ln(\tfrac{d}{\epsilon}) \;+\; d^3\,(???)\right).$$

8.3 Fast Leverage Score Sampling

Recall the leverage scores (for $n > d$) are $\ell_i = \|U_{(i)}\|_2^2$, the squared row norms of any orthonormal basis U spanning span(A). A direct QR or SVD of A costs $O(nd^2)$, so we want a faster approach.

Algorithm 3 Fast LS via Random Projection

1: **input** $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$, error parameter $\epsilon > 0$.
2: Construct $\Pi = PHD$ such that
- D is diagonal with random ± 1. H is an $n \times n$ normalized Hadamard matrix.
- $P \in \mathbb{R}^{r\times n}$ is a (typically sparse) sampling/projection matrix, with

$$r \;\gtrsim\; O\Big(\tfrac{d\,\ln(d)}{\epsilon}\Big).$$

3: Compute (ΠA) and (Πb): This takes $O(nd \ln n)$ time (for H and D) plus $O(rn)$ for the sampling/combination in P.
4: Solve the sketched LS problem:

$$\hat{x}_{\text{opt}} \;=\; (\Pi A)^{+}\,\Pi b.$$

5: **return** $\hat{x}_{\text{opt}}$.

We can approximate ℓ_i by

$$\ell_i \;=\; \|e_i\,A\,A^{+}\|_2^2 \;\approx\; \|e_i\,A\,(\Pi_1 A)^{+}\|_2^2 \;\approx\; \|e_i\,A\,(\Pi_1 A)^{+}\,\Pi_2\|_2^2$$

where

- $\Pi_1 \in \mathbb{R}^{r_1\times n}$ is a **fast** projection (FJLT) with $r_1 \approx d\log(d)$. Then, $\Pi_1 A$ is $r_1 \times d$, so its pseudoinverse $(\Pi_1 A)^{+}$ is only $d \times r_1$.
- $\Pi_2 \in \mathbb{R}^{r_1\times r_2}$ (with $r_2 = O(\log n)$) is another small JL transform to reduce the cost of multiplying A by $(\Pi_1 A)^{+}$.

Hence, we form

$$\Omega \;=\; A\,R^{-1}\,\Pi_2,$$

where R is the $d\times d$ factor from $\Pi_1 A = QR$. Each row of Ω approximates the row of UA -projection we need to estimate leverage. Let

$$\tilde{\ell}_i \;=\; \|\Omega_{(i)}\|_2^2.$$

Under suitable conditions on Π_1, Π_2, one proves

$$\Big|\ell_i - \tilde{\ell}_i\Big| \;\le\; \epsilon\,\ell_i, \quad \forall\, i.$$

Algorithm 4 Fast leverage score sampling

1: Sketch A to a small matrix $\Pi_1 A \in \mathbb{R}^{r_1\times d}$ via an FJLT Π_1.
2: Compute $(\Pi_1 A)^{+}$ in $O(r_1\,d^2)$.
3: Form $A(\Pi_1 A)^{+}$. Naively, this might be $O(ndr_1)$, still large.
4: Apply a second random embedding Π_2 to reduce the cost of explicitly multiplying out all rows.
5: Recover row norms $\|\Omega_{(i)}\|_2^2$ with $\Omega = A(\Pi_1 A)^{+}\Pi_2$. This approximates each leverage score ℓ_i up to $\epsilon\ell_i$ relative error.

Thus, the row norms $\tilde{\ell}_i$ suffice for sampling. The running time becomes $O\left(nd\log(n) + d^3\log(n)\right)$, which can be lower than nd^2 for certain regimes of n and d.

The following lemma shows that the leverage score approximation is probably approximately correct. The proof is in the next lecture. It will involve doing something that feels like an SVD, though not literally so.

Lemma 4. Let

$$U_i := e_i\, A\, A^{+}, \quad \tilde{U}_i := e_i\, A\, (\Pi_1 A)^{+}, \quad \tilde{\tilde{U}}_i := e_i\, A\, (\Pi_1 A)^{+}\, \Pi_2$$

- If $\left| U_i^T U_j - \tilde{U}_i^T \tilde{U}_j \right| \;\leq\; \frac{\epsilon}{1-\epsilon}\, \|U_i\|\, \|U_j\|$, then

$$|l_i - \tilde{l}_i| \leq \frac{\epsilon}{1-\epsilon} l_i$$

- If $\left| \tilde{U}_i^T \tilde{U}_j - \tilde{\tilde{U}}_i^T \tilde{\tilde{U}}_j \right| \;\leq\; 2\,\epsilon\, \|\tilde{U}_i\| \|\tilde{U}_j\|$, then

$$|l_i - \tilde{\tilde{l}}_i| \leq 2\epsilon \tilde{l}_i$$

$$|l_i - \tilde{\tilde{l}}_i| \leq |l_i - \tilde{l}_i| + |\tilde{l}_i - \tilde{\tilde{l}}_i| < 4\epsilon l_i$$

Hence, $\|\tilde{\tilde{U}}_i\|^2$ approximates the actual leverage score ℓ_i within a factor of $(1 \pm O(\epsilon))$.

Algorithm 5 Fast LS by leverage score sampling

1: Use the fast approximate leverage score procedure to get $\tilde{\ell}_i \approx \ell_i$.
2: Compute the sampling probabilities by $p_i = \tilde{\ell}_i / \sum_j \tilde{\ell}_j$.
3: Sample r rows IID, with probability p_i. Each sampled row is rescaled by $\frac{1}{\sqrt{r\, p_i}}$.
4: Solve the sketched system $\min_x \|SA\, x - S\, b\|$.

Similarly to before, this gives a $(1 \pm \epsilon)$ approximation in roughly $O(nd \log(r))$ time.

9 Tuesday, February 18th (Scribe: Songlin Zhao)

Typo on HW2 Question 3:

$$||A - CC^{+}||_2^2 = ||AA^T - CC^T||_2$$

Hint:

$$\begin{aligned} CC^{+} &= U_c \Sigma_c V_c^{\top} (U_c \Sigma_c V_c^{\top})^{+} && (1) \\ &= U_c \Sigma_c V_c^{\top} V_c \Sigma_c^{-1} U_c^{\top} && (2) \\ &= U_c U_c^{\top} = P_c && (3) \end{aligned}$$

9.1 Fast Random Projection and FJLT Continued

We continue proving Lemma 4 from last time and the recall notations defined from previous section:

Let $A \in \mathbb{R}^{n \times d}$ with $n >> d$, $\Pi_1 \in \mathbb{R}^{r_1 \times n}$ belongs to FJLT-based projections, $\Pi_2 \in \mathbb{R}^{r_1 \times r_2}$ belongs to JLT-based projections. We will show that

$$\begin{aligned} \ell_i &= \left\| e_i A A^{\dagger} \right\|_2^2 \\ &\approx \left\| e_i A \left(\Pi_1 A \right)^{\dagger} \right\|_2^2 = \hat{\ell}_i \\ &\approx \left\| e_i A \left(\Pi_1 A \right)^{\dagger} \Pi_2 \right\|_2^2 = \tilde{\ell}_i. \end{aligned} \tag{4}$$

The main theorem that we can establish for FastApproximateLeverageScores

Theorem 13 *The **FastApproximateLeverageScores** algorithm returns $\tilde{\ell}_{(i)}$ such that*

$$\left| \ell_i - \tilde{\ell}_i \right| \leq \epsilon \ell_i,$$

for all $i \in [n]$, with constant probability, in time

$$O\left(nd \log\left(d/\epsilon \right) + nd\epsilon^{-2} \log(n) \log\left(d\epsilon^{-1} \right) \right)$$

(assuming that $d \leq n$).

Proof:

Given $U_i = e_i A (\Pi_1 A^{\top})^{\dagger}$ and $\tilde{U}_i = e_i A (\Pi_2 A)^{\dagger}$

$$
\begin{aligned}
\tilde{U}_i^\top \tilde{U}_j &= e_i A(\Pi_1 A)^\dagger((\Pi_1 A)^\dagger)^\top A^\top e_j = e_i U\Sigma V^\top(\Pi U\Sigma V^\top)^\dagger((\Pi U\Sigma V^\top)^\dagger)^\top V\Sigma U^\top e_j \\
&= e_i U\Sigma V^\top V\Sigma^{-1}(\Pi U)^\dagger((\Pi U)^\dagger)^\top\Sigma^{-1}V^\top V\Sigma U^\top e_j \\
&= e_i U(\Pi U)^\dagger((\Pi U)^\dagger)^\top U^\top e_j \\
&= e_i U(V_{\Pi U}\Sigma_{\Pi U}^{-1}U_{\Pi U}^\top)(U_{\Pi U}\Sigma_{\Pi U}^{-1}V_{\Pi U}^\top)U^\top e_j \\
&= e_i U V_{\Pi U}\Sigma_{\Pi U}^{-2}V_{\Pi U}^\top U^\top e_j; \\
|U_i^\top U_j - \tilde{U}_i^\top \tilde{U}_j| &= |e_i U(I - V_{\Pi U}\Sigma_{\Pi U}^{-2}V_{\Pi U}^\top)U^\top e_j| \\
&\leq \|I - V_{\Pi U}\Sigma_{\Pi U}^{-2}V_{\Pi U}^\top\|_2\|U_i\|_2\|U_j\|_2 \\
&\leq \|I - \Sigma_{\Pi U}^{-2}\|_2\|U_i\|_2\|U_j\|_2 \quad \leq \epsilon\|U_i\|_2\|U_j\|_2 \\
&\leq \epsilon\|U_i\|_2^2 \quad \text{if } i = j
\end{aligned}
$$

9.2 Randomized Least Square in Practice

Now we proceed to Randomized Least Square in Practice:

Issues:

- Forward vs Backward Error
- Condition Number
- ϵ - dependence
- δ - failure probability
- hybrid / precondition
- downsample more aggressively

Definition 15: A problem P is well-posed if: the solution exists; the solution is unique; and the solution depends continuously on the input data in some reasonable topology.

For evaluation of numerical stability of an algorithm, we introduce some new concepts. Let's consider an algorithm as a function f attempting to map input data X to output data Y ; but, due to roundoff errors, random sampling, or whatever, the algorithm actually maps input data X to output data Y. That is, the algorithm "should" return Y , but it actually returns Y^* . In this case, we have the following.

- **Forward Error:** This is $\Delta Y = Y - Y^*$, and so this is the difference between the exact/true answer that was output by the algorithm. (This is typically what we want to bound, although note that one might also be interested in some function of it, like the objective function value in an optimization, rather than the argmin.)
- **Backward error.** This is the smallest ΔX such that $f(X + \Delta X) = Y$, and so this measures what input data the algorithm that we ran actually solved exactly.

In general, the forward error and backward error are related by a problem-specific complexity measure, often called the condition number as follows:

$$
|\text{forward error}| \leq |\text{condition number}| \times |\text{backward error}| \tag{5}
$$

In particular, backward stable algorithm provide accurate solutions to well-conditioned problems. TCS typically bounds the forward error directly in one step, while NLA bounds the forward error indirectly in two steps.

Theoretical Computer Science Perspective: In light of this discussion, observe that the bounds that we proved the a few classes ago bound the forward error in the TCS style.

forward error in the TCS style. In particular, recall that the bounds on the objective are of the form

$$\|A\tilde{x}_{opt} - b\|_2 \leq (1+\epsilon)\|Ax_{opt} - b\|_2$$

and this implies that

$$\|\tilde{x}_{opt} - x_{opt}\|_2 \leq \frac{\gamma}{\cos\theta}\epsilon\|x_{opt}\|_2 = \tan(\theta)\,\kappa(A)\sqrt{\epsilon}\|x_{opt}\|_2, \quad (65)$$

where $\theta = \cos^{-1}\left(\frac{\|Ax_{opt}\|_2}{\|b\|_2}\right)$ is the angle between the vector b and the column space of A.

This is very different than the usual stability analysis.

Numerical Linear Algebra Perspective:

In particular, recall that the bounds on the objective are of the form

$$\|\tilde{A}\tilde{x}_{opt} - b\|_2 \leq (1+\epsilon)\|Ax_{opt} - b\|_2$$

and this implies that

$$\tilde{x}_{opt} = \arg\min \|(A+\delta A)x - b\|_2, \quad (66)$$
$$\text{where } \|\delta A\|_\xi \leq \tilde{\epsilon}\|A\|_\xi.$$

(We could of course include a perturbed version of b in Equation (66).) By standard NLA methods, Equation (66) implies a bound on the forward error

$$\|\tilde{x}_{opt} - x_{opt}\|_2 \leq \left(\kappa(A) + \frac{\kappa^2(A)\tan(\theta)}{\eta}\right)\tilde{\epsilon}\|x_{opt}\|_2, \quad (67)$$
$$\text{where } \eta = \frac{\|A\|_2\|x\|_2}{\|Ax\|_2}.$$

Importantly, Equation (65) and Equation (67), i.e., the two different forward error bounds on the vector or certificate achieving the optimal solution, are not obviously comparable.

9.3 Methods for Solving Least Squares and Linear Systems

Several approaches exist to solve least squares (LS) problems or linear systems:

- **Exact Methods**: Direct approaches to solve linear systems or least squares problems, typically involving factorizations (e.g., QR, SVD).
- **Iterative Methods**: Techniques such as gradient descent or conjugate gradient, useful for large-scale problems where direct methods are impractical.

- **Preconditioned Methods**: Methods that accelerate convergence of iterative solvers by transforming the problem into a better-conditioned form, often using preconditioners. $Ax = b \Rightarrow M^{-1}Ax \simeq M^{-1}b$
- **Randomized Algorithms**: Practical randomized approaches to efficiently construct preconditioners or directly solve preconditioned linear systems, significantly speeding up computations for large datasets.
- **Randomized Algorithms for Least Squares**: Algorithms specifically designed to leverage randomness to quickly approximate solutions, often used either to directly solve problems or as preconditioners to enhance the efficiency of iterative solvers.

Definition 18 *Let $A \in \mathbb{R}^{n\times d}$ be a full rank matrix, with $n > d$, and let $U \in \mathbb{R}^{n\times d}$ be an orthogonal matrix for span(A). Then, if $U_{(i)}$ is the i^{th} row of U, the coherence of A is*

$$\mu(A) = \max_{i\in[n]} \|U_{(i)}\|_2^2.$$

That is, the coherence is—up to a scaling that isn't standardized in the literature—equal to the largest leverage score. Equivalently, up to the same scaling, it equals the largest diagonal element of $P_A = A\left(A^TA\right)^\dagger A^T$, the projection matrix onto the column span of A. Defined this way, i.e., not normalized to be a probability distribution, possible values for the coherence are

$$\frac{d}{n} \le \mu(A) \le 1.$$

Thus, with this normalization:

- If $\mu(A) \gtrsim \frac{d}{n}$, then the coherence is small, and all the leverage scores are roughly uniform.
- If $\mu(A) \lesssim 1$, then the coherence is large, and the leverage score distribution is very nonuniform (in that there is at least one very large leverage score).

The following result (which is parameterized for a uniform sampling process) describes the relationship between the coherence $\mu(A)$, the sample size r, and the condition number of the preconditioned system.

Lemma 18. Let A be defined as above, consider a uniform sampling operator $S \in \mathbb{R}^{r\times n}$. Define

$$\tau = C\sqrt{\frac{n\mu(A)\log(r)}{r}},$$

where C is an absolute constant. Suppose $\delta^{-1}\tau < 1$, where δ is the failure probability. Then, with probability at least $1-\delta$, we have

$$\text{rank}(SA) = d,$$

and if the QR decomposition of SA is $SA = \tilde{Q}\tilde{R}$, it follows that

$$\kappa\left(A\tilde{R}^{-1}\right) \le \frac{1+\delta^{-1}\tau}{1-\delta^{-1}\tau}.$$

Poof.

We first proceed by proving the claim $\kappa(SU) = \kappa(AR^{-1})$.

$$
\begin{aligned}
SU &= U_{SU}\Sigma_{SU}V_{SU}^{\top} \quad \text{by definition} \\
SA &= \tilde{Q}\tilde{R} \quad \text{by definition} \\
U_{SU} &= \tilde{Q}W, \quad \text{for a } d \times d \text{ unitary matrix } W, \text{ since they span the same space.}
\end{aligned}
$$

In this case, it follows that

$$
\begin{aligned}
\tilde{R} &= \tilde{Q}^{\top}SA \\
&= \tilde{Q}^{\top}SU\Sigma V^{\top} \\
&= \tilde{Q}^{\top}U_{SU}\Sigma_{SU}V_{SU}^{\top}\Sigma V^{\top} \\
&= W\Sigma_{SU}V_{SU}^{\top}\Sigma V^{\top}.
\end{aligned}
$$

From this (and since Σ_{SU} is invertible, by the approximate matrix multiplication bound, since we have sampled sufficiently many columns), it follows that

$$
\begin{aligned}
A\tilde{R}^{-1} &= UV\Sigma V^{\top}V\Sigma^{-1}V_{SU}\Sigma_{SU}^{-1}W^{\top} \\
&= UV_{SU}\Sigma_{SU}^{-1}W^{\top}.
\end{aligned}
$$

From this, it follows that

$$
\begin{aligned}
\left\| \left(A\tilde{R}^{-1}\right)^{\top} A\tilde{R}^{-1} \right\|_2 &= \left\| W\Sigma_{SU}^{-1}V_{SU}^{\top}U^{\top}UV_{SU}\Sigma_{SU}^{-1}W^{\top} \right\|_2 \\
&= \left\| W\Sigma_{SU}^{-2}W^{\top} \right\|_2 \\
&= \left\| \Sigma_{SU}^{-2} \right\|_2 \\
&= \left\| \Sigma_{SU}^{-1} \right\|_2^2,
\end{aligned}
$$

where the penultimate equality follows since W is orthogonal and the last equality follows since Σ_{SU} is diagonal.

Similarly,

$$
\left\| \left(\left(A\tilde{R}^{-1}\right)^{\top} \left(A\tilde{R}^{-1}\right) \right)^{-1} \right\|_2 = \|\Sigma_{SU}\|_2^2.
$$

This, using that $\kappa(\alpha) = \left(\|\alpha^{\top}\alpha\|_2 \| \left(\alpha^{\top}\alpha\right)^{-1} \|_2 \right)^{1/2}$ for a matrix α, it follows that

$$
\begin{aligned}
\kappa\left(A\tilde{R}^{-1}\right) &= \left(\left\| \left(A\tilde{R}^{-1}\right)^{\top} A\tilde{R}^{-1} \right\|_2 \left\| \left(\left(A\tilde{R}^{-1}\right)^{\top} A\tilde{R}^{-1} \right)^{-1} \right\|_2 \right)^{1/2} \\
&= \left(\|\Sigma_{SU}^{-1}\|_2^2 \|\Sigma_{SU}\|_2^2 \right)^{1/2} \\
&= \|\Sigma_{SU}^{-1}\|_2 \|\Sigma_{SU}\|_2 \\
&= \kappa(SU).
\end{aligned}
$$

10 Tuesday, February 20th (Scribe: Songlin Zhao)

We will continue to finish the proof for lemma 18.

10.1 Continuation of proof of lemma

Recall that

$$\mathbb{E}\big[\|I - U^\top S^\top S\, U\|_2\big] \;\leq\; \tau.$$

By Markov's Inequality, it follows that

$$\Pr\big[\|I - U^\top S^\top S\, U\|_2 \;>\; \delta^{-1}\tau\big] \;\leq\; \delta.$$

Thus, with probability at least $1 - \delta$, we have

$$\|I - U^\top S^\top S\, U\|_2 \;<\; \delta^{-1}\tau \;<\; 1,$$

Next, recall that every eigenvalue λ of $U^\top S^\top SU$ is the Rayleigh quotient of some vector $x \neq 0$, i.e.,

$$\lambda = \frac{x^\top U^\top S^\top SU\, x}{x^\top x} = \frac{x^\top \big(U^\top S^\top SU - I\big)\, x}{x^\top x} = 1 + \eta,$$

where η is the Rayleigh quotient of $I - U^\top S^\top SU$.

Since this is a symmetric matrix, its singular values are the absolute eigenvalues. Thus, $|\eta| < \delta^{-1}$. Therefore, all the eigenvalues of $U^\top S^\top SU$ lie in the interval $\big[\,1 - \delta^{-1},\, 1 + \delta^{-1}\big]$. Hence,

$$\kappa(SU) \;\leq\; \sqrt{\frac{1 + \delta^{-1}}{1 - \delta^{-1}}}.$$

10.2 Types of guarantees

Deterministic Running Time and Probabilistic Error Guarantees: We parametrize the problem such that we guarantee that we take not more than $\mathcal{O}(f(n, \epsilon, \delta))$ time, where $f(n, \epsilon, \delta)$ is some function of the size n of the problem, the error parameter ϵ, and a parameter δ specifying the probability with which the algorithm may completely fail. This is most common in TCS, where simpler analysis for worst-case input is of interest.

Probabilistic Running Time and Deterministic Error: NLA and other areas concerned with providing implementations. This involves making a statement of the form

$$\Pr\big[\text{Number of FLOPS for } \leq \epsilon \text{ relative error } > \; f(n, \delta)\big] \;\leq\; \delta.$$

That is, in this case, we can show that we are guaranteed to get the correct answer, but the running time is a random quantity. These algorithms are sometimes known in TCS as Las Vegas algorithms, to distinguish them from Monte Carlo algorithm that have a deterministic running time

but a probabilistic error guarantee. This parameterization/formulation is a particularly convenient when we down sample more aggressively than worst-case theory permits, since we can still get a good preconditioner (since a low-rank perturbation of a good preconditioner is still a good preconditioner) and we can often still get good iterative properties due to the way the eigenvalues cluster. In particular, we might need to iterate more, but we won't fail completely.

10.3 Blendenpik

Algorithm 6 The Blendenpik Algorithm

Require: $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$
Ensure: $\tilde{x}_{\text{opt}} \in \mathbb{R}^d$
while No return **do**
  Compute HD and HDb, where HD denotes a randomized Hadamard transform.
  Randomly sample $\gamma\, d/n$ rows from A and b, with $\gamma \approx 2$, and let S be the corresponding sampling-and-rescaling matrix.
  Compute the QR factorization of SHD, so that $SHD = QR$.
  Set $k \leftarrow \text{kestimate}(R)$ ▷ using LAPACK's `DTRCON` routine.
  if $k > 5\,\epsilon_{\text{mach}}$ **then**
    $\tilde{x}_{\text{opt}} \leftarrow \text{LSQR}(A, b, 10^{-14})$
  else if $iter > 3$ **then**
    return LAPACK
  end if
  $iter \leftarrow iter + 1$
end while

Why do we want to use HD transformation ?

