Tutorial on Matrix Sketching

NYU Tandon School of Engineering, Christopher Musco

WHAT IS SKETCHING?

Sketching is a very general technique in randomized algorithms. Two steps:

1. Use a very fast algorithm to compress an object or data set down to a smaller size that still maintains interesting information about the original object.
2. Compute using the smaller object.

Can be viewed as a kind of semantic compression.

Example: Johnson-Lindenstrauss dimensionality reduction

Suppose you have $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and you want to compute

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \qquad \text{for all } i, j.$$

Sketch:

- Pick $m = O\left(\frac{\log(n/\delta)}{\epsilon^2}\right)$. Choose random Gaussian $\mathbf{G} \in \mathbb{R}^{m \times d}$.
- Set $\tilde{\mathbf{x}}_i = \frac{1}{\sqrt{m}}\mathbf{G}\mathbf{x}_i$.

Claim: With probability $(1 - \delta)$, for all pairs $\mathbf{x}_i, \mathbf{x}_j$,

$$(1 - \epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2 \leq (1 + \epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2$$

EXAMPLE: JOHNSON-LINDENSTRAUSS DIMENSIONALITY REDUCTION

Prove via concentration of chi squared random variables.

Step 1: Show that, for any $\mathbf{y} \in \mathbb{R}^n$, with probability $\frac{\delta}{n^2}$,

$$(1-\epsilon)\|\mathbf{y}\|_2 \leq \|\tilde{\mathbf{y}}\|_2 \leq (1+\epsilon)\|\mathbf{y}\|_2,$$

where $\tilde{\mathbf{y}} = \frac{1}{\sqrt{m}}\mathbf{G}\mathbf{y}$ is a sketched vector.

Step 2: Union bound to say the bound holds simultaneously for all $\binom{n}{2}$ vectors of the form $\mathbf{y} = \mathbf{x}_i - \mathbf{x}_j$.

Example: Johnson-Lindenstrauss dimensionality reduction

Time complexity: Improved from $O(d)$ to $O(\log n)$ per vector pair.

Space complexity: Improved from $O(nd)$ to $O(n \log n)$

Communication complexity: Improved from $O(d)$ to $O(\log n)$ per vector.

Can be used as pre-processing step, or in conjunction with other algorithms (e.g. data structures for near neighbor search). **Very flexible technique.**

SKETCHING TODAY

Sketching is an important part of the modern algorithmic toolkit. Used in:

- Streaming algorithms for vector data, graphs, etc.
- Computational geometry
- Distributed algorithms (federated learning methods)
- Search algorithms
- Database algorithms
- Linear algebra

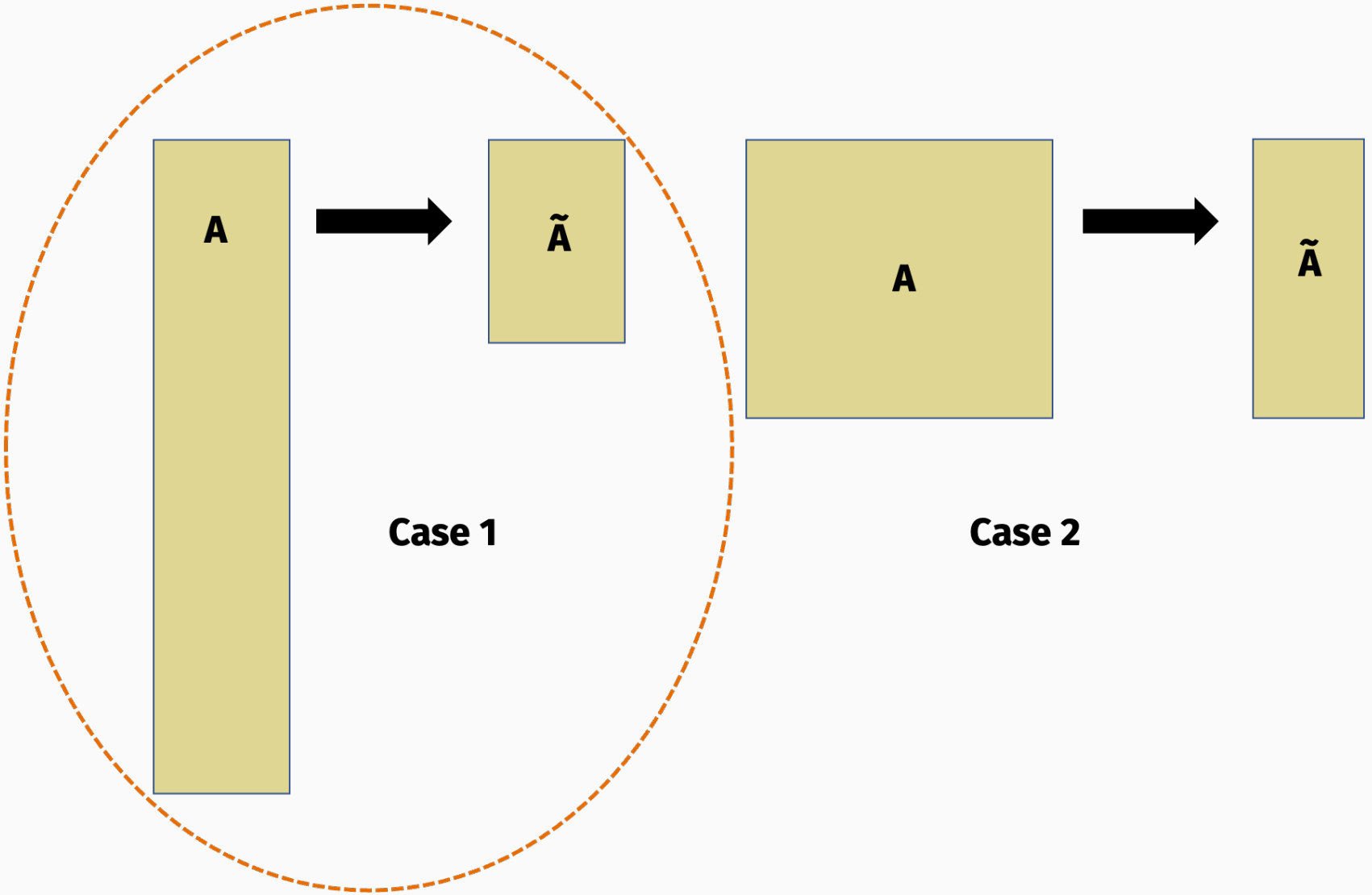
Other names: Dimensionality reduction, sparsification, coresets.

We are obviously all interested in the setting where the object being compressed is a matrix.

Today:

1. What information can we expect a matrix sketch to preserve?
    - Give two general purpose definitions that can be used in many applications.
2. How do you compute sketches satisfying these definitions?
3. What are different ways to use sketches?

TWO CASES



SUBSPACE EMBEDDINGS

[Sarlós, 2006] [Woolfe, Liberty, Rokhlin, Tygert 2008]

Definition (Subspace Embedding)

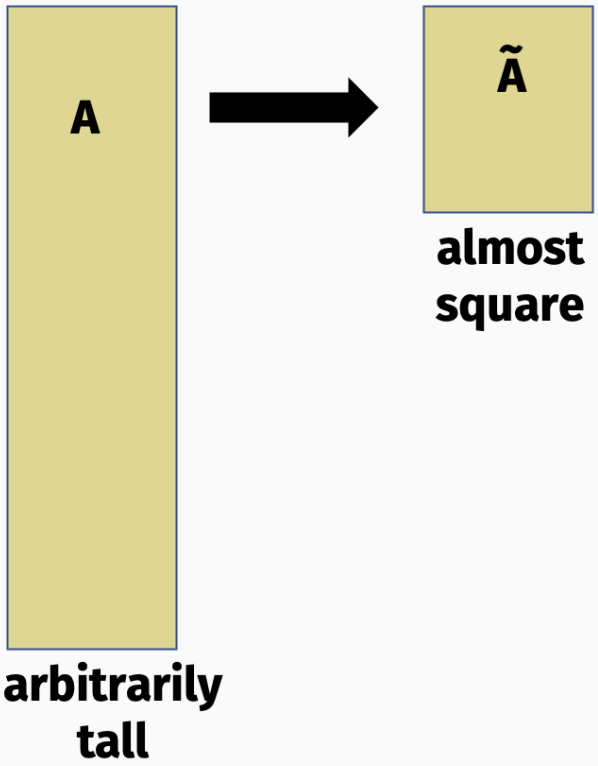
A matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{m \times d}$ is a subspace embedding for $\mathbf{A} \in \mathbb{R}^{n \times d}$ if, for all $\mathbf{x} \in \mathbb{R}^d$,