- Strong condition number
- Generating Gaussian matrix has high cost

Why do we use QR instead of SVD? We can, depending on the computation of the algorithm and the actual requirement.

We will now talk about a basic algorithm that uses perturbed QR factorization to solve linear LS problems.

Definition 5. Let $S, T \in \mathbb{R}^{n \times n}$. Then $\lambda = \lambda(S, T) \in \mathbb{R}$ is called a *finite generalized eigenvalue* of the matrix pencil (S, T) if there exists a vector $\mathbf{v} \neq 0$ such that

$$S\mathbf{v} \;=\; \lambda T\mathbf{v}, \quad T\mathbf{v} \;\neq\; 0.$$

In addition, ∞ is an *infinite generalized eigenvalue* of (S, T) if there exists a $\mathbf{v} \neq 0$ such that

$$T\mathbf{v} \;=\; 0, \quad S\mathbf{v} \;\neq\; 0.$$

Note that ∞ is an eigenvalue of (S, T) if and only if 0 is an eigenvalue of (T, S).

Definition 6. The finite and infinite eigenvalues of a matrix pencil (S, T) are called *determined eigenvalues* if the eigenvector uniquely determines the eigenvalue. If $S\mathbf{v} = T\mathbf{v} = 0$ for some $\mathbf{v} \neq 0$,

then $\mathbf{v}$ is an *indeterminate eigenvector*, since $S\mathbf{v} = \lambda T\mathbf{v}$ for all $\lambda \in \mathbb{R}$. We denote the set of determined eigenvalues of (S, T) accordingly.

Definition 7. Let $S, T \in \mathbb{R}^{n\times n}$. The *generalized condition number* is defined by

$$\kappa(S,T) \;=\; \frac{\lambda_{\max}(S,T)}{\lambda_{\min}(S,T)},$$

where $\lambda_{\max}(S,T)$ and $\lambda_{\min}(S,T)$ are the maximum and minimum, respectively, among the determined eigenvalues of (S,T).

Here is a fact. The behavior of preconditioned iterative methods is determined by the clustering of the generalized eigenvectors, and the number of iterations is bounded by a quantity proportional to the generalized condition number. In particular:

- **CG** converges in $\mathcal{O}\big(\sqrt{\kappa(A,M)}\big)$ iterations.
- **LSQR** on a system preconditioned by R converges in $\mathcal{O}\big(\sqrt{\kappa(A^TA,\, R^TR)}\big)$ iterations.

In particular, we describe here a theory (from the ANT paper) that is more general than RandNLA preconditioning but can be applied directly to RandNLA preconditioning. Let A be a matrix, and define

$$\widehat{A} \;=\; \begin{pmatrix} A \\ \\ B \end{pmatrix}.$$

Then

$$(\widehat{A}^T\widehat{A})^{-1}A^TA \;=\; (A^TA{+}B^TB)^{-1}A^TA \;=\; (A^TA{+}B^TB)^{-1}\big(A^TA{+}B^TB{-}B^TB\big) \;=\; I - (A^TA{+}B^TB)^{-1}B^TB.$$

Here is a fact: $\mathrm{rank}(\Omega) \;\leq\; \mathrm{rank}(B)$. Hence, if the matrix B is low-rank, then the matrix $(\widehat{A}^T\widehat{A})^{-1}A^TA$ is a low-rank perturbation of the identity I. This implies that a symmetric rank-k perturbation of the identity I has $\leq k$ non-unit eigenvalues.

The same analysis extends to other types of perturbations:

- To when $\hat{A}$ is singular
- To when rows are removed instead of added
- To when columns are exchanged
- To preconditioners for $\hat{A}$ other than the R factor

They also bound the size of non-unit eigenvalues, which is important when A is rank deficient.

Observe that the generalized spectrum of $(A^\top A, \hat{A}^\top\hat{A})$ is very simple: the penscil has rank(A) eigenvalues that are 1 and the rest are indeterminate.

lemma: If eigenvalues $(A^\top A, M)$ lies in $(\alpha, \beta) \in \mathbb{R}$, the rank(M) - k of the smallest eigenvalues of $(A^\top A + B^\top B, M)$ lies within (α, β).

11 Tuesday, February 25th (Scribe: Jiahai Feng)

11.1 Low rank approximation

Problem:

- Given $A = U\Sigma V^T$, where A is tall and skinny ($m \times n$), U is $m \times n$, Σ is $n \times n$ diagonal, V is $n \times n$.
- We can write outer product form $A = \sum_{i=1}^{n} \sigma_i u_i v_i^T$
- SVD has a nice 'recursive' structure, where the best rank-k approximation is part of the best rank-$k+1$ approximation
- In particular, if we arrange the singular values in descending order, the best rank-k approximation is the first k terms:

$$\begin{aligned} A &= \sum_{i=1}^{k} \sigma_i u_i v_i^T + \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \\ &= U_k \Sigma_k V_k^T + U_{k\perp} \Sigma_{k\perp} V_{k\perp}^T \\ &= A_k + A_{k\perp} \end{aligned}$$

- We can equivalently write:

$$\begin{aligned} A_k &= U_k \Sigma_k V_k^T \\ &= U_k U_k^T A \\ &= A V_k V_k^T \end{aligned}$$

11.2 Computing the SVD, (or a low rank approximation)

- How long does it take?
  - Hard to say, depends on how you set up the problem (e.g. precision, condition number, etc.)
  - Ballpark: $O(n^3)$ for the SVD, but $O(mnk)$ for the rank-k approximation
- Goal: reduce k to $\log k$, and reduce mn to something that depends on the number of matrix-vector multiplications

11.3 Background: Matrix perturbation theory

Suppose we have $A, E \in \mathbb{R}^{m \times n}$, where $m \geq n$.
We want to relate the singular values of $A + E$ to A.

Lemma 5 (Hoffman-Wielandt inequality)**.**

$$\max_{i \in [n]} |\sigma_i(A + E) - \sigma_i(A)| \leq \|E\|_2 \tag{6}$$

Lemma 6.

$$\sum_{i=1}^{n} (\sigma_i(A+E) - \sigma_i(A))^2 \leq \|E\|_F^2 \tag{7}$$

We'll take these claims as given.

11.4 Linear time SVD

- Input $A \in \mathbb{R}^{m \times n}$, k, probabilities $\{p_i\}_{i=1}^n$
- Output $H_k \in \mathbb{R}^{m \times k}$, which are spiritually the best k left singular vectors of A, U_k.

Algorithm 7 Linear Time SVD

1: **for** $t = 1$ **to** c **do**
2: Sample i_t with probability p_{i_t}
3: Set $C_t = A_{i_t}/\sqrt{cp_{i_t}}$
4: **end for**
5: Form matrix $C \in \mathbb{R}^{m \times c}$ from the columns C_t
6: Compute $C^T C$ in $O(mc^2)$ time
7: Compute SVD of $C^T C = Y \Sigma^2 Y^T$ in $O(c^3)$ time
8: Output $H = CY\Sigma_c^{-1}$ in $O(mkc)$ time

Lemma 7.

$$\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k}\|AA^T - CC^T\|_F \tag{8}$$

Since H_k has orthogonal columns, $H_k^T H_k$ is a projection matrix. If H_k is truly U_k, then we should get $\|A - A_k\|_F^2$ exactly.

Proof. Since we have

$$\|X\|_F^2 = \mathrm{Tr}(X^T X), \tag{9}$$

$$\mathrm{Tr}(X+Y) = \mathrm{Tr}(X) + \mathrm{Tr}(Y). \tag{10}$$

Therefore,

$$\begin{aligned}
\|A - H_k H_k^T A\|_F^2 &= \mathrm{Tr}((A - H_k H_k^T A)^T (A - H_k H_k^T A)) \\
&= \mathrm{Tr}(A^T A) - 2\,\mathrm{Tr}(A^T H_k H_k^T A) + \mathrm{Tr}(H_k H_k^T A^T A H_k H_k^T) \\
&= \|A\|_F^2 - \|A^T H_k\|_F^2,
\end{aligned}$$

where the last step followed from cyclic property of trace, and the fact that H_k is orthogonal.

Then, denoting h_t as the column of H_k,

$$\begin{aligned}
\left| \|A^T H_k\|_F^2 - \sum_t \sigma_t^2(C) \right| &\leq \sqrt{k} \left(\sum_{t=1}^{k} (\|A^T h_t\|_F^2 - \sigma_t^2(C))^2 \right)^{1/2} \\
&\leq \sqrt{k}\|AA^T - CC^T\|_F,
\end{aligned}$$

where we have done Jensen's inequality, then subscript chase. (We can also use the Cauchy-Schwarz inequality for the first step, by treating $|\sum_i a_i| = |\mathbb{1} \cdot a|$, where $\mathbb{1}$ is the all-ones vector.)

$$\begin{aligned}\left|\sum_{t=1}^{k} \sigma_t^2(C) - \sum_{t=1}^{k} \sigma_t^2(A)\right| &\leq \sqrt{k} \left(\sum_{t=1}^{k} (\sigma_t^2(C) - \sigma_t^2(A))^2\right)^{1/2} \\ &= \sqrt{k} \left(\sum_{t=1}^{k} (\sigma_t(CC^T) - \sigma_t(AA^T))^2\right)^{1/2} \\ &\leq \sqrt{k} \|AA^T - CC^T\|_F,\end{aligned}$$

where we have done Cauchy-Schwarz inequality, then applied Hoffman-Wielandt inequality.

Then finally, if $p_i \geq \beta \frac{\|A_i\|_2^2}{\|A\|_F^2}$, then we get the usual high probability bound on $\|AA^T - CC^T\|_F$, and thus on the error. Should get an additive $\epsilon \|A\|_F^2$ error.

□

12 Thursday, February 27th (Scribe: Ashley Zhang)

12.1 Perturbation Bounds

1. $\max_i |\sigma_i(A+E) - \sigma_i(A)| \leq \|E\|_2$
2. $\sum_{i=1}^{n} (\sigma_i(A+E) - \sigma_i(A))^2 \leq \|E\|_F^2$

where,

- σ_i denotes singular values.
- $\|\|_2$ represents the spectral norm.
- $\|.\|_F$ represents the Frobenius norm.

12.2 Algorithm

Input: A **Output**: H_k, $\{\sigma_i(C)\}_{i=1}^{r}$

1. Sample C from A, i.e. $C = AS$
2. SVD C, $C = U\Sigma V^T$
3. Project A into the first k columns of U

12.3 Error Bounds and Proofs

12.3.1 Initial Claims and Proof Sketch

Claim 1: $\|A - H_k H_k^T A\|_F^2 = \|A\|_F^2 - \|H_k^T A\|_F^2$

Claim 2: $\|A - H_k H_k^T A\|_2^2 \leq \|A - A_k\|_2^2 + 2\text{Tr}\|AA^T - CC^T\|_F \|H_k H_k^T A\|_F$

where A_k is the best rank-k approximation of A.

Claim 3: $\left| \|H_k^T A\|_F^2 - \sum_{i=1}^{k} \sigma_i^2(C) \right| \leq \sqrt{k}\|AA^T - CC^T\|_F$

Claim 4: $\left| \sum_{i=1}^{k} \sigma_i^2(C) - \sum_{i=1}^{k} \sigma_i^2(A) \right| \leq \sqrt{k}\|A^T A - C^T C\|_F$

- Intermediate step: $\sum \epsilon_i(\sigma_i^2(c) - \sigma_i A^T A)$
- (where $C = A \cdot W$)

Claim 5: $\|A - H_k H_k^T A\|_2^2 \leq \|A - A_k\|_2^2 + 2\|AA^T - CC^T\|_2 \qquad (\sigma_{k+1}^2(A))$

Proof of Claim 5:

Let $\mathcal{H}_k = \text{range}(H_k) = \text{span}\{h_1, \ldots, h_k\}$. Let $\mathcal{H}_{k^\perp}$ be the orthogonal complement of $\mathcal{H}_k$.

Consider an arbitrary vector x:

$$x = \alpha y + \beta z$$

where

- $\alpha^2 + \beta^2 = 1$
- $y \in \mathcal{H}_k$
- $z \in \mathcal{H}_{k^\perp}$

$$\begin{aligned}
\|A - H_k H_k^T A\|_2^2 &= \max_{\substack{x \in \mathbb{R}^n \\ \|x\|_2 = 1}} \|x^T (A - H_k H_k^T A)\|_2^2 \\
&= \max_{\substack{y \in \mathcal{H}_k, z \in \mathcal{H}_{k^\perp} \\ \alpha^2 + \beta^2 = 1}} \|(\alpha y^T + \beta z^T)(A - H_k H_k^T A)\|_2^2 \\
&\leq \max_y \|y^T (A - H_k H_k^T A)\|_2^2 + \max_z \|z^T (A - H_k H_k^T A)\|_2^2 \qquad (*) \\
&= \max_z \|z^T A\|_2^2
\end{aligned}$$

where

- The condition $x \in \mathbb{R}^n$ is implicitly included.
- The cross-terms vanish because $y^T(A - H_k H_k^T A) = 0$ (projection property).

$$\begin{aligned}
\|z^T A\|_2^2 &= z^T A A^T z \\
&= z^T A A^T z - z^T C C^T z + z^T C C^T z \\
&= -z^T C C^T z + z^T (A A^T - C C^T) z \\
&\leq \sigma_{k+1}^2(C) + \|A A^T - C C^T\|_2 \\
&\leq \sigma_{k+1}^2(A) + 2\|A A^T - C C^T\|_2
\end{aligned}$$

12.3.2 Additional Error Bounds

$$P_i = \frac{\|A_i u_i^T\|_F^2}{\|A\|_F^2}$$

$$\|A - A_k\|_2^2 + \epsilon \|A\|_F^2$$

1. $\|A - P_{C_k} A\|_2^2 \leq \|A - A_k\|_2^2 + \epsilon \|A\|_F^2 \qquad J(\epsilon)$
2. $\|A - P_{C_k} A\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon \|A\|_F^2$
3. $\|A - P_{C_k} A\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon \|A - A_k\|_F^2 = (1 \pm \epsilon)\|A - A_k\|_F^2$

$$\|A - P_{C_k} A\| \leq f(m, n, k, \mathrm{sr}, \dots) \|A - A_k\|.$$

	Additive Error	Relative Error
Sample	$P_i = \frac{\|A_i u_i^T\|_F^2}{\|A\|_F^2}$	$P_i \sim \frac{\|v_i^{(k)}\|_2^2}{k}$
Projection	Smid Cassia, eR.	Smid Cassia, eR.

12.4 General Considerations

Q: How to measure quality?

- All norms.
- Classical results (σ, h_k, etc.).
- Sampling vs. projection.
- More aggressive subsampling.
- "Hypothesized data" (meaning unclear).
- Describe a specific [application/scenario].
- Sampling regime vs. numerical robustness.

12.5 Tuned/Better Low Rank Approximation Algorithms

Tuned Better Low Rank Approx. Algs.

- Smooth unitary structure

Unclear

- Iterative
  - Iterative/[Unclear]
- Score over subsets of size k: $\begin{pmatrix} n \\ k \end{pmatrix}$

12.6 Optimization Problems

1. $\min \|AX - B\|_F \to X_{opt} = A^+ B$
2. $\min \|SAX - SB\|_F \to \hat{X}_{opt} = (SA)^+ (SB)$

Special Case:

$$A \to A_k$$
$$B \to A$$

Resulting expressions:

- $\|A(SA)^+SB\|_F - \|AA^+B - B\|_F$ (where $B \to I^{\perp T}$)
- $\|A_k(SA_k)^+SA - A\|_F$
- A_{ij}

12.7 Finding Good Columns

Q: How might you find good columns?

- Sampler
- QR decomposition
- Greedy iterative methods

Unclear

- Score over best set of k-columns.

13 Tuesday, March 4th (Scribe: Lecong Ding)

13.1 SelectColumnsMultiPass

Algorithm 8 SelectColumnsMultiPass

Input: An $m \times n$ matrix A, an integer c such that $1 \leq c \leq n$, and a positive integer t.
Output: An $m \times c$ matrix C such that $CC^+A \approx A$.
$S \leftarrow \varnothing$
for $\ell = 1$ **to** t **do**
    if $\ell = 1$ **then**
        $E_1 = A$
    else
        $E_\ell = A - A_S A_S^+ A$
    end if
    Compute (for some positive $\beta \leq 1$**) probabilities** $\{p_i\}_{i=1}^n$ **such that**
    $p_i \geq \beta \dfrac{\|E_\ell^{(i)}\|_2^2}{\|E_\ell\|_F^2}$, where $E_\ell^{(i)}$ is the i-th column of E_ℓ.
    for $j = 1$ **to** c **do**
        Pick $i_j \in \{1, \ldots, n\}$ with probability p_{i_j}.
        $S \leftarrow S \cup \{i_j\}$
    end for
end for
return $C = A_S$

What we aim to do is choose columns iteratively based on their magnitudes (e.g., their Euclidean norms). In each pass (or round), the probabilities of selecting columns depend on the current residual matrix. Specifically, we remove the contribution of the columns we have already chosen and then measure each remaining column's magnitude in that residual. We use these measurements to update the sampling probabilities for the next round.This algorithm takes t rounds, and each round is 2 passes over the data. In the first round, it computes the simple sampling probabilities

$$p_i \;=\; \frac{\|A^{(j)}\|_2^2}{\|A\|_F^2}$$

in the first pass and then pulls out the actual columns in the second pass. In the second and subsequent rounds, ditto, except that the sampling probabilities depend on the length of the columns of the matrix E, that is the residual after you subtract the projection of A onto the subspace spanned by the columns in the first $\ell - 1$ rounds. That is, if $\{S_1, S_2, \ldots, S_{\ell-1}\}$ is a multiset chosen in the first $\ell - 1$ rounds, then $C_{\ell-1} = A_{S_1 S_2 \cdots A_{\ell-1}}$ is the $m \times |S_1||S_2| \cdots |S_{\ell-1}|$ matrix with columns of A that has indices in $\{S_1, S_2, \ldots, S_\ell\}$. So, $E_\ell = A - A_{\{S_1,\ldots,S_{\ell-1}\}} A^+_{\{S_1,\ldots,S_{\ell-1}\}} A = A - C_{\ell-1} C^+_{\ell-1} A$. Each pass subtracts the projection onto the subspace spanned by the already chosen columns.We allow repeated selection of the same column across multiple trials or rounds, which is why S can be considered a multi-set. In practice, duplicates often contribute little, but the algorithm does not forbid them. Single-pass column selection uses the original A alone to sample columns. In contrast, multi-pass methods refine sampling probabilities based on the residual after removing already-chosen columns' contributions. This can drastically reduce approximation error in subsequent rounds. Conceptually, Gram-Schmidt takes columns one by one, projecting away the sub-

space spanned by previously selected vectors. Here, we do something similar but in a randomized framework. Instead of a purely greedy pick, we sample columns according to updated probabilities based on the residual. In practical large-scale tasks, we usually combine random projections or pre-conditioning to even out column norms before sampling. Alternatively, we directly sample from the residual in each pass. Either approach can help avoid pathological cases where columns are extremely correlated or ill-conditioned.

Theorem 2. Suppose $A \in \mathbb{R}^{m \times n}$ and let C be the $m \times tc$ matrix constructed by sampling c columns of A in each of t rounds with the SelectColumnsMultiPass algorithm. If $\eta = \sqrt{1 + \frac{8}{\beta} \log(\frac{1}{\delta})}$ for any $0 < \delta < 1$, then, with probability at least $1 - t\delta$,

$$\|A - CC^+A\|_F^2 \;\leq\; \frac{1}{1-\epsilon}\,\|A - A_k\|_F^2 \;+\; \epsilon^t\,\|A\|_F^2\,,$$

if $c \;\geq\; \frac{4\,\eta^2\,k}{\beta\,\epsilon^2}$ columns are picked in each of the t rounds.

Recall that we will go with the following, which is a simpler and improved proof, compared with that of RVW.

Proof. The proof will be by induction on the number of rounds t. Let S_1 denote the set of columns picked at the first round, and let $C_1 = A_{S_1}$. Thus, C_1 is an $m \times c$ matrix, where $c \geq 4\,\eta^2\,k/(\beta\,\epsilon^2)$. By Theorem 21 and since $1 < 1/(1-\epsilon)$ for $\epsilon > 0$, we have that

$$\|A - C_1\,C_1^+\,A\|_F^2 \;\leq\; \frac{1}{1-\epsilon}\,\|A - A_k\|_F^2 \;+\; \epsilon\,\|A\|_F^2 \tag{100}$$

holds with probability at least $1 - \delta$, thus establishing the base case of the induction.

Next, let $(S_1, \ldots, S_{t-1})$ denote the set of columns picked in the first $t-1$ rounds and let $C_{t-1} = A_{(S_1,\ldots,S_{t-1})}$. Assume that the proposition holds after $t-1$ rounds, i.e., assume that by choosing $c \;\geq\; 4\,\eta^2\,k/(\beta\,\epsilon^2)$ columns in each of the first $t-1$ rounds, we have

$$\|A - C_{t-1}\,C_{t-1}^+\,A\|_F^2 \;\leq\; \frac{1}{1-\epsilon}\,\|A - A_k\|_F^2 \;+\; \epsilon^{t-1}\,\|A\|_F^2 \tag{101}$$

holds with probability at least $1 - (t-1)\delta$.

We will prove that it also holds after t rounds. Let $E_t = A - C_{t-1}\,C_{t-1}^+\,A$ be the residual of the matrix A after subtracting the projection of A on the subspace spanned by the columns sampled in the first $t-1$ rounds. (Note that it is $\|E_t\|_F^2$ that is bounded by (101).)

Consider sampling columns of E_t at round t with probabilities proportional to the square of their Euclidean lengths, i.e., according to (98), and let Z be the matrix of the columns of E_t that are included in the sample. (Note that these columns of E_t have the same span and thus projection as the corresponding columns of A when the latter are restricted to the residual space.)