$$(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2 \leq \|\tilde{\mathbf{A}}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2$$

For a single $\mathbf{x}$, this is equivalent to the Johnson-Lindenstrauss guarantee for $\mathbf{y} = \mathbf{A}\mathbf{x}$. The tricky part is we need it to hold for all $\mathbf{x}$. I.e. for all $\mathbf{y}$ in a d dimensional subspace.

Claim (Sarlós, 2006)

Set $m = O\left(\frac{d+\log(1/\delta)}{\epsilon^2}\right)$. *If* $\tilde{\mathbf{A}} = \frac{1}{\sqrt{m}}\mathbf{GA}$ *where* $\mathbf{G}$ *is an* $m \times n$ *random Gaussian matrix then with probability* $(1-\delta)$, $\tilde{\mathbf{A}}$ *is a subspace embedding for* $\mathbf{A}$.

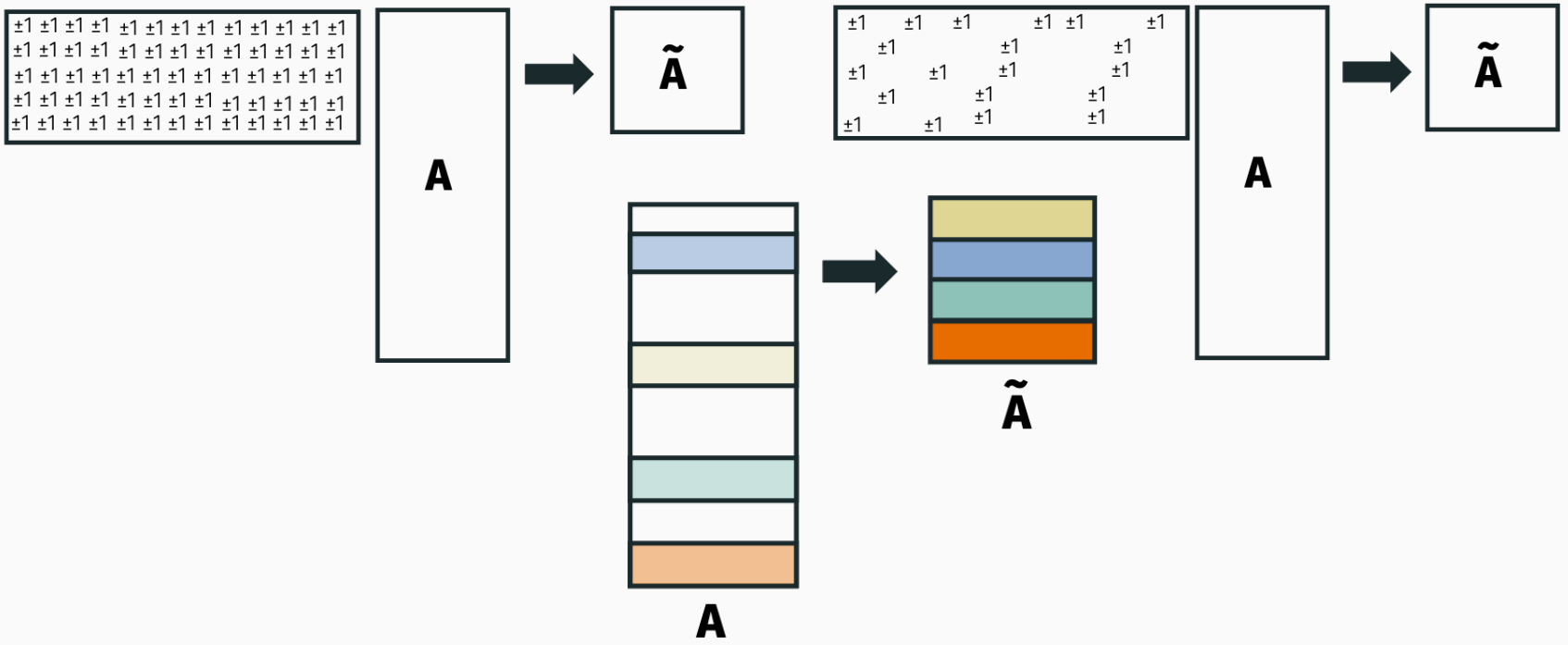


Proof is similar to standard JL lemma:

1. Prove that $\|\tilde{\mathbf{A}}\mathbf{x}\|_2^2 \approx \|\mathbf{A}\mathbf{x}\|_2^2$ for single $\mathbf{x}$.
2. Union bound over a net in d dimensions (which has 2^d vectors in it).

OTHER CONSTRUCTIONS FOR SUBSPACE EMBEDDINGS

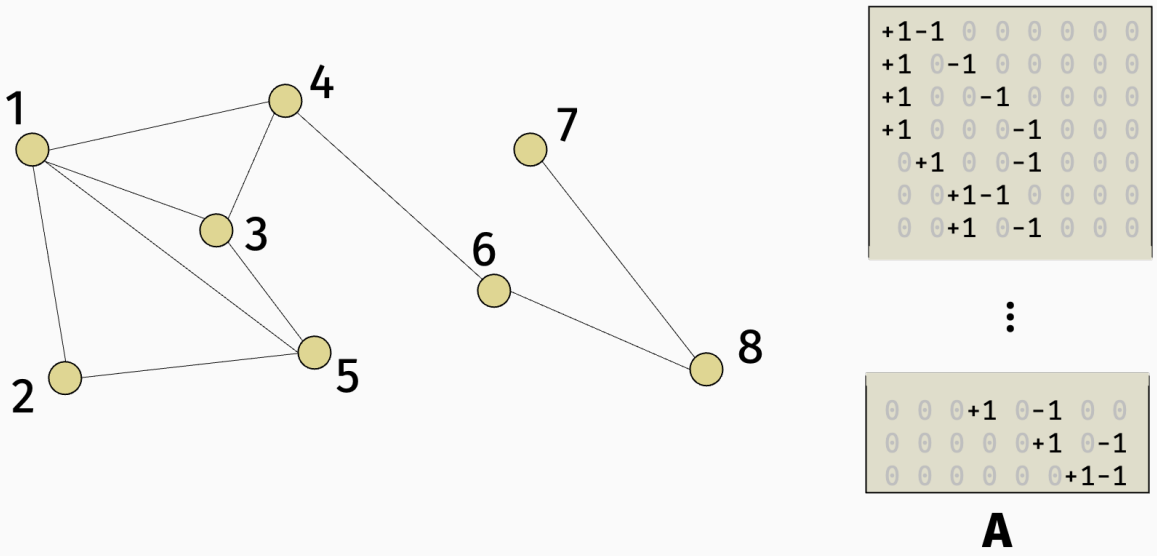
Can use sparse random matrices [Clarkson, Woodruff 2013], structured matrices [Ailon Chazelle 2006], sampling [Drineas, Mahoney, Muthukrishnan 2006, Spielman Srivastava 2008], etc.



Important: Can compute subspace embedding in $\tilde{O}(nd)$ time. Dense JL takes $O(nd^2)$ time, which usually isn't useful.

SPECTRAL SPARSIFICATION

If we are sketching the edge vertex incidence matrix of a graph, $\mathbf{A}$, and construct a subspace embedding $\tilde{\mathbf{A}}$ by selecting and reweighting rows, $\tilde{\mathbf{A}}$ is the incidence matrix of a spectral sparsifier for the graph.

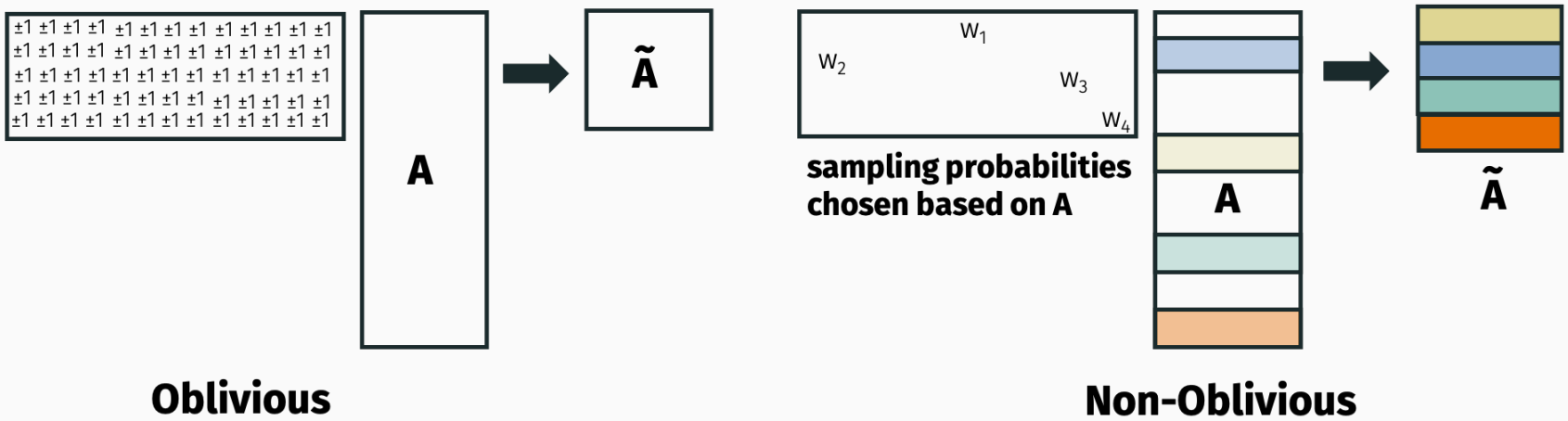


If the graph has d nodes, $\tilde{\mathbf{A}}$ won't have much more than $O(d)$ edges.

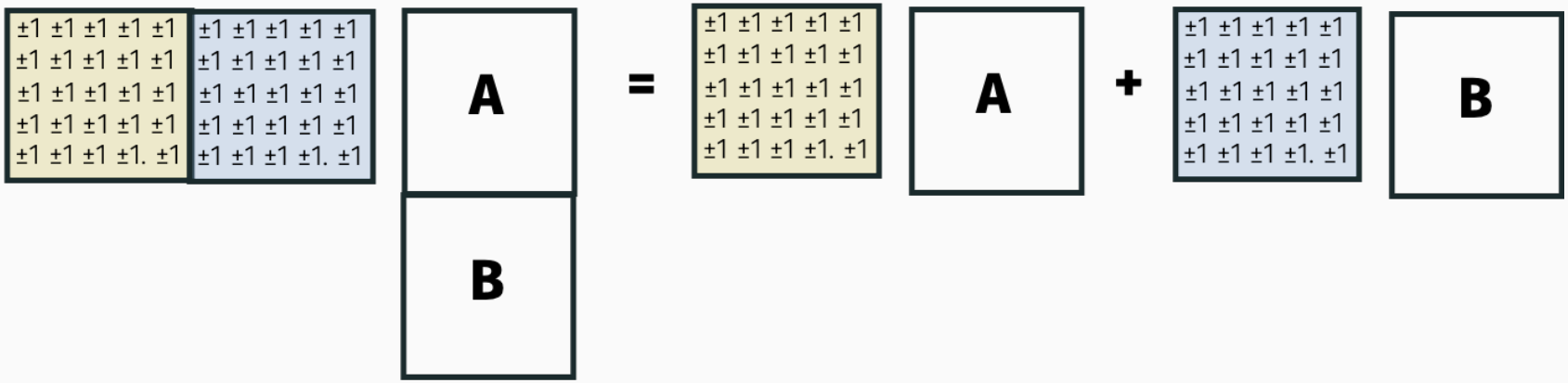
OBLIVIOUS VS. NON-OBLIVIOUS

Most sketches are linear, meaning that the operation can be written as **GA** for some matrix **G**.