Then, by choosing at least $c \geq 4\,\eta^2\,k/(\beta\,\epsilon^2)$ columns of E_t in the t-th round we can apply Theorem 21 to E_t and get that

$$\|\,E_t - Z\,Z^+\,E_t\,\|_F^2 \;\leq\; \|\,E_t - (E_t)_k\|_F^2 \;+\; \epsilon\,\|E_t\|_F^2, \tag{102}$$

which holds with probability at least $1-\delta$. By combining (101) and (102) we see that if at least $4\,\eta^2\,k/(\beta\,\epsilon^2)$ columns are picked in each of the t rounds, then

$$\| E_t - Z\,Z^+\,E_t \|_F^2 \;\;\leq\;\; \| E_t - (E_t)_k \|_F^2 \;+\; \frac{\epsilon}{1-\epsilon}\,\|A - A_k\|_F^2 \;+\; \epsilon^t\,\|A\|_F^2 \tag{103}$$

holds with probability at least $1-t\delta$. The theorem thus follows from (103) if we can establish that

$$E_t - Z\,Z^+\,E_t \;=\; A - C_t\,C_t^+\,A \quad \text{and} \quad \| E_t - (E_t)_k \|_F^2 \;\;\leq\;\; \| A - A_k \|_F^2.$$

But (104) follows from the definition of E_t, since $C_t\,C_t^+ = C_{t-1}\,C_{t-1}^+ + Z\,Z^+$ by the construction of Z, and since $Z\,Z^+\,C_{t-1}\,C_{t-1}^+ = 0$. To establish (105), and thus the theorem, notice that

$$\begin{aligned}
\|E_t - (E_t)_k\|_F^2 &= \Big\| (I - C_{t-1}C_{t-1}^+)\,A \;-\; \big((I - C_{t-1}C_{t-1}^+)A\big)_k \Big\|_F^2 \\
&\leq \Big\| (I - C_{t-1}C_{t-1}^+)\,A \;-\; \big(I - C_{t-1}C_{t-1}^+)A_k \Big\|_F^2 \\
&= \Big\| (I - C_{t-1}C_{t-1}^+)\,(A - A_k) \Big\|_F^2 \\
&\leq \|A - A_k\|_F^2.
\end{aligned}$$

Here,

- The first equality follows by definition of E_t,
- The next step follows since $(I - C_{t-1}\,C_{t-1}^+)\,A_k$ is a rank-k matrix (but not necessarily the optimal one),
- Then the inequality follows immediately,
- And the last inequality follows since $I - C_{t-1}\,C_{t-1}^+$ is a projection.

□

This algorithm and theorem demonstrate that by sampling in t rounds and by judiciously computing sampling probabilities for picking columns of A in each of the t rounds, the overall error drops exponentially with t. This is a substantial improvement over the results of Theorem 21. In that case, if $c \geq 4\,\eta^2\,k\,t/(\beta\,\epsilon^2)$ then the additional additive error is $(\epsilon/\sqrt{t})\,\|A\|_F^2$. Note also that although we have described this as an iterative additive-error low-rank matrix approximation algorithm, it becomes a relative-error approximation if the number of iterations depends on the stable rank.
Remarks: Since the algorithm proceeds in t rounds, each of which refines the residual, an inductive argument naturally captures how each pass contributes to reducing the error.A purely greedy method (like Gram-Schmidt) can be viewed as a non-random version of this. However, random sampling often avoids numerical instabilities and pathological configurations that purely deterministic approaches might encounter. In this presentation, we do not repeatedly compute a "best rank-k" approximation for each subset of columns—this simplifies the proof. In practice, one may choose to re-scale or truncate to rank k at each step, but that significantly increases bookkeeping and complexity in the proof. This inductive multi-pass sampling framework did not originate from classical linear algebra circles; rather, it arose from algorithmic and theoretical computer science considerations, showing once again that multiple passes plus randomization can yield strong approximation guarantees for large-scale problems.

13.2 Interaction between singular subspaces and sketching matrix

Recall that, given a matrix $A \in \mathbb{R}^{m\times n}$, many RandNLA algorithms seek to construct a sketch of A by post-multiplying A by some *sketching matrix* $Z \in \mathbb{R}^{n\times r}$, where r is much smaller than n. (For example, Z could represent the action of random sampling or random projection.) Thus, the resulting matrix $AZ \in \mathbb{R}^{m\times r}$ is a matrix that is much smaller than the original matrix A, and the interesting question is what kind of approximation guarantees does it offer for A.

A common approach is to explore how well AZ spans the principal subspace of A, and one metric of accuracy is the *error matrix*, $A - P_{AZ}A$, where P_{AZ} is the projection of A onto the subspace spanned by the columns of AZ. Formally,

$$P_{AZ} \;=\; (AZ)\,(AZ)^{+} \;=\; U_{AZ}\,U_{AZ}^{T},$$

where $(AZ)^{+} \in \mathbb{R}^{r\times m}$ is the Moore–Penrose pseudoinverse of AZ, and it can be computed via the SVD of AZ; see [38] for details. Similarly, $U_{AZ} \in \mathbb{R}^{m\times\rho}$ is the matrix of the left singular vectors of AZ, where ρ is the rank of AZ.

The following structural result offers a means to bound any unitarily invariant norm of the error matrix $A - P_{AZ}A$.

Lemma 8. Given $A \in \mathbb{R}^{m\times n}$, let $Y \in \mathbb{R}^{n\times k}$ be any matrix such that $Y^TY = I_k$. Let $Z \in \mathbb{R}^{n\times r}$ (with $r \geq k$) be any matrix such that Y^TZ and AY both have full rank. Then, for any unitarily invariant norm $\|\cdot\|$, we have

$$\|A - P_{AZ}A\| \;\leq\; \|A - AYY^T\| \;+\; \|A - AYY^TZ(Y^TZ)^{+}\|. \tag{1}$$

Lemma 8 holds for *any* matrix Z, regardless of whether Z is constructed deterministically or randomly. In the context of RandNLA, typical constructions of Z would represent a random sampling or random projection operation.

The orthogonal matrix Y in the above lemma is also arbitrary. In the context of RandNLA, one can think of Y either as $Y = V_k$, where $V_k \in \mathbb{R}^{n\times k}$ is the matrix of the top k right singular vectors of A, or as some other orthogonal matrix that approximates V_k. But Lemma 8 holds more generally.

As stated in Lemma 8, Y must satisfy two conditions: the matrix Y^TZ must have full rank, equal to k (since $r \geq k$), and the matrix AY must also have full rank, again equal to k. If $Y = V_k$, then the constraint that AY must have full rank is trivially satisfied, assuming that A has rank at least k. Additionally, the sampling and random projection approaches that are used in high-quality RandNLA algorithms with sufficiently large values of r guarantee that the rank condition on Y^TZ is satisfied. More generally, though, one could perform an *a posteriori* check that these two conditions hold.

In this section, we provide the proof of Lemma 1. We start by noting that

$$A - P_{AZ}A \;=\; A - (AZ)(AZ)^{+}A.$$

Then, for any unitarily invariant norm $\|\cdot\|$,

$$(AZ)^+A \;=\; \underset{X\in\mathbb{R}^{r\times n}}{\arg\min} \big\|A-(AZ)X\big\|. \tag{11}$$

This implies that in Eqn. (11) we can replace $(AZ)^+A$ with any other matrix in $\mathbb{R}^{r\times n}$, and the equality changes to an inequality. In particular, we replace $(AZ)^+A$ with $(A\,YY^TZ)^+AYY^T$, where AYY^T is a rank-k approximation to A (not necessarily the best rank-k approximation to A). Thus,

$$||A-P_{AZ}A|| \;=\; ||A-AZ(AZ)^+A|| \;\leq\; ||A-AZ(AYY^TZ)^+AYY^T||.$$

This suboptimal choice for X is essentially the heart of our proof: it allows us to manipulate and further decompose the error term, thus making the remainder of the analysis feasible. By expressing $A = A - AYY^T + AYY^T$ and applying the triangle inequality, we get

$$\begin{aligned}||A-P_{AZ}A|| \;&\leq\; ||A-AYY^T+AYY^T-(A-AYY^T+AYY^T)Z(AYY^TZ)^+AYY^T|| \\ &\leq\; ||A-AYY^T|| + ||AYY^T-AYY^TZ\big(AYY^TZ\big)^+AYY^T|| \\ &\quad+\; ||(A-AYY^TZ)\big(AYY^TZ\big)^+AYY^T||\end{aligned}$$

We now prove that the second term in the last inequality is equal to zero. Indeed,

$$\begin{aligned}||AYY^T-AYY^TZ(AYY^TZ)^+AYY^T|| &= ||AYY^T-AYY^TZ(Y^TZ)^+(AY)^+AYY^T|| \\ &= 0\end{aligned}$$

where we used the fact that both matrices Y^TZ and AY have full rank, which implies

$$(Y^TZ)\,(Y^TZ)^+ \;=\; I_k \quad\text{and}\quad (AY)^+(AY) \;=\; I_k.$$

This concludes the derivation. Using the same manipulations and dropping Y^T via unitary invariance, we finally get

$$||(A-AYY^T)Z\big(AYY^TZ\big)^+AYY^T|| \;=\; ||(A-AYY^T)Z(Y^TZ)^+||$$

which concludes the proof of Lemma 1.

14 Thursday, March 6th (Scribe: Hyunsuk Kim)

Today, we will discuss examples of applying the theorem introduced in the last lecture.

Theorem 3. Let $A \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{n\times k}$, and $S \in \mathbb{R}^{n\times r}$ and suppose Q^TS and AS are full rank. Then,

$$\|A - P_{AS}A\|_\xi \leq \|A - QQ^T\|_\xi + \|(A - AQQ^T)S(A^TS)^\dagger\|_\xi$$

for any unitarily invariant norm ξ.

Throughout the lecture we consider the case $Q = V_k \in \mathbb{R}^{n\times k}$, the first k columns of the matrix V from the SVD decomposition $A = U\Sigma V^T$.

There are four major applications for this theorem:

1) $(1+\epsilon)$-Frobenius norm bound: In previous lectures, we achieved this by sampling $\mathcal{O}(k \log k/\epsilon)$ columns.
2) Column Subset Selection Problem (CSSP): We now aim to select exactly k columns from the original matrix A.
3) Nyström approximations: Given a symmetric positive semidefinite (SPSD) matrix A, we seek a low-rank approximation that preserves the SPSD property.
4) Randomized SVD

In this lecture, we focus on the CSSP problem, leaving the remaining two applications for the next lecture.

14.1 CSSP

To begin, the CSSP problem involves constructing an $m \times k$ matrix C by selecting the 'best' or 'good' k columns from the original matrix A. The definition of 'best' or 'good' may vary; for example, it may refer to minimizing $\|A - P_CA\|_\xi$ for a unitarily invariant norm ξ or optimizing a function of R_{11}, R_{12}, R_{22} in the QR factorization $C = QR$, where $R = ((R_{11}, R_{12}), (0, R_{22}))$.

Regardless of the metric used, the optimal selection is intractable because it requires minimizing the metric over all $\binom{n}{k}$ possible choices for C. Consequently, we do not know the exact optimal value. For example, for $\|A - P_CA\|_\xi$, the optimal value $\min_C \|A - P_CA\|_\xi$ is unknown. However, we do have a simple lower bound: $\|A - A_k\|_\xi$, where $A_k = AV_kV_k^T$ and our goal is to obtain C such that $\|A - P_CA\|_\xi \leq \alpha\|A - A_k\|_\xi$ for some constant α.

To obtain exactly k columns from A, numerical linear algebra (NLA) and theoretical computer science (TCS) researchers take different approaches. NLA researchers use deterministic methods, such as selecting exactly k columns based on QR decomposition (e.g., choosing the first k linearly independent columns). In contrast, TCS researchers use randomized algorithms, where more than k but fewer than $k \log k/\epsilon$ columns are initially sampled before being cut back to k columns. The following algorithm combines elements of both approaches.

Algorithm 9 Randomized Column Subset Sampling Algorithm

Input: Matrix $A \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{m\times k}$, number of columns $k \in \mathbb{Z}^+$.

Initial Stage:

Compute (exactly or approximately) the top k right singular vectors of A denoted as V_k.

Compute (exactly or approximately) the importance sampling probabilities $\{p_i\}_{i=1}^n$, where

$$p_i = \frac{1}{k}\left\|(V_k)_{(i)}\right\|_2^2.$$

Set $c = \Theta(k \log k)$ (without dependence on ϵ, e.g. $1/2$).

Randomized Stage:

for each $t \in [c]$ **do**

    Choose an index $i \in [n]$ with probability p_i.

    If i is chosen, apply the scaling factor $1/\sqrt{cp_i}$.

end for

Form the sampling matrix $S_1 \in \mathbb{R}^{n\times c}$ and diagonal rescaling matrix $D_1 \in \mathbb{R}^{c\times c}$ according to the chosen columns and the corresponding scaling factors.

Deterministic Stage:

Run a deterministic QR algorithm (selecting k linearly independent columns out of total c columns, e.g., Pan or Gu-Eisenstat) on $V_k^T S_1 D_1$.

Extract exactly k columns from $V_k^T S_1 D_1$, forming the sampling matrix S_2.

Return $C = AS_1S_2$, the reconstructed k columns of A.

Note that S_1 contains randomness, whereas S_2 is deterministic once a QR algorithm is chosen. We now present the theoretical guarantee for the algorithm above.

Theorem 4. Let $A \in \mathbb{R}^{m\times n}$, and let $k \in \mathbb{Z}^+$. If we run the above CSSP algorithm (Algorithm 9) with Gu-Eisenstat QR decomposition, then with constant probability, the following hold.

$$\begin{aligned}
\|A - P_C A\|_2 &\le \Theta\left(k \log^{1/2}(k)\right)\|A - A_k\|_2 + \Theta\left(k^{3/4}\log^{1/4}(k)\right)\|A - A_k\|_F \\
\|A - P_C A\|_F &\le \Theta\left(k \log^{1/2}(k)\right)\|A - A_k\|_F \\
\|A - P_C A\|_* &\le \|A - A_k\|_* + \Theta\left(k^{3/2}\log^{1/2}(k)\right)\|A - A_k\|_F,
\end{aligned} \tag{12}$$

where $\|\cdot\|_*$ denotes the nuclear norm, which is the sum of singular values.

Before proceeding with the proof sketch, it is worth noting two important points. First, recall that the coefficient for the upper bound of $\|A - P_C A\|_F$ was previously $(1+\epsilon)$. Now, however, we have a much larger coefficient. This arises from a trade-off; we are selecting only k columns, which is significantly fewer than the $k \log k/\epsilon$ columns chosen in the previous setting. Second, observe that the relationship among the matrix norms $\|\cdot\|_2, \|\cdot\|_F$ and $\|\cdot\|_*$ resemble those among the vector norms $\|\cdot\|_\infty, \|\cdot\|_2$ and $\|\cdot\|_1$. This similarity exists because these three matrix norms correspond to, respectively, the maximum singular value, the root mean square of the singular values, and the

sum of the singular values. Consequently, we have

$$\|A\|_2 \leq \|A\|_F \leq \|A\|_* \leq \sqrt{r}\|A\|_F \leq r\|A\|_2$$

for $r \geq \operatorname{rank}(A)$.

Proof sketch. Theorem 3, applied with $Q = V^k$ and $S = S_1 D_1 S_2$, yields

$$\begin{aligned}\|A - P_C A\|_\xi &\leq \|A - A_k\|_\xi + \|(A - A_k)S_1 D_1 S_2 (V_k S_1 D_1 S_2)^\dagger\|_\xi \\ &\leq \|A - A_k\|_\xi + \|(A - A_k)S_1 D_1 S_2\|_\xi \|(V_k S_1 D_1 S_2)^\dagger\|_2 \\ &=\leq \|A - A_k\|_\xi + \|(A - A_k)S_1 D_1\|_\xi \|(V_k S_1 D_1 S_2)^\dagger\|_2.\end{aligned}$$

The second inequality follows from strong submultiplicativity, while the last equality holds due to the unitarily invariance of the norm $\|\cdot\|_\xi$. A theorem on the Gu-Eisenstat QR decomposition states that

$$\|(V_k^T S_1 D_1 S_2)^\dagger\|_2 = \sigma_k^{-1}(V_k S_1 D_1 S_2) \leq 2\sqrt{k(c-k)-1},$$

where c denotes the number of columns we sampled in the randomized stage. It now remains to bound $\|(A - A_k)S_1 D_1\|_\xi$. Applying the following lemma, which we state without proof, completes the argument.

Lemma 9. With probability at least 0.9, we have that

$$\begin{aligned}\|(A - A_k)S_1 D_1\|_2 &\leq \|A - A_k\|_2 + \frac{12}{c^{1/4}}\|A - A_k\|_F \\ \|(A - A_k)S_1 D_1\|_F &\leq \sqrt{10}\|A - A_k\|_F \\ \|(A - A_k)S_1 D_1\|_* &\leq \sqrt{10c}\|A - A_k\|_F.\end{aligned}$$

□

15 Tuesday, March 11th (Scribe: Hanyang Li)

15.1 Review of last week: CSSP

Error analysis of CSSP. Recall the key part in the proof:

$$\|A - QQ^\top A\|_\xi \le \|A - A_k\|_\xi + \left\|(A - A_k)S\,(Q^\top S)^\dagger\right\|_\xi .$$

Randomized SVD:

1. Find Q such that $Q^\top Q = I$ and $QQ^\top A \approx A$.
2. Compute an approximation of the left singular vectors or singular values from Q.

15.2 Basic Randomized SVD

Algorithm 10 AlgBasicRandSVD

Require: $A \in \mathbb{R}^{m\times n}$, target rank k, oversampling parameter p (e.g., $p = 2, 3, 10, \ldots$)
Ensure: Matrices U, D, V approximating the $(k+p)$-SVD of A, i.e., $A \approx UDV^\top$

1: Generate a random matrix $G \in \mathbb{R}^{n\times(k+p)}$ with elements i.i.d. Gaussian
2: Compute $Y = A\,G$
3: Compute the QR factorization $Y = Q\,R$
4: Set $Q = \mathrm{Orth}(Y)$ (alternatively, $Q = A\,G\,R^{-1}$)
5: Form $B = Q^\top A$
6: Compute the SVD of B, i.e., $B = U_B\,\Sigma_B\,V_B^\top$
7: Set $U = Q\,U_B$

Useful Facts. Let S, T be fixed matrices and let G be an i.i.d. Gaussian matrix. Then,

$$\mathbb{E}\left[\|S\,G\,T\|_F^2\right] = \|S\|_F^2\,\|T\|_F^2, \tag{13a}$$

$$\mathbb{E}\left[\|S\,G\,T\|_2^2\right] \le \|S\|_2\,\|T\|_F + \|S\|_F\,\|T\|_2, \tag{13b}$$

$$\mathbb{E}\left[\|G^\dagger\|_F^2\right] \le \frac{k}{p-1}, \tag{13c}$$

$$\mathbb{E}\left[\|G^\dagger\|_2\right] \le \frac{e\sqrt{k+p}}{p}. \tag{13d}$$

Claim 1:

$$\mathbb{E}\left[\|A - QQ^\top A\|_F\right] \le \sqrt{1 + \frac{k}{p-1}}\,\|A - A_k\|_F.$$

Proof. Write the singular value decomposition of A as

$$A = U\,\Sigma\,V^\top \quad \text{with} \quad \Sigma = \begin{bmatrix} \Sigma_k & \\ & \Sigma_{k\perp} \end{bmatrix} \text{ and } V = \begin{bmatrix} V_k^\top \\ V_{k\perp}^\top \end{bmatrix}.$$

Now define $\Omega_1 \triangleq \Sigma_k {V_k}^\top$ and $\Omega_2 \triangleq \Sigma_{k\perp} {V_{k\perp}}^\top$. Then we can write the SVD succinctly as

$$A = U \begin{bmatrix} \Omega_1 \\ \Omega_2 \end{bmatrix}.$$

Since G is rotationally invariant, the distribution of $V^\top G$ is the same as that of G. Therefore, Ω_1 and Ω_2 share the same stochastic properties. Then,

$$\begin{aligned}
\mathbb{E}\left[\|A - QQ^\top A\|_F\right] &\le \left(\mathbb{E}\left[\|A - QQ^\top A\|_F^2\right]\right)^{1/2} \\
&= \left(\|A - A_k\|_F^2 + \mathbb{E}\left[\|\Sigma_{k\perp}\,\Omega_2\,\Omega_1^\dagger\|_F^2\right]\right)^{1/2} \\
&= \left(\|A - A_k\|_F^2 + \mathbb{E}\Big[\mathbb{E}\big[\|\Sigma_{k\perp}\,\Omega_2\,\Omega_1^\dagger\|_F^2 \mid \Omega_1\big]\Big]\right)^{1/2} \\
&\overset{(13a)}{=} \left(\|A - A_k\|_F^2 + \mathbb{E}\Big[\|\Sigma_{k\perp}\|_F^2\,\|\Omega_1^\dagger\|_F^2\Big]\right)^{1/2} \\
&\overset{(13c)}{\le} \left(\|A - A_k\|_F^2 + \|\Sigma_{k\perp}\|_F^2\,\frac{k}{p-1}\right)^{1/2}.
\end{aligned}$$

Since $\|\Sigma_{k\perp}\|_F = \|A - A_k\|_F$, the claim follows. □

Claim 2:

$$\mathbb{E}\left[\left\|A - QQ^\top A\right\|_2\right] \le \left(1 + \sqrt{\frac{k}{p-1}}\right)\|A - A_k\|_2 + \frac{e\sqrt{k+p}}{p}\,\|A - A_k\|_F.$$

Proof. We begin by noting that

$$\begin{aligned}
\left\|A - QQ^\top A\right\|_2 &\le \left(\|\Sigma_{k\perp}\|_2^2 + \mathbb{E}\left[\left\|\Sigma_{k\perp}\,\Omega_2\,\Omega_1^\dagger\right\|_2^2\right]\right)^{1/2} \\
&\le \|\Sigma_{k\perp}\|_2 + \mathbb{E}\left[\left\|\Sigma_{k\perp}\,\Omega_2\,\Omega_1^\dagger\right\|_2\right] \qquad \text{(incorrect? GAP: left as homework)} \\
&\overset{(13b)}{\le} \|\Sigma_{k\perp}\|_2 + \mathbb{E}\left[\|\Sigma_{k\perp}\|_2\left\|\Omega_1^\dagger\right\|_F + \|\Sigma_{k\perp}\|_F\left\|\Omega_1^\dagger\right\|_2\right] \\
&\overset{(13d)(13c)}{\le} \|\Sigma_{k\perp}\|_2 + \|\Sigma_{k\perp}\|_2\sqrt{\frac{k}{p-1}} + \|\Sigma_{k\perp}\|_F\,\frac{e\sqrt{k+p}}{p}.
\end{aligned}$$

This completes the proof. □

16 Thursday, March 13th (Scribe: Roger Hsiao)

Basic Randomized SVD (BasicRandSVD)

Input: A matrix $A \in \mathbb{R}^{m \times n}$ and target column count k.