- **Oblivious Sketch:** **G** does not depend on **A**.
- **Non-oblivious Sketching:** **G** could depend on **A**.



Nice property of oblivious sketching: Easily applied to data in a stream, to data separated on different machines, or to data that will be later updated.



What can you do with this sketch?

Reminder of the guarantee we have:

Definition (Subspace Embedding)

A matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{m \times d}$ is a subspace embedding for $\mathbf{A} \in \mathbb{R}^{n \times d}$ if, for all $\mathbf{x} \in \mathbb{R}^d$,

$$(1-\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2 \leq \|\tilde{\mathbf{A}}\mathbf{x}\|_2^2 \leq (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_2^2$$

What can you do with this sketch?

Application 1: Solve regression problems approximately. Consider the matrix $[\mathbf{A}, \mathbf{b}]$ with n rows and $d+1$ columns.

Construct a subspace embedding sketch $[\tilde{\mathbf{A}}, \tilde{\mathbf{b}}]$ with $O(d/\epsilon^2)$ rows. Let $\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\tilde{\mathbf{A}}\mathbf{x} - \tilde{\mathbf{b}}\|_2$. Then:

$$\|\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}\|_2 \leq (1+\epsilon) \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

Why?

SKETCHED REGRESSION

For $n \times d$ matrices, there are methods for computing a subspace embedding in $O(nd)$ time.

After sketching, reduce the cost of solving an $n \times d$ regression problem from $O(nd^2)$ to $O(d^3/\epsilon^2)$.

You can improve the ϵ dependence to $O(d/\epsilon)$, but still this is not a very well used technique.**

** Some interesting uses in approximate reorthogonalization (Joel Tropp, Davide Palitta). Here you are essentially solving a regression problem, but don't need a very good solution.

Application 2: Preconditioning. [Rokhlin, Tygert 2008]

The subspace embedding guarantee for $\epsilon = 1/2$.

$$\frac{1}{2}\|\tilde{\mathbf{A}}\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq 2\|\tilde{\mathbf{A}}\mathbf{x}\|_2^2$$

Equivalently, for all $\mathbf{x}$,

$$\frac{1}{2}\mathbf{x}^T\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\mathbf{x} \leq \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} \leq 2 \cdot \mathbf{x}^T\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\mathbf{x}.$$

Multiplying on left and right by $(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1/2}$, we have that:

$$\frac{1}{2}\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1/2}\mathbf{A}^T\mathbf{A}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1/2}\mathbf{x} \leq 2 \cdot \mathbf{x}^T\mathbf{x}.$$

In other words, $\mathbf{A}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1/2}$ has singular values between 1/2 and 2. It's a super well conditioned matrix!

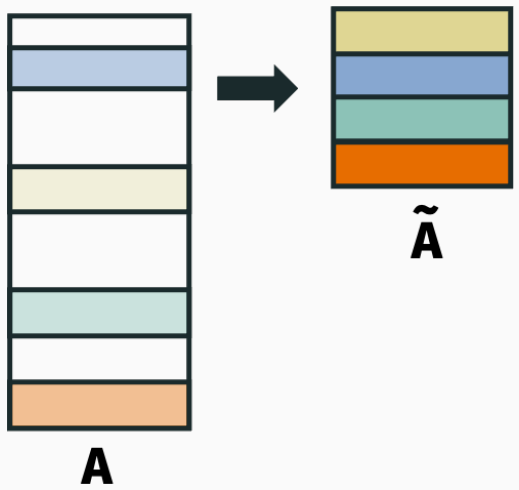
Punchline: $\mathbf{P} = (\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{1/2}$ is a very good preconditioner for $\mathbf{A}$. Solve the problem $\min_{\mathbf{y}} \|\mathbf{A}\mathbf{P}^{-1} - \mathbf{b}\|_2$. Set $\mathbf{x} = \mathbf{P}^{-1}\mathbf{y}$.