1. **Stage 1:** Compute $AG = QR$, where G is a Gaussian random matrix.
2. **Stage 2:** Use Q to compute downstream factorizations:
   - CUR decomposition
   - Interpolative Decomposition (ID)
   - Nyström approximation
   - etc.

Interpolative Decomposition (ID)

We will focus on Interpolative Decomposition (ID) today. The goal is to find a matrix C as an approximation of A,

$$A' = CC^+A,$$

where C consists of actual columns from A.

Definition: Let $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = k$. The interpolative decomposition has the form

$$A \approx CZ,$$

where $C \in \mathbb{R}^{m \times k}$ consists of k selected columns of A, and $Z \in \mathbb{R}^{k \times n}$ is "nice" (well-conditioned).

- **Column ID:** $A \approx CZ = A(:, I_s)Z$
- **Row ID:** $A \approx XR = XA(I_s, :)$
- **Two-sided ID:** $A \approx XA(I_{s1}, I_{s2})Z$

I_s, I_{s1}, and I_{s2} denote index sets used to select columns or rows from A.

Deterministic Algorithm for Column ID

A deterministic algorithm to compute the column ID (can be extended to other forms of ID) is the QR factorization with column pivoting (QRCP).

$$
\begin{aligned}
AP &= QR \\
&= \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} \\
&= Q_1 S_{11} \begin{bmatrix} I_k & S_{11}^{-1} S_{12} \end{bmatrix} + Q_2 \begin{bmatrix} 0 & S_{22} \end{bmatrix} \\
&= CZ + Q_2 \begin{bmatrix} 0 & S_{22} \end{bmatrix}
\end{aligned}
$$

P is the column reordering matrix. The approximation error is

$$\|AP - CZ\| = \left\| Q_2 \begin{bmatrix} 0 & S_{22} \end{bmatrix} \right\| = \|S_{22}\|.$$

Note: This approach is particularly useful in randomized algorithms — if we obtain a good basis for the top part of the spectrum of A, it can be useful in lots of other things.

Randomized Algorithm for ID

Input: A matrix A and target rank k.

1. Generate a Gaussian random matrix G with i.i.d. entries.
2. Compute the sketch: $Y = AG$.
3. Perform q power iterations (optional, to improve spectral decay):
   - For $j = 1$ to q:

     $$Y = A^\top Y$$
     $$Y = AY$$

     After q iterations, $Y = (AA^\top)^q G$.
4. Compute a row interpolative decomposition (row ID) of Y.

Two-sided ID

Given a matrix A, let $C = A(:, I_{s1})$ and $R = A(I_{s2}, :)$

We want

$$A \approx CUR,$$

where U is a "nice".

Recall

$$A \approx CZ.$$

Let

$$U = ZR^{+},$$

Then

$$A \approx CZ = CZR^{+}R = CUR.$$

CSSP

Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive semidefinite (SPSD), and let $S \in \mathbb{R}^{n \times \ell}$ be a sketching matrix. Define:

$$C = AS, \quad W = S^{\top}AS.$$

We construct an approximation to A as:

$$A' = CW^{+}C^{\top}.$$

If C consists of actual columns of A, then this approximation corresponds to the *Nyström method*.

Special Case: $S = V_k$

Suppose we set $S = V_k$, the matrix of the top k singular vectors of A. Then:

$$\begin{aligned} A' &= AS(S^{\top}AS)^{+}S^{\top}A \\ &= AV_k(V_k^{\top}AV_k)^{+}V_k^{\top}A \\ &= AA_k^{+}A, \end{aligned}$$

where $A_k = V_k\Sigma_k V_k^{\top}$ is the best rank-k approximation to A.

Nyström Approximation: General Case

Let $C = AS$ and consider the QR decomposition $C = QR$, where $Q = ASR^{-1}$ is an orthonormal basis for the range of C.

We consider two forms of approximations to A:

$$A' = QQ^{\top}AQQ^{\top}$$

and

$$A'' = AQ(Q^\top AQ)^+ Q^\top A$$

Then

$$\begin{aligned} CW^+C^\top &= AS(S^\top AS)^+ S^\top A \\ &= A^{1/2}A^{1/2}S\left(S^\top A^{1/2}A^{1/2}S\right)^+ S^\top A^{1/2}A^{1/2} \\ &= A^{1/2}P_{A^{1/2}S}A^{1/2}, \end{aligned}$$

where $P_{A^{1/2}S}$ denotes the orthogonal projector onto the column space of $A^{1/2}S$.

17 Tuesday, March 18th (Scribe: Ang Xu)

17.1 Approximation Method

A central requirement for random sketch is:

$$A^\top S^\top S\, A \;\approx\; A^\top A.$$

One way to make this precise is via a *subspace embedding* guarantee:

$$(1-\epsilon)\, A^\top A \;\preceq\; A^\top S^\top S\, A \;\preceq\; (1+\epsilon)\, A^\top A,$$

which can also be phrased in terms of spectral norms as

$$\left\| A^\top S^\top S\, A - A^\top A \right\| \;\le\; \epsilon\, \|A^\top A\|.$$

When such a bound holds with high probability (for an appropriately constructed random matrix S), one says S *embeds* the column space of A into $\mathbb{R}^r$ with small distortion.

In practice, each entry of S may be chosen as an independent Gaussian of variance $1/r$, or one might use a structured transform such as a random Hadamard or Fourier matrix (possibly combined with random sampling). The main takeaway is that multiplying by S can drastically reduce the size of data, yet still retain useful spectral information about A.

If $e = b - A\,x$, then

$$e^\top e = \|b - A\,x\|_2^2.$$

Sometimes one sets

$$\sigma^2 = \frac{1}{m}\, e^\top e$$

as an estimate of noise variance in a regression setting. Also, if

$$x_r = (A^\top S^\top S\, A)^{-1}\, (A^\top S^\top S\, b),$$

then $E[x_r] \approx x_{\text{opt}}$ provided $E[S^\top S] = I$ and similarly $E[(S^\top S)\, A] = A$. Another expression:

$$(A^\top S^\top S\, A)^{-1} \approx (A^\top A)^{-1}$$

when $\|A^\top S^\top S\, A - A^\top A\|$ is small. In such cases,

$$\|A\, x_r - b\|_2^2 \;\approx\; \|A\, x_{\text{opt}} - b\|_2^2$$

and

$$\|x_r - x_{\text{opt}}\|_2 \;\lesssim\; \epsilon\, \|x_{\text{opt}}\|_2.$$

A random matrix $S \in \mathbb{R}^{r \times m}$ can be used to reduce dimension. One common choice is to let each entry of S be an independent Gaussian with mean 0 and variance $1/r$. Then the transformed problem becomes:

$$\min_x \ \|S\,A\,x \ - \ S\,b\|_2^2.$$

Its solution is often written as:

$$x_r = \big(A^\top S^\top S\,A\big)^{-1}\,\big(A^\top S^\top S\,b\big).$$

Sometimes one denotes it by x_{sketch} or x_{rr}. The matrix $A^\top S^\top S\,A$ approximates $A^\top A$ when r is large.

One often uses norms to measure the approximation quality. For instance, if $\|\cdot\|$ is the spectral norm, then one wants

$$\big\|A^\top S^\top S\,A - A^\top A\big\| \le \epsilon\,\|A^\top A\|,$$

with high probability. Equivalently,

$$(1-\epsilon)\,A^\top A \ \preceq \ A^\top S^\top S\,A \ \preceq \ (1+\epsilon)\,A^\top A,$$

which ensures that the normal equations for the sketched system are close to the original. As a consequence,

$$\|x_r - x_{\text{opt}}\|_2 \le \delta\,\|x_{\text{opt}}\|_2$$

for some small δ, depending on ϵ. Another bound states:

$$\|A\,x_r - b\|_2^2 \le (1+\epsilon)\,\|A\,x_{\text{opt}} - b\|_2^2.$$

17.2 LS Regression

There are curves or diagrams illustrating how random sketches reduce the risk of overfitting in polynomial regression. If A is formed by columns $\{1, x, x^2, \dots, x^d\}$ for data points $\{x_i\}$, then the dimension of the problem grows with d. One can imagine a mean-squared error (MSE) plot vs. polynomial degree. At moderate d, the error is small, but as d becomes large, overfitting might increase MSE for new points. A random sketch projects the data into a lower dimension r, effectively reducing the number of degrees of freedom. Hence,

$$A \ \mapsto \ S\,A, \quad b \ \mapsto \ S\,b,$$

and

$$x_r = \arg\min_x \ \|S\,A\,x - S\,b\|_2^2$$

helps control overfitting.

One may also see references to moment calculations such as:

$$\mathbb{E}[\,S^\top S\,] = I_m, \quad \mathbb{E}\big[A^\top S^\top S\,A\big] = A^\top A,$$

and standard variance relations:

$$\mathbb{E}\big(\alpha^2\big) = \mathrm{Var}(\alpha) + \big(\mathbb{E}[\alpha]\big)^2.$$

These come up when analyzing the distribution of $\big(A^\top S^\top S\,A\big)$ or the distribution of the sketched solution x_r.

17.3 Low-rank factorization

If A is large but approximately low-rank, then sampling or projecting columns can capture its main subspace. One approach uses a random matrix $\Omega \in \mathbb{R}^{n\times k}$ so that

$$Y = A\,\Omega, \quad Q = \operatorname{orth}(Y), \quad A \approx Q\,Q^{\top} A.$$

This yields approximate SVD computations with complexity lower than classical algorithms. Sometimes the columns of A are sampled or combined to form a matrix C, leading to factorization $A \approx C\,U$.

$$x_{\mathrm{rr}} = \left(A^{\top} S^{\top} S\,A\right)^{-1} \left(A^{\top} S^{\top} S\,b\right),$$
$$\|A\,x_{\mathrm{rr}} - b\|_2^2, \quad \|x_{\mathrm{rr}} - x_{\mathrm{opt}}\|_2,$$
$$\mathbb{E}[\,x_{\mathrm{rr}}\,], \quad \operatorname{Var}(x_{\mathrm{rr}}).$$

These relate to the normal equations under the random transformation. One might also see an outline of how to use Johnson–Lindenstrauss ideas to show

$$\|S\,u\|_2^2 \approx \|u\|_2^2$$

for vectors u in certain subspaces of $\mathbb{R}^m$. In the case of polynomial fitting, let

$$A_d = \begin{bmatrix} 1 & x & x^2 & \dots & x^d \end{bmatrix}$$

for data points $\{x_1, \dots, x_m\}$. Then

$$\min_x \|A_d\,x - b\|_2^2$$

could overfit for large d. By applying S,

$$\min_x \|S\,A_d\,x - S\,b\|_2^2$$

acts as a dimension reduction. Graphically, the mean-squared error might remain stable for moderate d instead of blowing up.

Sometimes, one sees references to an iterative method: if $\kappa(A^{\top}A)$ is large, then preconditioning with $(A^{\top}S^{\top}S\,A)^{-1}$ can accelerate convergence. The random approach forms a rough but effective preconditioner at lower cost than classical factorization. The main analysis step is bounding

$$\|(A^{\top}S^{\top}S\,A)^{-1} - (A^{\top}A)^{-1}\|.$$

Overall, many formulas revolve around:

$$A^{\top}S^{\top}S\,A \approx A^{\top}A, \quad x_r \approx x_{\mathrm{opt}}, \quad \|Ax_r - b\|_2^2 \approx \|Ax_{\mathrm{opt}} - b\|_2^2.$$

One can also combine sampling or structured transforms (like discrete Hadamard) to reduce time complexity. In any case, the dimension r must scale with the rank or the number of parameters to ensure a reliable approximation.

18 Thursday, March 20th (Scribe: Jingrong Guan)

In this lecture, we consider the problem of finding the solution to the problem $Ax = b$, where $A \in \mathbb{R}^{n \times d}$ and $d > n$. Although there is no unique solution in general, the minimum l_2 norm solution is unique. The optimal solution is defined as follows:

$$x_{OPT} = \min_{Ax=b} \|x\|_2^2$$

18.1 Minimum Norm Solution

One classical solution to the minimum norm problem is as follows.

Claim: $x_{OPT} = A^\dagger b = A^\top (AA^\top)^{-1} b = V \Sigma^{-1} U^\top b$

Proof. Assume A has full row rank, which guarantees $(AA\top)^{-1}$ exists.

1) To show that it is a valid solution.

$$Ax = AA^\top (AA^\top)^{-1} b = b$$

2) To show it is the smallest norm solution.

Let x' be solution to $Ax' = b$. We first show that $(x_{OPT} - x') \perp x_{OPT}$

$$\begin{aligned}(x_{OPT} - x')^\top x_{OPT} &= (x_{OPT} - x')^\top A^\top (AA^\top)^{-1} b \\ &= (A(x_{OPT} - x'))^\top (AA^\top)^{-1} b \\ &= (b - b)^\top (AA^\top)^{-1} b \\ &= 0.\end{aligned}$$

Then, we have $\|x'\|_2^2 = \|x' - x_{OPT} + x_{OPT}\|_2^2 = \|x' - x_{OPT}\|_2^2 + \|x_{OPT}\|_2^2 \geq \|x_{OPT}\|_2^2$

Thus, we have proved that $x_{OPT} = A^\top (AA^\top)^{-1} b$.
Now, let us rewrite A using its SVD. Recall that any matrix A can be written as

$$A = U \Sigma V^\top,$$

where $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times d}$, and $V \in \mathbb{R}^{d \times d}$. Also recall that the columns U and V are orthogonal and unit norm, so that $U^\top U = I$ and $V^\top V = I$ and Σ is a diagonal matrix. Substituting in for A, we get that

$$
\begin{aligned}
x_{OPT} &= A^\top (AA^\top)^{-1} b \\
&= V\Sigma U^\top (U\Sigma V^\top V \Sigma U^\top)^{-1} b \\
&= V\Sigma U^\top (U\Sigma^2 U^\top)^{-1} b \\
&= V\Sigma U^\top U \Sigma^{-2} U^\top b \\
&= V\Sigma^{-1} U^\top b.
\end{aligned}
$$

□

18.2 Right Sketch

Similarly, we multiply A to form AS where $S \in \mathbb{R}^{d \times m}$ and solve

$$\arg \min_{ASx=z} \|z\|_2^2.$$

Because we right multiplied A by S, we changed the dimension of our optimization parameter, i.e., the vector whose norm we minimized is no longer the same dimension as x. We deal with this problem by using the fact that $x = Sz$ and hope that Sz is a good approximation for x.

Let's define

$$\tilde{z}_{OPT} := \arg \min_{ASx=z} \|z\|_2^2,$$

and use the approximation that $\tilde{x}_{OPT} = S\tilde{z}_{OPT}$. A solution for $\tilde{z}_{OPT}$ is $(AS)^\dagger b$. Plugging into the constraint formula, we see that

$$
\begin{aligned}
A\tilde{x}_{OPT} &= AS\tilde{z}_{OPT} \\
&= AS(AS)^\dagger b \\
&= b
\end{aligned}
$$

This shows that $\tilde{x}_{OPT} = S\tilde{z}_{OPT}$ is a valid solution to the original constraint $Ax = b$ and exists as long as AS has full row rank.

Now, suppose that $S \overset{\text{iid}}{\sim} \mathcal{N}(0, \frac{1}{\sqrt{l}})$ is the right sketch matrix and $\tilde{z}_{OPT}$ is the solution to the sketched problem. Our estimate for x_{OPT}, $\tilde{x}_{OPT} = S\tilde{z}_{OPT}$, has a number of important properties.

Lemma 1. For a fixed AS, $\tilde{x}_{OPT}$ is an unbiased estimator of x_{OPT}, i.e., $\mathbb{E}[\tilde{x}_{OPT}] = x_{OPT}$.

Proof. Consider the non-compact right singular vector matrix $[V \ V^\perp]$, where $V^\top \in \mathbb{R}^{d \times l}$ and $V^{\perp\top} \in \mathbb{R}^{(l-d) \times l}$. Left multiplying $\tilde{x}_{OPT}$ by $V^\top$, we get that

$$
\begin{aligned}
V^\top \tilde{x}_{OPT} &= V^\top S \tilde{z}_{OPT} \\
&= V^\top S (AS)^\dagger b \\
&= V^\top S S^\top A^\top (A S S^\top A^\top)^{-1} b \\
&= V^\top S S^\top V \Sigma U^\top (U \Sigma V^\top S S^\top V \Sigma U^\top)^{-1} b \\
&= V^\top S S^\top V \Sigma U^\top U \Sigma^{-1} (V^\top S S^\top V)^{-1} \Sigma^{-1} U^\top b \\
&= V^\top S S^\top V (V^\top S S^\top V)^{-1} \Sigma^{-1} U^\top b \\
&= \Sigma^{-1} U^\top b \\
&= V^\top V \Sigma^{-1} U^\top b \\
&= V^\top x_{OPT}.
\end{aligned}
$$

Similarly, left multiplying $\tilde{x}_{OPT}$ by $V^{\perp\top}$, we get that

$$
\begin{aligned}
V^{\perp\top} \tilde{x} &= V^{\perp\top} S \tilde{z}_{OPT} \\
&= V^{\perp\top} S S^\top V (V^\top S S^\top V)^{-1} \Sigma^{-1} U^\top b \quad \text{(steps identical to above)} \\
&= V^{\perp\top} S S^\top V (V^\top S S^\top V)^{-1} V^\top x_{OPT} \\
&= V^{\perp\top} S S^\top V (V^\top S S^\top V)^{-1} V^\top x_{OPT}.
\end{aligned}
$$

Taking the expectation of $V^{\perp\top} \tilde{x}$:

$$
\begin{aligned}
\mathbb{E}[V^{\perp\top} \tilde{x}_{OPT}] &= \mathbb{E}[V^{\perp\top} S S^\top V (V^\top S S^\top V)^{-1} V^\top x_{OPT}] \\
&= \mathbb{E}[S^\top V^\perp] S^\top V (V^\top S S^\top V)^{-1} V^\top x_{OPT} \\
&= 0.
\end{aligned}
$$

We can move $S^\top V (V^\top S S^\top V)^{-1} V^\top x_{OPT}$ out of the expectation because $S^\top V^\perp$ and $S^\top V$ are uncorrelated, and thus independent, Gaussian vector random variables. Now considering the expectation of the orthogonal matrix-vector product $[V\ V^\perp]^\top \tilde{x}_{OPT}$, we get

$$
E \begin{bmatrix} V^\top \tilde{x}_{OPT} \\ \\ V^{\perp\top} \tilde{x}_{OPT} \end{bmatrix} = \begin{bmatrix} \mathbb{E}[V^\top \tilde{x}_{OPT}] \\ \\ \mathbb{E}[V^{\perp\top} \tilde{x}_{OPT}] \end{bmatrix} = \begin{bmatrix} V^\top x_{OPT} \\ \\ 0 \end{bmatrix}.
$$

Because the expectation of $\tilde{x}$ times an orthogonal matrix is equal to x_{OPT} times an orthogonal matrix, we can conclude that $\mathbb{E}[\tilde{x}] = x_{OPT}$. □

Lemma 2. For a fixed AS, $\tilde{x}_{OPT} \sim \mathcal{N}(x_{OPT}, \frac{1}{l}(I - VV^\top) b^\top (ASS^\top A^\top)^{-1} b)$.

Proof. We did not cover this in class, but the following may serve as a reference.
Using the fact above that $\tilde{x}_{OPT} = S\tilde{z}_{OPT} = SS^\top V(V^\top SS^\top V)^{-1}V^\top x_{OPT}$, we have that

$$\mathbb{E}[(\tilde{x}_{OPT} - x_{OPT})(\tilde{x}_{OPT} - x_{OPT})^\top]$$

$$= \mathbb{E}[\tilde{x}_{OPT}\tilde{x}_{OPT}^\top] - \mathbb{E}[x_{OPT}x_{OPT}^\top]$$

$$= \mathbb{E}[SS^\top V(V^\top SS^\top V)^{-1}V^\top x_{OPT}x_{OPT}^\top V(V^\top SS^\top V)^{-\top}V^\top SS^\top] - \mathbb{E}[x_{OPT}x_{OPT}^\top]$$

$$= \mathbb{E}[SS^\top (V^\top SS^\top V)^{-1}V^\top V\Sigma^{-1}U^\top b(V\Sigma^{-1}U^\top b)^\top V(V^\top SS^\top V)^{-\top}V^\top SS^\top] - \mathbb{E}[V\Sigma^{-1}U^\top b(V\Sigma^{-1}U^\top b)^\top]$$

$$= \mathbb{E}[SS^\top V(V^\top SS^\top V)^{-1}\Sigma^{-1}U^\top bb^\top U\Sigma^{-1}(V^\top SS^\top V)^{-\top}V^\top SS^\top] - \mathbb{E}[V\Sigma^{-1}U^\top b(V\Sigma^{-1}U^\top b)^\top]$$

$$= \mathbb{E}[SS^\top V(U\Sigma V^\top SS^\top V)^{-1}bb^\top (V^\top SS^\top V\Sigma U^\top)^{-\top}V^\top SS^\top] - \mathbb{E}[V\Sigma^{-1}U^\top b(V\Sigma^{-1}U^\top b)^\top].$$

□

Lemma 3. $\mathbb{E}\|\tilde{x}_{OPT} - x_{OPT}\|_2^2 = \frac{d-n}{l-n-1}\|x_{OPT}\|_2^2$

Proof. We use the fact that $\mathbb{E}[(ASS^\top A^\top)^{-1}] = (AA^\top)^{-1}\frac{l}{l-n-1}$ and that for $z \sim \mathcal{N}(0, K)$, we have $\mathbb{E}[\|z\|_2^2] = \mathbb{E}[\mathrm{tr}zz^\top] = \mathbb{E}[\mathrm{tr}K]$.

$$\begin{aligned}
\mathbb{E}\|\tilde{x}_{OPT} - x_{OPT}\|_2^2 &= \mathbb{E}[(\tilde{x}_{OPT} - x_{OPT})^\top(\tilde{x}_{OPT} - x_{OPT})] \\
&= \mathbb{E}[\mathrm{tr}(\tilde{x}_{OPT} - x_{OPT})(\tilde{x}_{OPT} - x_{OPT})^\top] \\
&= \mathbb{E}[\mathrm{tr}\frac{1}{l}(I - VV^\top)b^\top(ASS^\top A^\top)^{-1}b] \\
&= \frac{1}{l}\mathrm{tr}(I - VV^\top)b^\top\mathbb{E}[(ASS^\top A^\top)^{-1}]b \\
&= \frac{1}{l}\frac{l}{l-n-1}\mathrm{tr}(I - VV^\top)b^\top(AA^\top)^{-1}b \\
&= \frac{d-n}{l-n-1}b^\top(AA^\top)^{-1}b \\
&= \frac{d-n}{l-n-1}b^\top(U\Sigma^2U^\top)^{-1}b \\
&= \frac{d-n}{l-n-1}b^\top U\Sigma^{-2}U^\top b \\
&= \frac{d-n}{l-n-1}b^\top U\Sigma^{-1}V^\top V\Sigma^{-1}U^\top b \\
&= \frac{d-n}{l-n-1}(V\Sigma^{-1}U^\top b)^\top V\Sigma^{-1}U^\top b \\
&= \frac{d-n}{l-n-1}\|x_{OPT}\|_2^2.
\end{aligned}$$

□

18.3 Applications

We discuss how to improve the solution of the least squares problem by using the sketching technique, and its application in preprocessing and low-rank approximation. It mainly includes the following points: Sketch & Solve, Sketch + Precondition, Low-Rank Approximation. In addition, there are many issues, but the basic question is the approximation of $F((AA^\top)^{-1})$. For example,

- $(A^\top A)^{-1}A^\top b$
- $A(A^\top A)^{-1}A^\top b$
- $x^\top(A^\top A)^{-1}x$: When $x = a_i$, this is leverage score. When $x = e_i$, this gives the reference.
- $Tr(A^\top A)^{-1}$
- $\mathbb{E}(A^\top S^\top SA)^{-1}(\neq \mathbb{E}(A^\top A)^{-1})$

19 Tuesday, March 25: SPRING BREAK; No Class

20 Thursday, March 27: SPRING BREAK; No Class

21 Tuesday, April 1st (Scribe: Xingjian Wang)

Last Time Recap

We discussed sketching for solving linear systems.

Given $A \in \mathbb{R}^{n \times d}$, consider sketching with $\tilde{A} = SA$, where $S \in \mathbb{R}^{\ell \times n}$ is a random projection matrix.

- For Gaussian sketches, the smallest singular value of SU_d satisfies $\sigma_{\min}(SU_d) \sim 1 \pm \sqrt{d/\ell}$.
- Example: solve the sketched least squares problem

$$\tilde{x}_{\text{opt}} = \arg\min_x \|SAx - Sb\|_2$$

  Then,

$$\mathbb{E}[\|\tilde{x}_{\text{opt}} - x_{\text{opt}}\|_2] \leq \frac{1}{\sigma_d^2}\|Ax_{\text{opt}} - b\|_2, \quad \ell \geq d + 2$$

- Preconditioning: Use $Q = 8AR^{\dagger}$, where $R^\top R = (SA)^\top (SA)$, so that $\text{cond}(AR^\top) \leq 10$ if $\ell \geq 2d$.

Low-Rank Approximation and Sketching

A common guarantee:

$$\mathbb{E}[\|A - QQ^\top A\|_F^2] \leq \frac{k}{\delta_k}\|A - A_k\|_F^2$$

This motivates using sketching for matrix approximation. However, for inverse-related tasks (e.g., in regression or inference), bias becomes a concern.

Problem: Even if $\mathbb{E}[\tilde{A}^\top \tilde{A}] = A^\top A$, in general

$$\mathbb{E}[(\tilde{A}^\top \tilde{A})^{-1}] \neq (A^\top A)^{-1}$$

This is called *inversion bias*.

When Does the Inverse Matter?

- Appears in ridge regression: $(\tilde{A}^\top \tilde{A})^{-1}\tilde{A}^\top b$
- Confidence intervals: involves terms like $x^\top (A^\top A)^{-1} x$
- Statistical risk: trace of $(A^\top A)^{-1}$

So we often care about linear functionals of the inverse rather than the full matrix.

Note: Distributed averaging approaches can suffer from systematic bias in these quantities.

Johnson–Lindenstrauss (JL) Embedding

The JL lemma implies:

$$(1-\varepsilon)A^\top A \preceq (SA)^\top (SA) \preceq (1+\varepsilon)A^\top A$$

This gives relative spectral guarantees and avoids inversion entirely.

To achieve this, the sketch dimension must be:

$$\ell \gtrsim \frac{d \log d}{\varepsilon}$$

Motivation: Avoiding inversion in high dimensions is computationally attractive.

Covariance Estimation Setup

- Population covariance: what's stored on disk, i.e., $A \in \mathbb{R}^{n \times d}$
- Sample: sketched matrix $SA \in \mathbb{R}^{\ell \times d}$
- Interested in estimating $(A^\top A)^{-1}$

Random Matrix Theory (RMT)

- **Resolvent:** $Q(z) = (A^\top A - zI)^{-1}$, for $z \in \mathbb{C} \setminus \mathbb{R}^+$
- Stieltjes transform: normalized trace of the resolvent
- Traditional RMT: large n, p limit
- In ML: interested in finite-sample nonasymptotic behavior

Bias Correction with Gaussian Sketches

With Gaussian sketching, we have:

$$\text{If } \gamma = \frac{\ell}{d} I, \quad \mathbb{E}[(\gamma \tilde{A}^\top \tilde{A})^{-1}] = (A^\top A)^{-1}$$

This is a key unbiasedness property of Gaussian sketches.

Why is it easier to correct? Gaussian projections preserve more structure and concentration, which helps average out the inversion bias.

Question: Anything similar for the other sketches?

2 of 5 take a look:

Rademacher, subgaussian, Hadamard-based, sparse.

Hold?	×	×	×	×	✓
Similar?	×	✓	×	×	✓

Subgaussian sketches (including Gaussian) are robust. Two-height subgaussians may also work.

Nearly Unbiased Estimators (NUE)

Definition: $\hat{C}$ is an (ε, δ) unbiased estimator of PSD matrix C if:

$$\mathbb{E}[\hat{C} \mid \text{good event }] = C, \quad \text{and } \hat{C} \preceq \text{const} \cdot C$$

Fact: If $S \in \mathbb{R}^{\ell \times n}$ is a subgaussian sketch, then

$$\mathbb{E}\left[\left(\frac{\ell}{d} A^\top S^\top S A\right)^{-1}\right] \approx (A^\top A)^{-1}, \quad \text{with } \varepsilon = O\left(\frac{d}{\ell}\right)$$

Hence, we obtain an NUE:

$$\left(\frac{\ell}{\ell - d} A^\top S^\top S A\right)^{-1}$$

Compare this with JL-style bounds: JL guarantees a spectral sandwich, NUE gives an expected value sandwich.

Variance Considerations

We care about estimators $\tilde{A}$ such that inversion bias is small and variance is controlled:

$$\text{Want: } \operatorname{Var}[S_i^\top B S_i] \leq \operatorname{Tr}(B^2), \quad \forall B \in \mathbb{R}^{n \times n} \text{ PSD}$$

with $PBP^\top = \text{Proj}_{\text{colspace}(A)}$, and ideally, bias$^2 \ll$ variance.

Application: Model Averaging

Given $g \in O(\ell)$, let $S_1, \ldots, S_g$ be independent subgaussian sketches. Then:

$$\frac{1}{g}\sum_{i=1}^{g}\left(\frac{\ell}{kd}A^\top S_i^\top S_i A\right)^{-1} \approx (1\pm\varepsilon)(A^\top A)^{-1}$$

Example:

Solve $\tilde{x}_i = \arg\min \|S_i A x - S_i b\|$, average them:

$$\tilde{x}_{\text{avg}} = \frac{1}{g}\sum \tilde{x}_i \Rightarrow \mathbb{E}[\|A\tilde{x}_{\text{avg}} - b\|] \le (1+\varepsilon)\|Ax_{\text{opt}} - b\|$$

22 Thursday, April 3rd (Scribe: TBD)

23 Tuesday, April 8th (Scribe: Luke Triplett)

Today's main topic is random matrix theory.

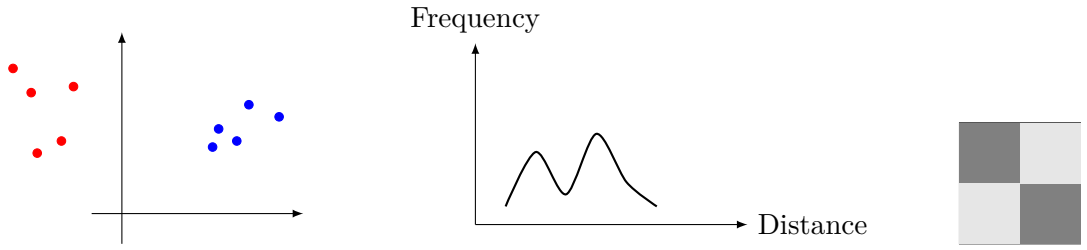
First, let's briefly recap the discussion from last time. A nice property of Gaussian projections is that they don't have inversion bias (more precisely, the inversion bias is correctable). In particular, it holds that if $A \in \mathbb{R}^{n\times d}$ and $S \in \mathbb{R}^{\ell\times d}$ is a properly scaled Gaussian projection, then $\mathbb{E}[(\gamma A^T S^T S A)^{-1}] = (A^T A)^{-1}$, where $\gamma = \frac{\ell}{\ell-d-1}$.

The previous two classes have revolved around this result and how it can be extended (approximately) to subgaussians and Leverage score sparsified embeddings (LESS).

Now, we will move on to RMT. Broadly speaking, RMT studies the eigenvalues of random matrices. Physicists are historically interested in RMT because eigenvalues of the energy operator give the possible energy levels of a system. In RMT, there are famous eigenvalue distributions with characteristics properties, such as the Marchenko-Pastur (MP) distribution.

There is a delineation between RMT for RandNLA and for ML. In RandNLA, we want a subspace embedding, and typically have eigenvalue distributions with a spectral gap. On the other hand, in ML we are often interested in the eigenvectors (not just eigenvalues) and don't have a spectral gap.

We start by looking at a toy problem. Suppose that we have sets of points in $\mathbb{R}^n$ and that within each set, the points are drawn from some distribution. We want to know whether we can figure out that the points came from two different distributions looking at pairwise similarities. Of course, if the distributions underlying X and Y are sufficiently different (compared to the variance), it will be obvious from looking at the histogram. But, it turns out we can tell more reliably by looking at spectral information. Define the Gram matrix which measures pairwise similarities between points: $K_{ij} = e^{\beta||x_i - x_j||^2}$. Then the second eigenvector will reveal information about the clusters.



(a) Data in 2D (b) Histogram of Distances (c) Kernel Matrix Heat Map

Figure 1: (a) 2D scatter plot of data; (b) histogram showing intra- vs inter-cluster distances; (c) heat map of the kernel matrix. Eigenvectors of this matrix (particularly v_2) help classify points into clusters.

Now, we give some definitions

23.1 Resolvent.

Let $M \in \mathbb{R}^{n \times n}$ be a symmetric (or Hermitian) matrix. The *resolvent* $Q_M(z)$ is defined by

$$Q_M(z) \;=\; (M - zI)^{-1},$$

for all complex z not in the spectrum of M. In other words, z is restricted so that $\det(M - zI) \neq 0$. The resolvent is often called a *regularized inverse*, particularly useful when some eigenvalues of M could be close to z or near zero.

23.2 Empirical Spectral Distribution (ESD).

Let the eigenvalues of M be $\lambda_1, \lambda_2, \ldots, \lambda_n$. The *empirical spectral distribution* (ESD), denoted $\mu_M(\lambda)$, is defined as

$$\mu_M(\lambda) \;=\; \frac{1}{n} \sum_{i=1}^{n} \delta_{\lambda_i}(\lambda),$$

where δ_{λ_i} is the Dirac delta at λ_i. Equivalently, μ_M is the probability measure that places a mass of $\frac{1}{n}$ at each eigenvalue of M.

23.3 Stieltjes Transform.

For a probability measure μ on the real line, its *Stieltjes transform* $m_\mu(z)$ is defined by

$$m_\mu(z) \;=\; \int_{\mathbb{R}} \frac{1}{\lambda - z}\, d\mu(\lambda), \quad z \in \mathbb{C} \setminus \operatorname{supp}(\mu).$$

In our case, $\mu = \mu_M$ is the ESD of M. Thus,

$$m_{\mu_M}(z) \;=\; \int_{\mathbb{R}} \frac{1}{\lambda - z}\, d\mu_M(\lambda) \;=\; \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\lambda_i - z} = \frac{1}{n} Tr(Q(z)).$$

A useful property is that if z is at least some positive distance δ away from the real line segment containing all eigenvalues of M, then

$$\left| m_{\mu_M}(z) \right| \;=\; \left| \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\lambda_i - z} \right| \;\leq\; \frac{1}{\delta}.$$

In other words, the magnitude of $m_{\mu_M}(z)$ is controlled by how far z is from the support of μ_M.

It was also mentioned that we can define Stieltjes transforms for a matrix measure $\Sigma : \mathbb{R} \to \mathbb{R}^{n \times n}$ as

$$m_\Sigma(z) \;=\; \int_{\mathbb{R}} \frac{1}{\lambda - z}\, d\Sigma(\lambda).$$

This will now output another matrix measure, i.e. will output an $n \times n$ matrix for each z.

23.4 Inverse Stieltjes Transform.

Given $m_{\mu_M}(z)$, the measure μ_M can (in principle) be recovered by integrating the imaginary part of $m_{\mu_M}(z)$ along a contour crossing the real axis:

$$\mu_M(\lambda) \;=\; -\frac{1}{\pi}\lim_{\epsilon\to 0^+}\mathrm{Im}\left(m_{\mu_M}(\lambda+i\,\epsilon)\right).$$

Poles of the resolvent $(M-zI)^{-1}$ or $m_{\mu_M}(z)$ occur at the eigenvalues $\{\lambda_i\}$.

23.5 Cauchy Integral Formula.

Given a simple closed curve Γ which encloses a region D, and an analytic function $f(z)$,

$$f(z_0)=\frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{z-z_0}.$$

for all $z_0\in D$. This theorem will allow us to relate certain matrix observables to its resolvent.

23.6 Linear Spectral Statistics.

Any linear spectral statistic can be represented as an integral of the resolvent's trace. For example,

$$\frac{1}{n}\sum_{i=1}^{n} f(\lambda_i)=\frac{1}{2\pi i n}\int_\Gamma f(z)Tr(Q_M(z))dz=\frac{1}{2\pi i}\int_\Gamma f(z)m_{\mu_M}(z)dz$$

For another example, let $M=U\Lambda U^T=\sum\lambda_i u_i u_i^T$. Then,

$$Q_M(z)=\sum\frac{u_iu_i^T}{\lambda_i-z},$$

so

$$u_iu_i^T=\frac{-1}{2\pi i}\int_{\Gamma_{\lambda_i}}Q_M(z)dz.$$

If we choose Γ_{λ_i} to be a contour which encloses λ_i and none of the other eigenvalues. This is true because the other terms of the integrand are analytic within the contour, so the contour integrals are 0.

We can see that this allows us to generalize the idea of projections for example onto random vectors/subspaces. Along these lines, consider v deterministic and u_i as before, then

$$|v^Tu_i|^2=\frac{-1}{2\pi i}\int_{\Gamma_{\lambda_i}}v^TQ_M(z)vdz.$$

23.7 Deterministic Equivalent.

In random matrix theory, it is useful to define a weak notion of convergence called deterministic equivalence.

Given a random matrix $Q \in \mathbb{R}^{n\times n}$, the matrix $\bar{Q} \in \mathbb{R}^{n\times n}$ is a deterministic equivalent of Q if for any sequence of unit norm matrices $\{A_n\}$ and unit vectors $a_n, b_n \in \mathbb{R}^n$

$$\frac{1}{n}\operatorname{Tr}(A_n(Q-\bar{Q})) \to 0,$$
$$a^T(Q-\bar{Q})b \to 0.$$

To get an intuition, typically $\bar{Q} = \mathbb{E}[Q]$ will be a deterministic equivalent under mild conditions, but we don't often use it and instead look for something simpler.

24 Thursday, April 10th (Scribe: Sultan Daniels)

Facts

- $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$
- If $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{n\times p}$, $z \in \mathbb{C} \setminus \{\{0\}, \{\text{eigenvalues}(AB)\}\}$, then $A(BA - zI_p)^{-1} = (AB - zI_n)^{-1}A$
- $-\det(BA - zI_n) = \det(AB - zI_p)(-z)^{p-n}$
- $\operatorname{Tr}(Q_{AB}^{(z)}) = \operatorname{Tr}(Q_{BA}(z)) + \frac{n-p}{z}$
- $m_{\mu_{AB}}(z) = \frac{n}{p}m_{\mu_{BA}}(z) + \frac{n-p}{pz}$
- **Sherman-Morrison-Woodbury:** $(A + UV^\top)^{-1} = A^{-1} - A^{-1}U(I_n + V^\top A^{-1}U)^{-1}V^\top A^{-1}$
- Let $A, M \in \mathbb{R}^{p\times p}$, $u \in \mathbb{R}^p$, then
  $|\operatorname{tr}\left(A(M + \tau uu^\top - zI_p)^{-1}\right) - \operatorname{tr}\left(A(M - zI_p)^{-1}\right)| \le \frac{\|A\|_2}{|z|}$

24.1 Trace Lemma

Let $x \in \mathbb{R}^p$, x_i are independent with $\operatorname{Var}(x_i) = 1$ and $E[x_i] = 0$. We want a tail property s.t. $E[|x_i|^2] \le \mathcal{V}_N$.

Given $X \in \mathbb{R}^{p\times n}$, let $\mathbb{C}(x)$ be a deterministic function of x. Let $f(\phi(x))$ be a 1-Lipschitz and scalar $f(u)$. We say that $\bar{X}_\phi$ is z.

24.2 High-dimension Equivalent

$$|f(\phi(x)) - f(\bar{X})| \le \varepsilon(n, p)$$

with probability at least $1 - \delta(n, p)$, where n, p are dimensions.

Challenges

- **Challenge 1:** For X, ε across dimensions,

  $$\text{L-cc Concentrations: } f(\phi(x + e))$$

  $$E[f(\phi(x; \varepsilon))]$$
- **Challenge 2:** Analysis of certain deterministic quantities.
- **Challenge 3:** Nonlinearities in the context.

Facts about Scalar Random Variables

-
$$E[|x|^p] = \int_0^\infty pt^{p-1}P(|x| > t)dt$$

- $$P(|x| \geq t) \leq \exp(-\lambda t) M(\lambda)$$
- For a random variable X with mean $\mu = E[X]$ and variance $\theta^2 = \text{Var}(X)$:
$$P(|x_1| \leq t\theta^{-1}) \leq t^{-2}$$
- If x is sub-Gaussian:
$$P(|x - \mu| \geq t\sigma) < e^{-t^2/2}$$
- With high probability:
$$X \in [\mu - \sigma\sqrt{\log(1/\delta)},\ \mu + \sigma\sqrt{\log(1/\delta)}]$$

Variance of Sample Mean

Given a collection of scalar random variables X_i with mean μ and variance σ^2:

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n}$$

As $n \to \infty$, the variance goes to zero.

Random Vectors

Consider random vectors $x \in \mathbb{R}^n$. Fact: Random vectors do not concentrate about their mean as tightly as scalars.

Claim: Linear Scalar Observations

Let $f(x) = \frac{\boldsymbol{\ell}^\top x}{n}$, with $E[x_i] = \mu$, $\text{Var}(x_i) = \sigma^2$:

$$P\left(|f(x) - \mu| \geq \frac{\sigma}{\sqrt{n}}\right) \leq t^2$$

$$f(x) \in \left[\mu - \frac{\sigma}{\sqrt{\sigma n}},\ \mu + \frac{\sigma}{\sqrt{\sigma n}}\right]$$

with high probability.

25 Tuesday, April 15th (Scribe: Aneesh Durai (2nd))

Fact (Polarization Identity)

$$x^T y \;=\; \frac{1}{2}\Big(\|x+y\|_2^2 - \|x\|_2^2 - \|y\|_2^2\Big).$$

Example

Consider fixed $y \in \mathbb{R}^n$ with $\|y\|_2 = 1$, and random $x \in \mathbb{R}^n$ with $\mathbb{E}\|x\|_2^2 = 1$. Then (by LLN/CLT):

$$\begin{aligned} x^T y &\approx 0 \;+\; \tfrac{1}{\sqrt{n}}\, N(0,1), \\ \|x\|_2^2 = x^T x &\approx 1 \;+\; \tfrac{1}{\sqrt{n}}\, N(0, m_2 - 1), \\ \|x-y\|_2^2 = x^T x + y^T y - 2\, x^T y \;&=\; Z \;+\; O\big(n^{-1/2}\big). \end{aligned}$$

Notes (Matrices)

Similarly, for $X, Y \in \mathbb{R}^{m\times n}$:

$$\begin{aligned} \mathrm{Tr}(X^T Y) &= \tfrac{1}{2}\Big(\|X\|_F^2 + \|Y\|_F^2 - \|X-Y\|_F^2\Big), \\ \mathrm{Tr}(X^T Y) &= 0 \;+\; \tfrac{1}{\sqrt{mn}}\, N(0,1), \\ \|X\|_F^2 &= 1 \;+\; \tfrac{1}{\sqrt{mn}}\, N(0, m_2 - 1), \\ \|X-Y\|_F^2 &= \|X\|_F^2 + \|Y\|_F^2 + O\big((mn)^{-1/2}\big). \end{aligned}$$

On Matrix-Norm Equivalence

Let $x_1, \ldots, x_n \in \mathbb{R}^p$ be i.i.d. with $\mathbb{E}[x_i] = 0$ and $\mathbb{E}[x_i x_i^T] = I_p$. The sample covariance is

$$\widehat{C} \;=\; \frac{1}{n}\sum_{i=1}^n x_i x_i^T.$$

Observe two very different regimes:

1. **Entrywise (max-norm) control.**

   $$\|\widehat{C} - I_p\|_{\max} = \max_{j,k}\big|\widehat{C}_{jk} - \delta_{jk}\big| \;\xrightarrow{\text{a.s.}}\; 0 \quad \text{as } n \to \infty,$$

   by a coordinate-wise Law of Large Numbers (and Gaussian or sub-Gaussian tail CLT). In fact one even shows with high probability

   $$\|\widehat{C} - I_p\|_{\max} = O\Big(\sqrt{\tfrac{\log p}{n}}\Big).$$

But note $\|\cdot\|_{\max}$ is only entrywise control.

2. **Spectral (operator)-norm control.**

$$\|\widehat{C} - I_p\|_2 = \sup_{\|v\|=1} \left|v^T(\widehat{C} - I_p)v\right| \;=\; \max_{\text{eig. } \lambda} \left|\lambda(\widehat{C}) - 1\right|.$$

Even though each $\widehat{C}_{jk}$ is well concentrated, the fact that $\widehat{C}$ has at most n nonzero eigenvalues means $\|\widehat{C} - I_p\|_2$ typically stays bounded away from zero if $n < p$. Concretely, one shows by matrix-concentration (e.g. matrix Bernstein)

$$\|\widehat{C} - I_p\|_2 = O\Big(\sqrt{\tfrac{p}{n}} + \tfrac{\log p}{n}\Big) \quad \text{with high probability,}$$

which *blows up* when p is of the same order as—or larger than—n.

Why care about the spectral norm?

- *Eigenvalue stability.* By Weyl's inequality, a small $\|\widehat{C} - I\|_2$ implies every eigenvalue of $\widehat{C}$ lies near 1.
- *Eigenvector/subspace stability.* By the Davis–Kahan $\sin\Theta$ theorem, small $\|\widehat{C} - I\|_2$ guarantees that the principal subspaces of $\widehat{C}$ remain close to those of I (i.e. any fixed subspace).

Ways to Characterize the Sample Covariance Matrix

Classical regime ($n \gg p$)

- **Asymptotic:** $n \to \infty$ with p fixed $\Longrightarrow$ both $\|\cdot\|_{\max}$ and $\|\cdot\|_2$ tend to zero a.s.
- **Non-asymptotic:** $n \gg p$, finite $\Longrightarrow$ high-probability bounds of the form

$$\|\widehat{C} - I\|_2 \;\lesssim\; \sqrt{\tfrac{p}{n}} + \tfrac{\log p}{n}.$$

Proportional regime ($n \sim p$)

- **Asymptotic:** $n, p \to \infty$, $p/n \to c \in (0, \infty) \Longrightarrow$ need RMT tools (MP law, deterministic equivalents) to control $\|\cdot\|_2$.
- **Non-asymptotic:** $n \sim p \gg 1 \Longrightarrow$ one seeks high-probability finite-sample analogues of RMT results (e.g. non-asymptotic deterministic equivalents, local laws).

Classical Asymptotic

Let

$$\widehat{C} \;=\; \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T.$$

By the Law of Large Numbers,

$$\widehat{C} \xrightarrow{\text{a.s.}} I_p,$$

so

$$\left\|\widehat{C} - I_p\right\|_{\max} \xrightarrow{\text{a.s.}} 0.$$

Since

$$\left\|\widehat{C} - I_p\right\|_2 \le p \left\|\widehat{C} - I_p\right\|_{\max},$$

we also obtain

$$\left\|\widehat{C} - I_p\right\|_2 \xrightarrow{\text{a.s.}} 0.$$

Classical Non-Asymptotic

For finite $n \gg p$ one can show (via matrix concentration) that with high probability

$$\left\|\widehat{C} - I_p\right\|_2 \le C \max(\delta_1, \delta_2),$$

where typically

$$\delta_1 \sim \sqrt{\frac{p}{n}} \quad \text{and} \quad \delta_2 \sim \frac{\log n}{n}$$

(or equivalently $\delta \approx \sqrt{p/n} + O(\log n/n)$).

Proportional Regime: Traditional RMT & Marchenko–Pastur Law

In the proportional regime $p/n \to c \in (0, \infty)$, one has the following classical random-matrix result:

- The *limiting spectral distribution* of the sample covariance

  $$\widehat{C} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$$

  converges to the *Marchenko–Pastur (MP) distribution*.
- This MP law is fully determined by the aspect ratio

  $$c = \frac{p}{n}$$

  and the population variance σ^2. Its density on $[\lambda_-, \lambda_+]$ is

  $$f_{\text{MP}}(\lambda) = \frac{1}{2\pi c \sigma^2} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda}, \quad \lambda_\pm = \sigma^2\big(1 \pm \sqrt{c}\big)^2.$$

- In the "classical" limit p fixed, $n \to \infty$ (so $c \to 0$), the MP distribution collapses onto a point mass at σ^2, recovering the Law of Large Numbers regime.
- Understanding the MP law is a cornerstone of high-dimensional random-matrix theory and underpins many modern results in statistics and data science.

Theorem 5 (Marchenko–Pastur)**.** Let $X \in \mathbb{R}^{p\times n}$ have i.i.d. columns x_i, each satisfying

$$\mathbb{E}[x_i] = 0, \qquad \mathbb{E}[x_i x_i^T] = \sigma^2 I_p,$$

and having subgaussian tails. Define the resolvent

$$Q(z) \;=\; \Big(\tfrac{1}{n}\, XX^T - z\, I_p\Big)^{-1}, \quad z \in \mathbb{C}^+.$$

As $n, p \to \infty$ with $p/n \to c \in (0,\infty)$, one has the *deterministic equivalent*

$$Q(z) \;\longrightarrow\; \overline{Q}(z) \;=\; m(z)\, I_p,$$

where $m(z)$ is the unique solution in $\mathbb{C}^+$ of the quadratic equation

$$z\, m(z)^2 - \big(1 - c - z\big)\, m(z) + 1 \;=\; 0.$$

Equivalently, $m(z)$ is the Stieltjes transform of a probability measure μ with mass $\big(1 - c^{-1}\big)^+$ at zero and density

$$\mu'(x) = \frac{1}{2\pi\sigma^2 c\, x}\sqrt{\big(x - \sigma^2(1-\sqrt{c})^2\big)^+\,\big(\sigma^2(1+\sqrt{c})^2 - x\big)^+},$$

supported on $\big[\sigma^2(1-\sqrt{c})^2,\ \sigma^2(1+\sqrt{c})^2\big]$. In particular, when $\sigma = 1$ this is the standard MP law.

Proof Idea

Define

$$Q(z) \;=\; \Big(\tfrac{1}{n}\, XX^T - z\, I_p\Big)^{-1}, \qquad \overline{Q}(z) \;=\; F(z),$$

where $F(z)$ is the candidate deterministic equivalent. We "guess" that $\overline{Q}(z) = F(z)$, and then check by the resolvent identity:

$$Q(z) - \overline{Q}(z) = Q(z)\Big(\overline{Q}(z)^{-1} - Q(z)^{-1}\Big)\,\overline{Q}(z) = Q(z)\Big(F(z) + zI_p - \tfrac{1}{n}\sum_{i=1}^{n} x_i x_i^T\Big)\overline{Q}(z).$$

If $\overline{Q}(z)$ is indeed the correct deterministic equivalent, then in particular we must have

$$\frac{1}{p}\mathrm{trace}\big[A\,(Q(z) - \overline{Q}(z))\big] \;\longrightarrow\; 0 \quad \text{for every fixed } A \text{ with } \|A\|_2 = 1.$$

Equivalently,

$$\begin{aligned}\frac{1}{p}\mathrm{trace}\big[A\,(Q - \overline{Q})\big] &= \frac{1}{p}\mathrm{trace}\Big[A\,Q\big(F + zI_p\big)\overline{Q}\Big] - \frac{1}{pn}\sum_{i=1}^{n}\mathrm{trace}\big[A\,Q\,x_i x_i^T\,\overline{Q}\big] \\ &= \frac{1}{p}\mathrm{trace}\Big(\big(F + zI_p\big)\overline{Q}\,A\,Q\Big) - \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p}\,x_i^T\big(\overline{Q}\,A\,Q\big)x_i.\end{aligned}$$

By a matrix–trace lemma (controlling the average of the quadratic forms $x_i^T[\cdots]x_i$) and norm bounds on $F, A, Q, \overline{Q}$, each term vanishes as $n, p \to \infty$. Hence

$$\frac{1}{p}\text{trace}\big[A\,(Q(z) - \overline{Q}(z))\big] \;\longrightarrow\; 0,$$

as required.

Note: The key concentration step is that $\frac{1}{p}\, x_i^T\, \overline{Q}\, A\, Q\, x_i$ concentrates around its trace, allowing one to replace the random quadratic form by $\frac{1}{p}\text{trace}(\overline{Q}\, A\, Q)$.

Let

$$Q(z) \;=\; \Big(\tfrac{1}{n} X X^T - z I_p\Big)^{-1}, \quad Q_{-i}(z) \;=\; \Big(\tfrac{1}{n}\sum_{j\neq i} x_j x_j^T - z I_p\Big)^{-1}.$$

Since $Q_{-i}(z)$ is independent of x_i, the Sherman–Morrison identity gives

$$Q(z)\, x_i = \frac{Q_{-i}(z)\, x_i}{1 + \frac{1}{n}\, x_i^T\, Q_{-i}(z)\, x_i}.$$

We consider the quadratic form

$$\frac{1}{p}\, x_i^T\, \overline{Q}(z)\, A\, Q(z)\, x_i = \frac{1}{p}\, \frac{x_i^T\, \overline{Q}(z)\, A\, \big(Q_{-i}(z) x_i\big)}{1 + \frac{1}{n}\, x_i^T\, Q_{-i}(z)\, x_i}.$$

By the matrix-trace lemma (and since $Q_{-i}(z) \approx Q(z)$), each numerator concentrates to $\frac{1}{p}\text{trace}\big(\overline{Q}(z)\, A\, Q(z)\big)$ and each denominator to $1 + \frac{1}{n}\text{trace}Q(z)$. Hence

$$\frac{1}{p}\, x_i^T\, \overline{Q}(z)\, A\, Q(z)\, x_i \;\approx\; \frac{\frac{1}{p}\text{trace}\big(\overline{Q}\, A\, Q\big)}{1 + \frac{1}{n}\text{trace}Q}.$$

Recall from above that

$$\frac{1}{p}\text{trace}\big[A\,(Q - \overline{Q})\big] = \frac{1}{p}\text{trace}\big((F + zI_p)\, \overline{Q}\, A\, Q\big) - \frac{1}{n}\sum_{i=1}^{n} \frac{1}{p}\, x_i^T\, \overline{Q}\, A\, Q\, x_i.$$

Replacing each quadratic form by its trace-lemma approximation yields

$$\frac{1}{p}\text{trace}\big[A\,(Q - \overline{Q})\big] \;\approx\; \frac{1}{p}\text{trace}\big((F + zI_p)\, \overline{Q}\, A\, Q\big) - \frac{1}{1 + \frac{1}{n}\text{trace}Q}\; \frac{1}{p}\text{trace}\big(\overline{Q}\, A\, Q\big).$$

For this to vanish for all A, we must have

$$F(z) \;=\; \frac{1}{z + \frac{1}{n}\text{trace}Q(z)}\; I_p,$$

which is exactly the fixed-point equation determining the Stieltjes transform.

To close the loop, Taking $A = I_p$, the requirement

$$\frac{1}{p}\text{trace}\big[A\,(Q(z) - \overline{Q}(z))\big] \;\longrightarrow\; 0$$

becomes

$$m(z) \;=\; \frac{1}{p}\text{trace}Q(z) \;\approx\; \frac{1}{p}\text{trace}\overline{Q}(z) \;=\; \frac{1}{p}\text{trace}F(z).$$

Hence

$$\frac{1}{p}\text{trace}Q(z) \;=\; m(z) \;=\; \frac{1}{-\,z \;+\; \frac{1}{1+\frac{p}{n}\,\frac{1}{p}\text{trace}Q(z)}} \;\approx\; \frac{1}{-\,z \;+\; \frac{1}{1+c\,m(z)}}.$$

Thus we set

$$\overline{Q}(z) = F(z) = m(z)\,I_p, \quad m(z) = \Big(-\,z + \tfrac{1}{1+c\,m(z)}\Big)^{-1}.$$

Equivalently, $m(z)$ satisfies the quadratic

$$z\,c\,m(z)^2 \;-\; (1-c-z)\,m(z) \;+\; 1 \;=\; 0,$$

whose solutions are

$$m(z) = \frac{1-c-z}{2cz} \;\pm\; \frac{1}{2cz}\sqrt{(1+c-z)^2 - 4c}\,.$$

Next Steps

Similar methods can be used to derive *non-asymptotic deterministic equivalents* for the resolvent $Q(z)$.

26 Thursday, April 17th (Scribe: XXX)

27 Tuesday, April 22nd (Scribe: Divit Rawal)

27.1 Stochastic Optimization

We consider optimization problems of the form

$$\min_{x\in\mathbb{R}^d} f(x) \quad \text{where} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} \psi_i(x).$$

Here each term $\psi_i(x)$ may depend on a random data point or sample, making f a stochastic objective.

27.1.0.1 Stochastic Gradient Descent (SGD) A prototypical algorithm for such problems is *Stochastic Gradient Descent*:

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

In practice, one often approximates the full gradient by sampling a subset S of indices:

$$\hat{g}_t = \frac{1}{|S|}\sum_{j\in S} \nabla\psi_j(x_t), \quad x_{t+1} = x_t - \eta_t\, \hat{g}_t.$$

Although straightforward, vanilla SGD suffers from several drawbacks:

- **Step-size tuning:** The sequence $\{\eta_t\}$ is typically unknown and difficult to choose.
- **Instability:** Poor choice of η_t can lead to diverging iterates.
- **High variance:** The random gradient estimate $\hat{g}_t$ often has large variance, slowing convergence.
- **Conditioning:** Ill-conditioned problems exacerbate these issues.

27.1.0.2 Sketch-and-Solve Techniques from Randomized Numerical Linear Algebra (RandNLA) To mitigate these problems, one can incorporate *sketching* and *preconditioning* ideas:

- *Iterative sketching* (e.g. PW-SGD)
- *Newton's Sketch* (a sketchy second-order method)
- *Sketch & project*
- *Sketch & precondition*

27.2 SSN (Sketched Second-order Newton)

In classical Newton's method, the update is

$$x_{t+1} = x_t - H_t^{-1} g_t, \quad g_t = \nabla f(x_t), \;\; H_t = \nabla^2 f(x_t).$$

This uses full Hessian information but is expensive. In contrast, SGD uses only first-order information with

$$x_{t+1} = x_t - \eta_t \, \hat{g}_t, \quad \mathbb{E}[\hat{g}_t] = g_t,$$

but suffers from high variance $\mathbb{E}\big[\|\hat{g}_t - g_t\|^2\big]$.

27.2.0.1 Reducing Variance via RandNLA

RandNLA offers two main tools:

- **Weighted gradient sampling:** Importance-sampling gradients to lower variance.
- **Sketching-based preconditioners:** Approximate the Hessian inverse cheaply.

27.3 PW-SGD (Preconditioned & Weighted SGD)

Specializing to the least-squares objective

$$f(x) = \frac{1}{n} \, \|Ax - b\|_2^2,$$

PW-SGD proceeds as follows:

1. Compute a sketch SA of A using a suitable sketching matrix S.
2. Perform a QR-factorization of SA to obtain R.
3. Estimate leverage scores $\{\hat{\ell}_i\}$, where

   $$\hat{\ell}_i \approx (A \, (A^T A)^{-1} A^T)_{ii},$$

   to form sampling probabilities.
4. For $t = 0, 1, \ldots, T-1$:
   - Sample indices according to $\hat{\ell}_i$, and compute

     $$\hat{g}_t = \sum_{i=1}^{n} \frac{n}{\hat{\ell}_i} \, \nabla \psi_i(x_t).$$

   - Update with preconditioning:

     $$x_{t+1} = x_t - \eta_t \, R R^T \, \hat{g}_t.$$

27.3.0.1 Convergence Guarantees for PW-SGD

PW-SGD enjoys two types of guarantees:

1. *Error decay over iterations* (TCS bounds).
2. *Expected suboptimality bound:*

   $$\mathbb{E}\big[f(x_t) - f(x^\star)\big] \;\le\; \frac{f(x_0)}{1 + (st)/(c \, \eta \, d)}$$

   for some constant c and sampling size s. To reach ϵ-optimality requires

   $$t = O\big(\tfrac{d}{s \, \epsilon}\big),$$

   which improves on vanilla SGD's $t = O\big(\tfrac{n \, \kappa^2}{s \, \epsilon}\big)$ when the condition number κ is large.

27.4 Newton's Method

Newton's method seeks to minimize a second-order approximation of f around x_t:

$$g_t = \nabla f(x_t), \quad H_t = \nabla^2 f(x_t), \quad x_{t+1} = x_t - \eta_t \, H_t^{-1} g_t.$$

27.4.0.1 Example: Regularized Empirical Risk

For

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(a_i^T x) + \frac{\lambda}{2} \, \|x\|_2^2,$$

one has

$$\nabla^2 f(x) = A^T D_x \, A, \quad D_x = \operatorname{diag}\big(\ell_1''(a_1^T x), \ldots, \ell_n''(a_n^T x)\big).$$

Forming and inverting this Hessian costs $O(nd^2)$ per iteration.

27.4.1 Newton Sketch

To accelerate, we apply a sketch S_t at each iteration:

$$\tilde{A} = S_t \, D_x^{1/2} \, A, \quad \hat{H} = \tilde{A}^T \tilde{A} + \lambda I,$$

and update via

$$x_{t+1} = x_t - \eta_t \, \hat{H}^{-1} \, g_t.$$

This reduces dimension from n to the sketch size, lowering per-iteration cost.

27.5 Sketch & Project (Kaczmarz Method)

A classical method for solving linear systems $Ax = b$ is the (randomized) Kaczmarz algorithm:

1. Start from an initial guess x_0.
2. For $t = 0, 1, \ldots, \tau - 1$:
   - Randomly select row index $i \in \{1, \ldots, m\}$ with probability $p_i \propto \|a_i\|_2^2$.
   - Project onto the hyperplane $a_i^T x = b_i$:

$$x_{t+1} = x_t - \frac{a_i^T x_t - b_i}{\|a_i\|_2^2} \, a_i.$$

One can show linear convergence in expectation:

$$\mathbb{E}\big[\|x_t - x^\star\|_2^2\big] \;\le\; \Big(1 - \tfrac{\sigma_{\min}^2(A)}{\|A\|_F^2}\Big)^t \|x_0 - x^\star\|_2^2.$$

28 Thursday, April 24 (Scribe: Jingrong Guan)

Announcement:
- **Project** due on **Thursday, 5/13**.
- **Next Thursday, 5/1**, will be over **Zoom**: https://berkeley.zoom.us/j/4633527821.
- **No** Final for this semester.

This lecture continues to explore topics related to gradient descent, especially its behavior and improvements when solving LS problems. Specifically, the standard form of an LS problem is:

$$\min_x \frac{1}{2}\|Ax - b\|_2^2$$

where $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$, and $x \in \mathbb{R}^d$. Solving this problem typically requires $\mathcal{O}(nd^2)$ time.

Before diving into today's main content, we quickly revisited basic approaches to use RandNLA methods more generally:

1. *Sketch + Solve*: We randomly construct a smaller LS problem, and then solve it using a traditional NLA method. The computational cost consists of the time for random projection plus the cost of solving the subproblem, which can be around sd^2 or $sd\log(1/\epsilon)$, where s is the sketch dimension. The random projection time can be characterized as $\mathcal{O}(\frac{1}{\epsilon}nd\log d)$.
2. *Iterative Sketching*: we repeatedly sketch/sub-sample the problem randomly, and then iteratively refine the estimate using a convex optimization method.
3. *Sketch + Precondition*: We randomly construct an equivalent but well-conditioned problem, and then solve it using a traditional deterministic NLA iterative method. The total cost includes the time needed for random projection and for building the preconditioner, followed by faster iterations on the preconditioned system, which costs $\mathcal{O}(\frac{1}{\epsilon}nd)$.

28.1 Gradient Descent Algorithm

Consider the unconstrained minimization of $f : \mathbb{R}^d \to \mathbb{R}$, a differentiable function, that is, solving

$$\min_{x\in\mathbb{R}^d} f(x)$$

To solve this, we use the Gradient Descent algorithm. It works by taking steps proportional to the negative gradient direction, which is the direction of largest instantaneous decrease.

Algorithm 11 Gradient Descent Algorithm

1: **Input:** differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, step size μ_t, iteration number $\mathcal{T}$
2: **Initialize:** $x_0 \in \mathbb{R}^d$
3: **for** $t = 0, 1, \ldots, \mathcal{T}$ **do**
4: $\quad x_{t+1} = x_t - \mu_t \nabla f(x_t)$
5: **end for**
6: **Output:** $x_{\mathcal{T}}$

If the step size μ_t is sufficiently small and the gradient $\nabla f(x_t) \neq 0$, then the value of f decreases at each iteration. Moreover, if f is convex, gradient descent converges to a global minimum under mild conditions. To better illustrate these concepts, we now present several examples and visualizations:

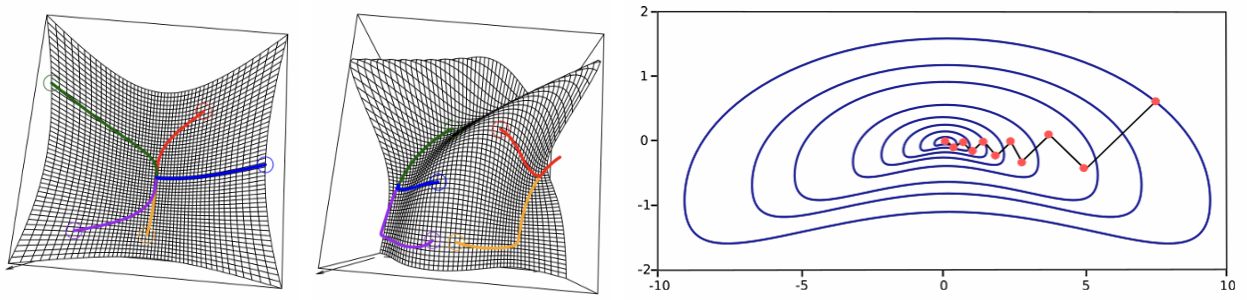


Figure 2: Examples illustrating the behavior of Gradient Descent.

28.2 Gradient Descent for LS Problems

Now, back to the LS problem we are focusing on. Our goal is:

$$\min_x \underbrace{\frac{1}{2}\|Ax - b\|_2^2}_{f(x)}$$

Where f here is convex and differentiable. And the gradient of $f(x)$ is given by:

$$\nabla f(x) = A^\top (Ax - b).$$

The first-order optimality condition requires setting the gradient to zero:

$$\nabla f(x_{OPT}) = 0,$$

Which leads to the normal equations:

$$A^\top A x_{OPT} = A^\top b.$$

For n and d are large, direct methods become computationally expensive. In such cases, iterative methods such as Gradient Descent are preferred.

Algorithm 12 Gradient Descent Algorithm with constant step-size for LS Problems

1: **Input:** $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$ step size μ, iteration number $\mathcal{T}$
2: **Initialize:** $x_0 \in \mathbb{R}^d$
3: **for** $t = 0, 1, \ldots, \mathcal{T}$ **do**
4: $\quad x_{t+1} = x_t - \mu A^\top (Ax_t - b)$
5: **end for**
6: **Output:** $x_{\mathcal{T}}$

We define the error for every iteration as

$$\Delta_t = x_t - x_{OPT}$$

The optimal solution x^* minimizes $f(x)$, hence $\nabla f(x^*) = A^\top(Ax^* - b) = 0$. Then we get the iterative formula for Δ_t:

$$\begin{aligned}\Delta_{t+1} &= \Delta_t - \mu A^\top A \Delta_t \\ &= (I - \mu A^\top A)\Delta_t \\ &= (I - \mu A^\top A)^{t+1}\Delta_0\end{aligned}$$

If we run the gradient descent method for $\mathcal{T}$ iterations, we can get an upper bound of the final output as follows.

$$\begin{aligned}\|\Delta_{\mathcal{T}}\|_2 &= \|(I - \mu A^\top A)^{\mathcal{T}}\Delta_0\| \\ &\leq \|(I - \mu A^\top A)^{\mathcal{T}}\|_2 \|\Delta_0\|_2 \\ &= \sigma_{\max}(I - \mu A^\top A)^{\mathcal{T}} \|\Delta_0\|_2 \\ &= \left(\max_{i=1,\ldots,d} |1 - \mu\lambda_i(A^\top A)|^d\right) \|\Delta_0\|_2\end{aligned}$$

Where λ_i is the i-th eigenvalue in decreasing order.

28.2.1 Optimal Stepsize

To get a small error, we want to get an optimal stepsize. we define λ_+ as the largest eigenvalue of $A^\top A$ and λ_- as the smallest eigenvalue of $A^\top A$. Then we get the fact that

$$\max_{i=1,\ldots,d} |1 - \mu\lambda_i(A^\top A)| = \max\{|1 - \mu\lambda_+|, |1 - \mu\lambda_-|\}$$

Thus, the optimal step size that minimizes the above is

$$\mu^* = \arg\min_{\mu \geq 0} \max\{|1 - \mu\lambda_+|, |1 - \mu\lambda_-|\}$$

This means that μ^* satisfies $|1 - \mu\lambda_+| = |1 - \mu\lambda_-|$, which implies

$$\mu^* = \frac{2}{\lambda_+ + \lambda_-}$$

28.2.2 Running Time

With the optimal step size μ^* determined, we can now analyze the running time of gradient descent based on the resulting convergence rate. The convergence rate is

$$\max\{|1 - \mu^*\lambda_+|, |1 - \mu^*\lambda_-|\} = \frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-} = \frac{\lambda_+/\lambda_- - 1}{\lambda_+/\lambda_- + 1} = \frac{\kappa - 1}{\kappa + 1}$$

Where $\kappa = \frac{\lambda_+}{\lambda_-}$ is the condition number of $A^\top A$. Plug it into the error mormula and we get,

$$\|x_{\mathcal{T}} - x_{OPT}\| \leq (\frac{\kappa - 1}{\kappa + 1})^{\mathcal{T}} \|x_0 - x_{OPT}\|_2$$

Consider two extremes here.

- $\lambda_- = \lambda_+$: it means $\kappa = 1$ and it could convergence in one step.
- $\lambda_+ \gg \lambda_- > 0$: it means $\frac{\kappa-1}{\kappa+1} \approx 1$ which leads to slow convergence.

28.2.3 Computational Complexity

To get ϵ approximation solution, we initialize $x_0 = 0$ for simplicity and consider

$$\|x_{\mathcal{T}} - x_{OPT}\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{\mathcal{T}} \|x_0 - x_{OPT}\|_2 \leq \epsilon$$

We take the logarithm of it to consider the number of iterations $\mathcal{T}$:

$$\mathcal{T} \log \frac{\kappa - 1}{\kappa + 1} + \log \|x_{OPT}\|_2 \leq \log(\epsilon)$$

Then we get the computational complexity of

$$\mathcal{T} = \mathcal{O}\left(\frac{\log(1/\epsilon)}{\log\left(\frac{\kappa+1}{\kappa-1}\right)}\right)$$

And it equals to $\mathcal{O}(\kappa \log(\frac{1}{\epsilon}))$ for large κ, since $\log(\frac{\kappa+1}{\kappa-1}) \approx \frac{2}{\kappa-1}$. Finally, the total computational cost is $\kappa nd \log(\frac{1}{\epsilon})$ for ϵ accuracy.

28.3 Improving condition number dependence: momentum

To accelerate convergence speed, momentum is added to gradient descent. It also called accelerated gradient descent, or heavy-ball method. The iterative formula with momentum is

$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1})$$

The term $\beta_t(x_t - x_{t-1})$ is referred to as momentum. It introduces short-term memory into the update and it is related to a discretization of the second order ordinary differential equation $\ddot{x} + a\dot{x} + b\nabla f(x)$, which models the motion of a body in a potential field given by f. The iterative formula can also be rewritten as

$$\begin{aligned} p_t &= \beta_t p_{t-1} - \nabla f(x_t) \\ x_{t+1} &= x_t + \alpha_t p_t \end{aligned}$$

Where p_t is the search direction and typically we set $p_0 = 0$. Note that when $\beta = 0$, we get the classical gradient descent algorithm.

28.4 Gradient Descent with Momentum for LS Problems

Since there is one time step memory, we consider

$$V_t := \|\Delta_{t+1}\|_2^2 + \|\Delta_t\|_2^2$$

instead of Δ_t. V_t is an energy function that decays to zero and upper-bounds error, i.e., $\|\Delta_t\|_2^2 \leq V_t$. Moreover, we can also write V_t in terms of $V_{t-1} = \|\Delta_t\|_2^2 + \|\Delta_{t-1}\|_2^2$.

Note that $b = Ax_{OPT} + b^{\perp}$ and $\nabla f(x_t) = A^\top A \Delta_t$

$$\begin{bmatrix} \Delta_{t+1} \\ \\ \Delta_t \end{bmatrix} = \begin{bmatrix} x_t - \alpha \nabla f(x_t) + \beta(x_t - x_{t-1}) - x^\star \\ \\ \Delta_t \end{bmatrix} = \begin{bmatrix} (1+\beta)I - \alpha A^\top A & \beta I \\ \\ I & 0 \end{bmatrix} \begin{bmatrix} \Delta_t \\ \\ \Delta_{t-1} \end{bmatrix}$$

Iterating for $t = 1, ..., \mathcal{T}$ and taking the norms, we have

$$\begin{aligned} \sqrt{V_t} = \sqrt{\|\Delta_{t+1}\|_2^2 + \|\Delta_t\|_2^2} &= \left\| \begin{bmatrix} \Delta_{t+1} \\ \\ \Delta_t \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} (1+\beta)I - \alpha A^\top A & \beta I \\ \\ I & 0 \end{bmatrix}^{\mathcal{T}} \begin{bmatrix} \Delta_1 \\ \\ \Delta_0 \end{bmatrix} \right\|_2 \\ &\leq \left\| \begin{bmatrix} (1+\beta)I - \alpha A^\top A & \beta I \\ \\ I & 0 \end{bmatrix}^{\mathcal{T}} \right\|_2 \left\| \begin{bmatrix} \Delta_1 \\ \\ \Delta_0 \end{bmatrix} \right\|_2 \\ &= \sigma_{\max} \left(\begin{bmatrix} (1+\beta)I - \alpha A^\top A & \beta I \\ \\ I & 0 \end{bmatrix}^{\mathcal{T}} \right) \left\| \begin{bmatrix} \Delta_1 \\ \\ \Delta_0 \end{bmatrix} \right\|_2 \\ &= \sigma_{\max} \left(\begin{bmatrix} (1+\beta)I - \alpha A^\top A & \beta I \\ \\ I & 0 \end{bmatrix}^{\mathcal{T}} \right) \sqrt{V_0} \end{aligned}$$

Next, similar to the analysis above, we analyze the step size and convergence result under this method.

28.4.1 Optimal Stepsize

Let $M \in \mathbb{R}^{d \times d}$ with eigrnvalues $\lambda_1, ..., \lambda_d$. Spectral radius of M is defined as

$$\rho(M) := \max_{i=1,...,d} |\lambda_1(M)|$$

Lemma 10. $\lim_{k \to \infty} \sigma_{\max}(M^k)^{\frac{1}{k}} = \rho(M)$

This lemma shows that we can compute the spectral radius of the iterative matrix as the coverage rate.

Lemma 11. Let λ_i denote the eigenvalues of $A^\top A$ for $i = 1, ..., d$. The eigenvalues of

$$\begin{bmatrix} (1+\beta)I - \alpha A^\top A & \beta I \\ I & 0 \end{bmatrix}$$

are given by the eigenvalues of 2×2 matrices

$$\begin{bmatrix} 1+\beta-\alpha\lambda_i & -\beta \\ 1 & 0 \end{bmatrix}$$

for $i = 1, .., d$.

From this lemma, we solve the roots of $u^2 - (1 + \beta - \alpha\lambda_i)u + \beta = 0$ and get the stepsize as

$$\alpha = \frac{4}{\sqrt{\lambda_+} + \sqrt{\lambda_-}}, \qquad \beta = \frac{\sqrt{\lambda_+} - \sqrt{\lambda_-}}{\sqrt{\lambda_+} + \sqrt{\lambda_-}}$$

28.4.2 Convergence result

Setting $\alpha = \frac{4}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$ and $\beta = \frac{\sqrt{\lambda_+}-\sqrt{\lambda_-}}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$ yield

$$\left\| \begin{bmatrix} \Delta_{t+1} \\ \Delta_t \end{bmatrix} \right\|_2 \leq \sigma_{\max} \left(\frac{\sqrt{\lambda_+} - \sqrt{\lambda_-}}{\sqrt{\lambda_+} + \sqrt{\lambda_-}} \right)^\tau \left\| \begin{bmatrix} \Delta_1 \\ \Delta_0 \end{bmatrix} \right\|_2$$

Brief Summery: For ϵ-accuracy.

- Gradient Descent ($\beta = 0$) total computational cost: $\mathcal{O}\left(\kappa nd \log\left(\frac{1}{\epsilon}\right)\right)$.
- Gradient Descent with Momentum total computational cost: $\mathcal{O}\left(\sqrt{\kappa} nd \log\left(\frac{1}{\epsilon}\right)\right)$.
- To find the optimal step sizes, we need to know the eigenvalues of $A^\top A$.
- Conjugate Gradient does not require the eigenvalues explicitly and the total computational cost is: $\mathcal{O}\left(\sqrt{\kappa} nd \log\left(\frac{1}{\epsilon}\right)\right)$

29 Tuesday, April 29th (Scribe: Jingrong Guan)

In the last lesson we discussed the gradient descent algorithm and some of his improvements using the least squares problem as an example. In this lesson, we continue to discuss the effectiveness of some second-order optimization methods through least squares problems.

We first reformulate the least squares problem. Assume $x \in \mathbb{R}^d$, and matrix A is of size $n \times d$. For the function $f(x) = \|Ax - b\|_2^2$. Our goal is to

$$\min_x f(x)$$

Its gradient is:

$$\nabla f(x) = 2A^\top (Ax - b)$$

And the Hessian matrix is:

$$\nabla^2 f(x) = 2A^\top A$$

Note that in this case, the Hessian remains constant during the optimization process. However, for more general non-quadratic functions, the Hessian will change with each iteration. In addition, we can define the condition number of the matrix using its singular values:

$$\kappa = \frac{\sigma_{\max}(A^\top A)}{\sigma_{\min}(A^\top A)}$$

Sometimes, we may also take the square root of this value, depending on specific requirements.

29.1 Newton's Method

The core idea of Newton's method can be understood through the second-order Taylor expansion. Suppose we expand the function f at the point $x^{(t)}$, we have:

$$f(y) \approx f\left(x^{(t)}\right) + \nabla f\left(x^{(t)}\right)^\top \left(y - x^{(t)}\right) + \frac{1}{2}\left(y - x^{(t)}\right)^\top \nabla^2 f\left(x^{(t)}\right)\left(y - x^{(t)}\right)$$

If we directly minimize this quadratic approximation model, we obtain the Newton update formula.

$$x^{(t+1)} = x^{(t)} - \mu^{(t)} \left[\nabla^2 f\left(x^{(t)}\right)\right]^{-1} \nabla f\left(x^{(t)}\right)$$

For least squares problems, since the objective function itself is quadratic, this approximation is actually exact. In Newton's method, the step size $\mu^{(t)}$ is typically set to 1, but it can be adjusted appropriately to ensure convergence.

29.1.1 Convergence

One significant drawback of Newton's method is that it does not always guarantee global convergence. If the initial point is far from the optimal solution, the method may even diverge. This is because Newton's method fundamentally solves for the root of the gradient function rather than

performing pure optimization. Therefore, unless the initial point is already very close to the optimum, directly applying Newton's method may result in incorrect update directions.

However, when the initial point is sufficiently close to the optimal solution, Newton's method exhibits very fast local convergence, even achieving quadratic convergence, where each iteration significantly improves the solution accuracy. In least squares problems, gradient descent can reach the optimal solution in a single iteration if all eigenvalues are identical. Similarly, for such objective functions, Newton's method can also find the optimum in just one step.

29.1.2 Computational Complexity

Calculating and solving linear systems involving the Hessian matrix typically requires $\mathcal{O}(d^3)$ computational complexity, which becomes a major bottleneck for large-scale problems. In contrast, gradient descent only requires one gradient computation per iteration, with a computational cost of $\mathcal{O}(nd)$.

Depending on the problem size and characteristics, the following strategies are commonly adopted:

- If the dataset is small and well-conditioned, directly applying Newton's method can be very efficient;
- If the dataset is large and Hessian computation is expensive, it is more appropriate to use Stochastic Gradient Descent (SGD);
- If the problem lies between these two extremes, consider using Preconditioned SGD or other approximate second-order optimization methods.

In such cases, we often adopt the idea of Hessian approximation to reduce computational costs, for example, by applying sketching techniques to simplify the Hessian computation.

29.2 Randomized Newton Method

The core idea of this method is to approximate or reduce the dimensionality of the Hessian matrix to lower computational complexity while preserving optimization performance as much as possible.

Introduce a sketch matrix S, which is typically a random matrix with much smaller dimensions than the original data. The approximate Hessian is computed as:

$$\widetilde{H} = A^\top S^\top S A$$

This provides a low-dimensional approximation of the original Hessian $A^\top A$. The purpose of introducing S is to reduce the size of the matrices involved and thus simplify the computation. Essentially, the Sketch technique leverages the idea of the *Johnson-Lindenstrauss Lemma*, which suggests that even in high-dimensional spaces, projecting onto a randomly selected lower-dimensional subspace can approximately preserve geometric structures. In the context of optimization, this allows us to preserve the main characteristics of the Hessian through Sketching while avoiding significant computational overhead. Common Sketch methods include:

- Gaussian Random Projection: Provides strong theoretical guarantees but is computationally expensive.
- Sparse Embedding: Computationally efficient and suitable for sparse matrices.
- Fast Johnson-Lindenstrauss Transform (FJLT): Balances computational efficiency with theoretical guarantees.

It's worth mentioning that directly applying Sketch to the Hessian often yields better results than applying it to the gradient. This is because gradient information is only first-order and tends to be noisier, while the Hessian contains second-order curvature information, which has a larger influence on the optimization trajectory.

The updated formula ofthe randomized Newton Method then becomes:

$$x^{(t+1)} = x^{(t)} - \mu^{(t)} \widetilde{H}^{-1} \nabla f\left(x^{(t)}\right) = x^{(t)} - \mu^{(t)} \left(A^\top S^\top S A\right)^{-1} \nabla f\left(x^{(t)}\right)$$

The main computational cost of this method comes from:

- Computing the Sketch Hessian: The complexity is $\mathcal{O}(sd^2)$, where s is the sketch dimension.
- Solving the Linear System: Typically $\mathcal{O}(d^3)$, but since $\widetilde{H}$ is much smaller, the actual computational cost is reduced.

In the iterative formula, we need to be concerned that $A^\top S^\top S A$ is not always reversible. To avoid it being irreversible, we usually introduce a regularization term λI. Therefore, in practice, one would just artificially add the regularization term: i.e., we optimize the function as:

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2}\lambda \|x\|_2^2$$

Its Hessian is:

$$\nabla^2 \widetilde{f}(x) = A^\top A + \lambda I \approx A^\top S^\top S A + \lambda I$$

λ is used to ensure the invertibility of the matrix while improving the numerical stability of the optimization process.

29.2.1 Convergence

We first define$\Delta_t = A(x_t - x^*)$, where $x^* = A^\dagger b$. From the iteration formula, we have

$$\begin{aligned}
\Delta_{t+1} &= A(x_{t+1} - x^*) = A(x_t - x^* - \mu_t (A^\top S^\top S A)^{-1} A^\top (Ax - b)) \\
&= \Delta_t - \mu_t A (A^\top S^\top S A)^{-1} A^\top \Delta_t \\
&= (I - \mu_t A (A^\top S^\top S A)^{-1} A^\top) \Delta_t \\
&= (I - \mu_t A (A^\top S^\top S A)^{-1} A^\top)^t \Delta_0
\end{aligned}$$

Let $A = U \Sigma V^\top$, Note that

$$\begin{aligned}
A(A^\top S^\top S A)^{-1} A^\top &= U \Sigma V^\top (V \Sigma U^\top S^\top S U \Sigma V^\top)^{-1} V \Sigma U^\top \\
&= U (U^\top S^\top S U)^{-1} U^\top
\end{aligned}$$

After M steps, $\Delta_M = (I - \mu U(U^\top S^\top SU)^{-1}U^\top)^M \Delta_0$

Since $\Delta_t \in \text{Range}A$, $UU^\top \Delta_t = \Delta_t$. Thus,

$$\begin{aligned}
\|\Delta_M\|_2 = \|U^T\Delta_M|_2 &= \|U^T(I - \mu U(U^\top S^\top SU)^{-1}U^\top)\Delta_{M-1}\|_2 \\
&= \|U^\top - \mu U^\top U(U^\top S^\top SU)^{-1}U^\top \Delta_0\|_2 \\
&= \|(I - \mu(U^\top S^\top SU)^{-1})U^\top \Delta_{M-1}\|_2 \\
&= \|(I - \mu(U^\top S^\top SU)^{-1})^M U^\top \Delta_0\|_2 \\
&\leq \|(I - \mu(U^\top S^\top SU)^{-1})^M\|_2 \|U^\top \Delta_0\|_2 \\
&\leq \sigma_{\max}\left(I - \mu(U^\top S^\top SU)^{-1}\right)^M \|\Delta_0\| \\
&= \max_i |1 - \mu\lambda_i(U^\top S^\top SU)^{-1}|^M \|\Delta_0\|
\end{aligned}$$

Since we know $U^\top S^\top SU \approx U^\top U = I_d$, $\|U^\top S^\top SU - I\|_2 \leq \epsilon$. Thus,

$$\sigma_{\max}(I - U^\top S^\top SU) \leq \epsilon$$

Then,

- eigenvalues of $U^\top S^\top SU$ lies in $(1-\epsilon, 1+\epsilon)$
- eigenvalues of $(U^\top S^\top SU)^{-1}$ lies in $(\frac{1}{1+\epsilon}, \frac{1}{1-\epsilon})$

Plug it back and we have $\|\Delta_M\|_2 \leq \max\{|1 - \mu\frac{1}{1-\epsilon}|, |1 - \mu\frac{1}{1+\epsilon}|\}^M \|\Delta_0\|_2$. Then we get the optimal step size by letting the two terms equal.

$$\mu^* = 1 - \epsilon^2$$

And then we get the convergence of $\|\Delta_M\|_2 \leq \epsilon^M \|\Delta_0\|_2$

29.2.2 Computational Complexity

The computational complexity of the Randomized Newton Method consists mainly of the following:

- SA: $\mathcal{O}(nd \log n)$
- $A^\top S^\top SA$: $\mathcal{O}(sd^2)$
- $(A^\top S^\top SA)^{-1}$: $\mathcal{O}(d^3)$
- M iterations: $\mathcal{O}(ndM)$

Say $\epsilon = 10^{-M}$, we have $M = \log \frac{1}{\epsilon}$. Then the total cost is

$$\mathcal{O}(nd \log n + d^3 + nd \log \frac{1}{\epsilon})$$

29.3 The Relationship Between Regularization and Dual Problems

Finally, we further explore the dual formulation of the least squares problem:

Primal Problem:

$$\min_x \|Ax - b\|^2 + \lambda \|x\|^2$$

Dual Problem:

$$\min_y \|A^\top y - c\|^2 + \lambda' \|y - b\|^2$$

The solutions to the primal and dual problems are equivalent under certain conditions. This equivalence is highly important in algorithm design because directly solving the dual problem is often more computationally efficient. This is also the mathematical foundation behind Kernel Methods, which avoid high-dimensional computations by operating directly on inner products.

30 Thursday, May 1st (Scribe: Jingrong Guan)

Last time, we discussed second-order optimization methods—an important class of algorithms widely used in optimization. In particular, we focused on their application to least squares problems. A key advantage of second-order methods is their fast convergence per iteration compared to first-order methods such as gradient descent or stochastic gradient descent (SGD). However, this comes at a cost: each iteration is significantly more computationally expensive due to the need to construct or solve a linear system involving the Hessian matrix.

This motivates the use of randomized sampling techniques to reduce computational cost. One powerful class of such methods is the sketch-and-project framework, which randomly samples multiple constraints per iteration. Unlike SGD, these methods do not follow gradient directions, and unlike second-order methods, they do not explicitly search for function roots. Instead, each iteration solves a restricted least squares problem based on the sampled constraints, effectively projecting the current iterate onto a lower-dimensional solution space.

This lecture—the final one of the semester—focuses on **Krylov subspace methods**, which serve as a classical yet foundational framework for designing efficient iterative solvers. Many modern randomized algorithms implicitly operate within Krylov-like subspaces or explicitly leverage Krylov ideas (e.g., via preconditioning). Understanding Krylov methods provides the theoretical foundation for analyzing convergence rates, designing efficient preconditioners, and ultimately going beyond with randomized techniques.

30.1 Krylov Subspace Methods

To better understand where Krylov methods fit in the broader landscape of solvers, we revisit a fundamental question in numerical linear algebra: **How long does it take to solve** $Ax = b$? As we've seen, there is no single answer to this question. The runtime depends on a variety of factors:

- **Sparsity:** sparsity of the input matrix A?
- **Spectral:** how quickly do the singular values of A decay?
- **Structure:** whether A has some domain-specific structure, such as Laplacian, Toeplitz, Hankel, circulant, etc.? For example, certain transforms like the randomized Hadamard transform can be implemented efficiently via recursive FFTs.
- **Precision:** whether we need exact solutions, or solutions with medium or low precision?
- **Robustness:** whether we can tolerate a small chance of failure?

Two broad categories of methods are typically employed to address such problems:

Direct methods: such as Gaussian elimination, QR decomposition, LU factorization, and SVD, aim to compute the exact solution in a finite number of steps. While these methods are robust and reliable, they often incur high computational costs—typically $\mathcal{O}(n^3)$ time for an $n \times n$ system.

Iterative methods: These include algorithms like Conjugate Gradient, MINRES, GMRES, and LSQR, which are more suitable for large-scale problems. They iteratively refine the solution, and each iteration costs $\mathcal{O}(n^2)$, with the total number of iterations depending on the conditioning of the matrix and the desired precision.

Among iterative approaches, a particularly important class is formed by **Krylov subspace methods**. These methods have been a dominant force since the 1970s. At each iteration, they repeatedly apply the matrix A to the current residual, thereby constructing increasingly accurate approximations within the so-called Krylov subspace. For example, CG, GMRES, MINRE, and LSQR are all Krylov subspace methods. These methods converge faster in well-conditioned problems and have the ability to exploit spectral structure without forming the full matrix explicitly.

This naturally raises two important questions:

Q1: How do Krylov methods exploit the spectral (eigenvalue) structure of the problem to achieve strong performance?
Q2: Can randomization be used to go *beyond* the capabilities of Krylov methods?

To explore these questions, we begin by examining the foundations of Krylov subspace methods.

30.1.1 Krylov subspace

Definition 8. Given matrix $A \in \mathbb{R}^{n\times n}$ and vector $b \in \mathbb{R}^n$, the k-th Krylov subspace is defined as:

$$\mathcal{K}_k(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}.$$

Intuitively, this subspace is constructed by repeatedly applying the matrix A to the initial vector b. If b is an eigenvector of A, then Ab, A^2b, etc., will all lie in the same direction—simply scaled versions of b—and the Krylov subspace will be one-dimensional. However, for a general vector b, each successive multiplication A^jb may point in a new direction.

As a result, the set $\{b, Ab, A^2b, \ldots, A^{k-1}b\}$ spans a richer subspace, and this space grows as k increases. Typically, it will be of dimension k, though in degenerate cases (e.g., if A is rank-deficient or b lies in a low-dimensional invariant subspace), the dimension may be smaller.

Each vector v in $\mathcal{K}_k(A, b)$ can be written as:

$$v = p(A)b,$$

where $p(x) = c_0 + c_1x + \cdots + c_{k-1}x^{k-1}$ is a polynomial of degree $k-1$. That is, Krylov subspaces are precisely the set of all vectors of the form $p(A)b$ for polynomials p of degree less than k.

This perspective enables us to approximate the solution $x^* = A^{-1}b$ by carefully choosing a p, relying only on matrix-vector products without directly computing the inverse of A.

30.1.2 Recipe for Linear Solver

Start by choosing a subspace dimension $k = 1, 2, \ldots$, and construct the Krylov subspace $\mathcal{K}_k(A, b)$. At each step, we maintain an approximate solution $\hat{x}$ in the subspace, which can be written as

$$\hat{x} = p(A)b$$

where $p(x)$ is a polynomial chosen to minimize the error to $A^{-1}b$ in some norms.

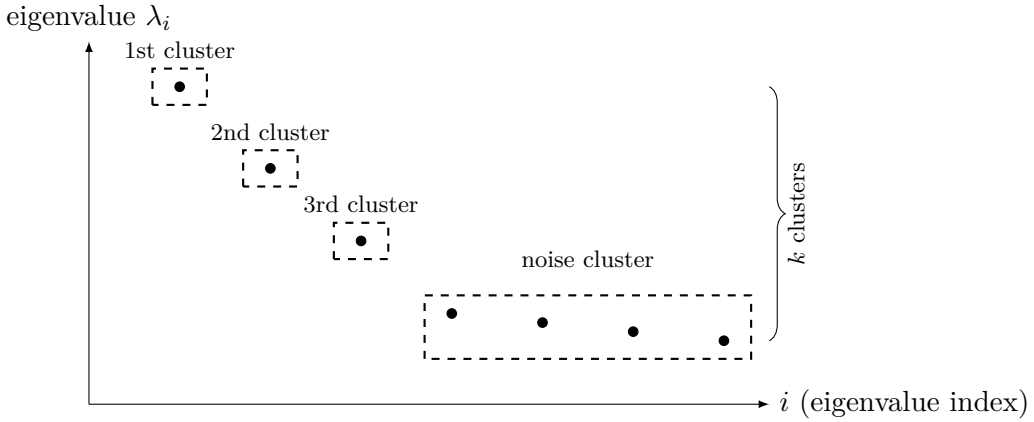


Figure 3: Eigenvalue clustering for a real symmetric matrix A

The existence of such p is guaranteed by interpolation theory, which ensures that a degree $k-1$ polynomial can be constructed to approximate the k clusters. So if the eigenvalues of A form k distinct and well-separated clusters, then there exists a polynomial p of degree $k-1$ such that

$$p(\lambda_i) \approx \lambda_i^{-1}, \quad \forall i$$

and hence

$$p(A)b \approx A^{-1}b,$$

implying that $A^{-1}b$ lies approximately in $\mathcal{K}_k(A,b)$. Then the Krylov subspace $\mathcal{K}_k(A,b)$ contains a high-quality approximation to $x^* = A^{-1}b$.

The primary computational cost is in computing matrix-vector products $A^k b = A(A^{k-1}b)$, which only requires repeated applications of A to a vector, avoiding explicit computation of A^k. Each matrix-vector product costs $\mathcal{O}(n^2)$ operations.

Now, we shift from an eigenvalue-based to a singular value-based perspective to handle general (possibly non-symmetric) matrices.

Theorem 6. If $A \in \mathbb{R}^{m\times n}$ has singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ such that $\frac{\sigma_k}{\sigma_n} \leq \kappa_k$, then there exists a Krylov subspace method (e.g., LSQR) that converges to an ε-approximate solution of $Ax = b$ in

$$T = \mathcal{O}\left(k + \kappa_k \log\left(\frac{1}{\varepsilon}\right)\right)$$

iterations.

To solve an $n \times n$ linear system with at most k large, isolated singular values and a cluster of width κ_k, we require

$$(\text{cost of } v \mapsto Av) \quad \times \quad \mathcal{O}(k + \kappa_r \log(1/\varepsilon)) \quad \approx \quad \mathcal{O}(n^2 k + n^2 \kappa_k \log(1/\varepsilon)).$$

There is a trade-off there, as we decrease k, the cluster width κ_k grows larger. So, can we do better than Krylov subspace methods? The answer depends on the model of access to the matrix A:

- If we only have access to matrix-vector products, then Krylov subspace methods are optimal.
- If we have direct access to the entries/structure of A, then RandNLA methods can do better.

An example of this, we can solve an $n \times n$ linear system with at most k large isolated singular values plus a cluster of width κ_k of ε-precision in (ignores logarithmic terms in n and κ_r)

$$\tilde{\mathcal{O}}\left(nk^2 + n\kappa_k \log(1/\varepsilon)\right)$$

In traditional Krylov methods, the runtime is $\mathcal{O}\left(n^2k + n^2\kappa_k \log(1/\varepsilon)\right)$. For $k \leq \sqrt{n}$, the nk^2 term is negligible compared to the rest.

30.2 Improving Krylov Convergence via Sketching

In recent years, randomized algorithms have emerged as powerful tools for solving large-scale linear systems. These methods offer new paradigms for iterative solvers. Key classes include:

- **Sketch-and-solve**: the core method, with deep connections to random matrix theory (RMT).
- **Sketch-and-precondition**: a technique that recovers standard Krylov-style convergence guarantees by building preconditioners from sketches.
- **Sketch-and-project**: a general and increasingly popular paradigm that can outperform Krylov methods on large problems. In particular, **sketch-and-project++** is a recent enhancement that achieves state-of-the-art performance.

The idea of Sketch-and-Solve was originally designed for overdetermined linear systems. However, when the linear system is square, more refined techniques are needed.

We can improve Krylov Convergence via two main randomized methods:

- **Sketch-and-precondition:** Recovers classical Krylov guarantees even in challenging regimes.
- **Sketch-and-project:** In certain regimes, this method can even outperform Krylov methods.

30.2.1 Sketch-and-precondition

Suppose we want to solve a linear system $Ax = b$, but A has a large condition number $\kappa(A)$. Instead of solving this directly, we consider a preconditioned system

$$AM^{-1}z = b, \quad \text{where } x = M^{-1}z,$$

such that the preconditioned matrix AM^{-1} has a smaller condition number $\kappa(AM^{-1})$. The key idea is to construct a matrix M that approximates A but is easier to invert or apply.

There are several practical strategies for constructing a good preconditioners M:

- Exploit special structure (e.g., Laplacian matrices).
- Use partial matrix decompositions (e.g., incomplete LU).
- Apply randomization, in particular via sketch-and-precondition.

For example, the "ideal" preconditioner of a general matrix $A \in \mathbb{R}^{n\times n}$ comes from its SVD

$$A = U\Sigma V^\top.$$

If we retain only the top-k singular values, we obtain the rank-k approximation

$$A_k = U_k \Sigma_k V_k^\top$$

which preserves the dominant spectral structure and is often sufficient for fast convergence.

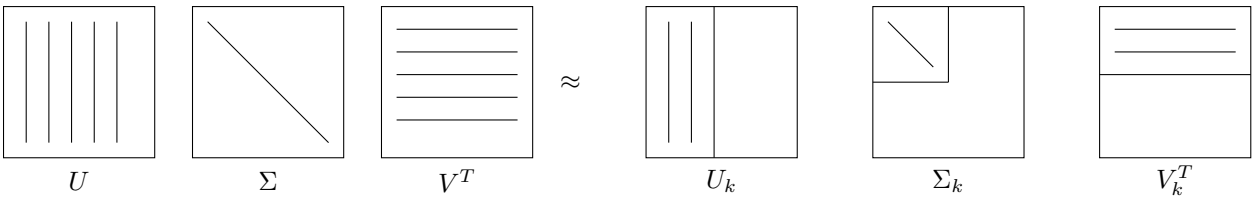


Figure 4: Rank-k Approximation from SVD

To avoid expensive SVD operations, we can use Randomized SVD to approximate the top-k SVD and construct fast preconditioner by combining spectral approximation with modern sketching tools. In practice, such randomized preconditioners can significantly reduce the number of iterations required by Krylov-type solvers.

In summary, the strategy is to construct fast preconditioners by extracting top-k right singular vectors using sketching:

1) Compute $\tilde{A} = SA$ and the right singular vectors $V \in \mathbb{R}^{n\times k}$ of $\tilde{A}$, where $S \in \mathbb{R}^{k\times n}$, $\tilde{A} \in \mathbb{R}^{k\times n}$.
2) Let $P = VV^\top$ be the projection onto the row space of $\tilde{A}$.
3) Compute the SVD of the projected matrix AP. Error is measured as $\|A - APP\|_F$.
4) **Refinement**: Use $SA(A^\top A)^q$ instead of SA for improved quality.

Before preconditioning, the condition number is $\kappa(A) \approx \frac{\sigma_1}{\sigma_n}$. After applying this preconditioner, the effective condition number improves to approximately $\kappa(AM^{-1}) \approx \kappa_k = \frac{\sigma_k}{\sigma_n}$

Using randomized top-k SVD preconditioning, the total runtime for solving an $n \times n$ linear system is given by:

$$\underbrace{\mathcal{O}(n^2 k)}_{\text{preconditioning}} + \underbrace{\mathcal{O}(n^2 \kappa_k \log(1/\epsilon))}_{\text{solving}}.$$

The result shows this approach matches the theoretical complexity guarantees of Krylov subspace methods. In other words, we are performing essentially the same operation on the lower end of the spectrum, but using a different approach. For the randomized top-k preconditioning method, it does not proceed step by step like Krylov; instead, it performs the randomness upfront before starting the iteration. That is, the "setup" phase of the iteration is done all at once in advance.

While the theoretical complexity is the same, randomized preconditioning can be more advantageous in practice, particularly when:

- The cost of preconditioning can be reused across multiple solves;
- The cost of preconditioning can be significantly reduced through optimization.

This observation motivates a further question: if it only matches, what benefits do randomized techniques bring? The answer is: we can further improve it using the Sketch-and-Project method!

30.2.2 Sketch-and-project

The Sketch-and-Project paradigm avoid solving the full system or applying a preconditioned system. Instead, it iteratively approximates the optimal solution by projecting onto a sequence of sketched systems. It eliminates the need for explicit preconditioning while still achieving strong convergence guarantees. As an simple example, we first introduce the Randomized Kaczmarz algorithm:

Algorithm 13 Randomized Kaczmarz (RK) Algorithm

1: **Input:** $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, initial value $x_0 \in \mathbb{R}^n$
2: **for** $t = 0, 1, 2, \ldots, T$ **do**
3: Sample index I_t with probability $\sim \|a_{i_t}\|^2$
4: Project current iterate onto solution of I_t-th equation

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

5: **end for**
6: **Output:** x_T, an approximation to the solution of $Ax = b$

Theorem 7. RK converges to optimal x^* in expectation:

$$\mathbb{E}\left[\|x_{t+1} - x^*\|_2^2\right] \le \left(1 - \frac{\sigma_{\min}(A)}{\|A\|_F^2}\right) \|x_t - x^*\|_2^2,$$

RK is a minimalist version of the Sketch-and-Project method: at each step, it samples a single row and projects onto the solution space of that individual constraint. That is, solve:

$$x_{t+1} = \arg\min_x \|x_t - x\|_2^2 \quad \text{subject to} \quad a_{I_t}^\top x = b_{I_t}.$$

In contrast, Sketch-and-Project projects the current iterate onto the solution space of $SA = Sb$, where S is a sampled sketching matrix. That is, solve:

$$x_{t+1} = \arg\min_x \|x_t - x\|_2^2 \quad \text{subject to} \quad SAx = Sb.$$

Conceptually, this is similar to the projection step in truncated SVD or low-rank approximations, but applied iteratively. When S is a good sketch, convergence of this method is theoretically guaranteed, with analysis similar to that of low-rank Krylov methods. The projection matrix $P = VV^\top$, constructed from the top-k right singular vectors of SA, defines the subspace onto which x_t is projected at each step.

Theorem 8. [Convergence of Sketch-and-Project]Let P be the orthogonal projection onto the row space of SA. Then the following bound holds:

$$\mathbb{E}[\|x_{t+1} - x^*\|_2^2] \leq (1 - \lambda_{\min}(\mathbb{E}[P])) \cdot \|x_t - x^*\|_2^2$$

Proof. We define the iteration as:

$$x_{t+1} = \arg\min_x \|x - x_t\|_2^2 \quad \text{subject to} \quad SAx = Sb$$

This has a closed-form solution:

$$x_{t+1} = x_t - (SA)^\dagger S(Ax_t - b)$$

Let x^* be a solution to $Ax = b$, then

$$x_{t+1} - x^* = x_t - x^* - (SA)^\dagger S(Ax_t - Ax^*) = (I - P)(x_t - x^*)$$

where P is the orthogonal projection onto the row space of SA.

$$\begin{aligned}
\mathbb{E}[\|x_{t+1} - x^*\|_2^2] &= \mathbb{E}[(x_t - x^*)^\top (I - P)^2 (x_t - x^*)] \\
&= (x_t - x^*)^\top \mathbb{E}[(I - P)^2](x_t - x^*) \\
&= (x_t - x^*)^\top (I - \mathbb{E}[P])(x_t - x^*) \\
&\leq (1 - \lambda_{\min}(\mathbb{E}[P])) \cdot \|x_t - x^*\|_2^2
\end{aligned}$$

□

To understand the convergence behavior of the Sketch-and-Project method, it is essential to analyze the spectrum of the expected projection operator $\mathbb{E}[P]$. This operator governs the contraction in each iteration and ultimately determines how quickly the iterates converge to the solution. In what follows, we study the spectral properties of $\mathbb{E}[P]$.

1st. Vanilla RK: Recall RK samples $I \in \{1, \ldots, n\}$ with probability $\mathbb{P}[I = i] \sim \|a_i\|^2$. Then the projector is:

$$P = \frac{a_i a_i^\top}{\|a_i\|_2^2}, \quad \mathbb{P}[I = i] = \frac{\|a_i\|_2^2}{\|A\|_F^2}$$

We can compute the expected projection $\mathbb{E}[P]$ as:

$$\mathbb{E}[P] = \sum_{i=1}^n \mathbb{P}[i = i] \cdot \frac{a_i a_i^\top}{\|a_i\|_2^2} = \sum_{i=1}^n \frac{\|a_i\|_2^2}{\|A\|_F^2} \cdot \frac{a_i a_i^\top}{\|a_i\|_2^2} = \frac{1}{\|A\|_F^2} \sum_{i=1}^n a_i a_i^\top = \frac{1}{\|A\|_F^2} A^\top A$$

Since

$$\lambda_{\min}(\mathbb{E}[P]) = \frac{\lambda_{\min}(A^\top A)}{\|A\|_F^2} = \frac{\sigma_{\min}(A)^2}{\|A\|_F^2}$$

we get the convergence guarantee for RK:

$$\mathbb{E}\left[\|x_{t+1} - x^*\|_2^2\right] \leq \left(1 - \frac{\sigma_{\min}(A)^2}{\|A\|_F^2}\right) \cdot \|x_t - x^*\|_2^2$$

2nd. Generalization Sketch-and-Project: If we choose different distributions for the sketching matrix S, the resulting projection matrix P will also differ, and this in turn affects the convergence behavior of the algorithm. To analyze the spectral properties of $\mathbb{E}[P]$, one can leverage Random Matrix Theory (RMT), which provides tools to characterize the spectrum (e.g., CMD, singular value distributions) of random matrix ensembles.

To obtain desirable spectral properties of the expectation $\mathbb{E}[P]$, we must use sketches that are **nearly unbiased**. Examples of such sketches include sub-Gaussian sketches, etc..

This leads to the question: What happens if the sketching matrices induce orthogonal projection matrices? The spectrum of a deterministic projection matrix P is always contained in $\{0, 1\}$. However, the spectrum of its expectation $\mathbb{E}[P]$ can covering virtually any subset of the real interval $[0, 1]$. A key insight is the following:

Given a matrix A and an isotropic random sketching matrix S, consider the orthogonal projection matrix P onto the row span of SA. Then, under mild conditions on S, there exists a regularization parameter $\lambda > 0$ such that

$$\mathbb{E}[P] \approx A^\top A(A^\top A + \lambda I)^{-1}.$$

This result implies that, although no explicit regularization term is added, the structure of the random sketch inherently induces a regularization effect similar to ridge regression.

Theorem 9. Let P be the orthogonal projection onto the row span of SA. If S is a $k \times n$ (sub-)Gaussian matrix, then Sketch-and-Project gives:

$$\mathbb{E}\left[\|x_{t+1} - x^*\|^2\right] \leq \left(1 - \frac{k\sigma_{\min}(A)/2}{\mathbb{E}[\|A - PA\|_F^2]}\right) \|x_t - x^*\|^2.$$

This follows from new RMT results for orthogonal projectors. And this convergence rate shows that Sketch-and-Project behaves as if a top-k SVD preconditioner had been applied, even though it wasn't actually built.

Recall Randomized SVD, if $S \in \mathbb{R}^{(2k+1)\times n}$ is a Gaussian sketching matrix, then the projection matrix P satisfies:

$$\mathbb{E}[\|A - AP\|_F^2] \leq 2\|A - A_k\|_F^2 \leq 2\sum_{i>k} \sigma_i^2 \leq 2n\sigma_k^2.$$

Plug this into the convergence guarantee of Sketch-and-Project,

$$\mathbb{E}\left[\frac{\|x_{t+1} - x^*\|_2^2}{\|x_t - x^*\|_2^2}\right] \leq 1 - \frac{k \cdot \sigma_{\min}(A)/2}{\mathbb{E}[\|A - AP\|_F^2]} \leq 1 - \frac{k}{4n\kappa_k}$$

So, ignoring the cost of sketching and projecting, the "ideal" complexity of sketch-and-project should be:

$$\underbrace{\mathcal{O}(nk)}_{\text{sketch SA}} \quad \times \quad \underbrace{\mathcal{O}\left(\frac{n}{k} \cdot \kappa_k^2 \cdot \log k\right)}_{\text{iterates}} \quad = \quad \mathcal{O}(n^2\kappa_k^2 \log k)$$

30.3 Extensions and Outlook

The convergence rate and implicit regularization effects of Sketch-and-Project naturally suggest several avenues for further improvement and integration with other numerical methods:

- **Combining with Classical Techniques.** Sketch-and-Project can be combined with traditional strategies such as momentum acceleration. These combinations can yield better empirical performance.
- **Toward a New Family of Solvers.** Sketch-and-Project, together with related iterative algorithms like Kaczmarz, coordinate descent (CD), forms a new family of randomized linear solvers. These methods offer distinct trade-offs in terms of iteration complexity and structure exploitation.
- **Improving Krylov Methods.** Use sketching to approximate $v \mapsto Av$ and reduce per-iteration cost in Krylov methods for large-scale problems.

As pointed out by Y. Saad, *"The current buoyant activity in RandNLA is somewhat reminiscent of the golden era of iterative methods three decades ago!"* Although many aspects of the current field of Randomized Numerical Linear Algebra (RandNLA) are still maturing, there has been a surge of exciting progress since the development of several core techniques around a decade ago. This renewed momentum reflects a growing trend of introducing randomness in more refined ways—enhancing not only the convergence speed of traditional algorithms but also their stability. Perhaps in a few more years, the field will become more mature, but even now, we can already see the new perspectives and methods that RandNLA offers beyond classical matrix algorithms.