Using preconditioned CG or whatever algorithm you want, once you compute the sketch $\tilde{\mathbf{A}}$, can solve $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ in $O((nd + d^3)\log(1/\epsilon))$ time.

Started with a very poor dependence on $1/\epsilon$. Got to something much more reasonable by combining with more classical methods. This is also how subspace embeddings/graph sparsifiers have been used in fast Laplacian system solvers since [Spielman, Teng 2004].

SUBSAMPLING FOR SUBSPACE EMBEDDING

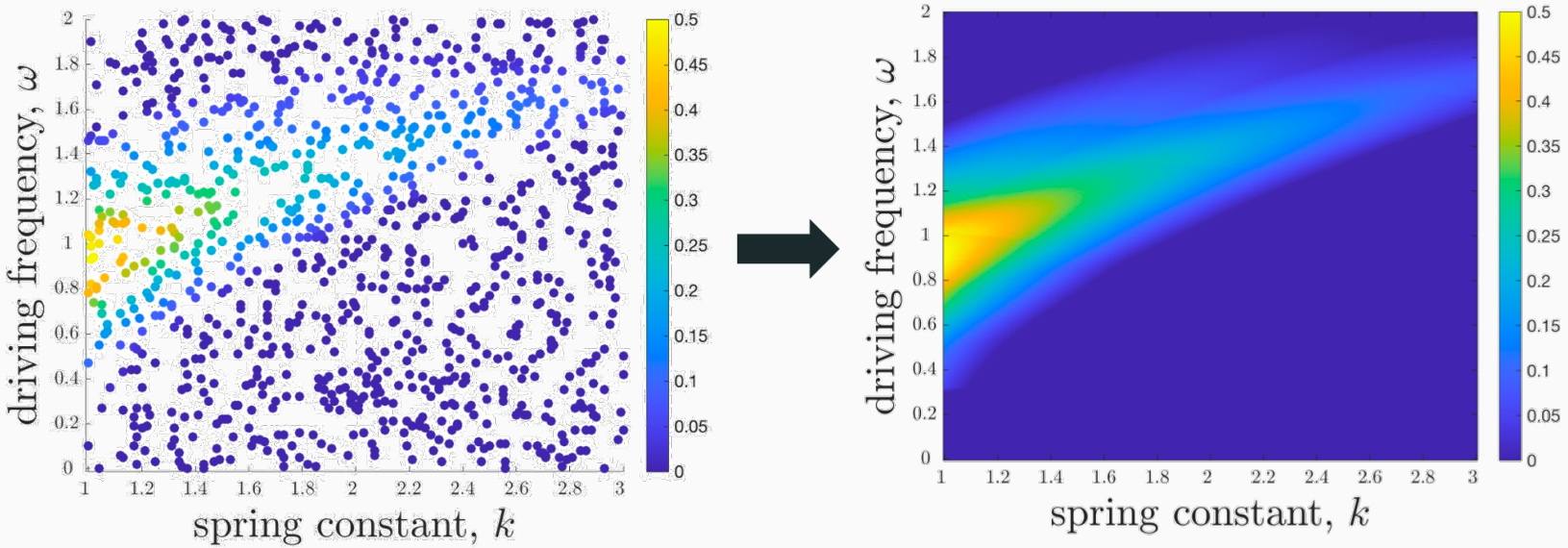
If you sample rows of $\mathbf{A}$ via their statistical leverage scores, you can obtain a subspace embedding $\tilde{\mathbf{A}}$ with $O(d \log d/\epsilon^2)$ rows [Spielman, Srivastava 2008]. Lots of work by Petros, Michael, others. And other sampling methods (DPPs, adaptive sampling, etc.)



These scores are relatively easy to compute. For row i, sampling probability is proportional to $\mathbf{a}_i^T(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{a}_i$.

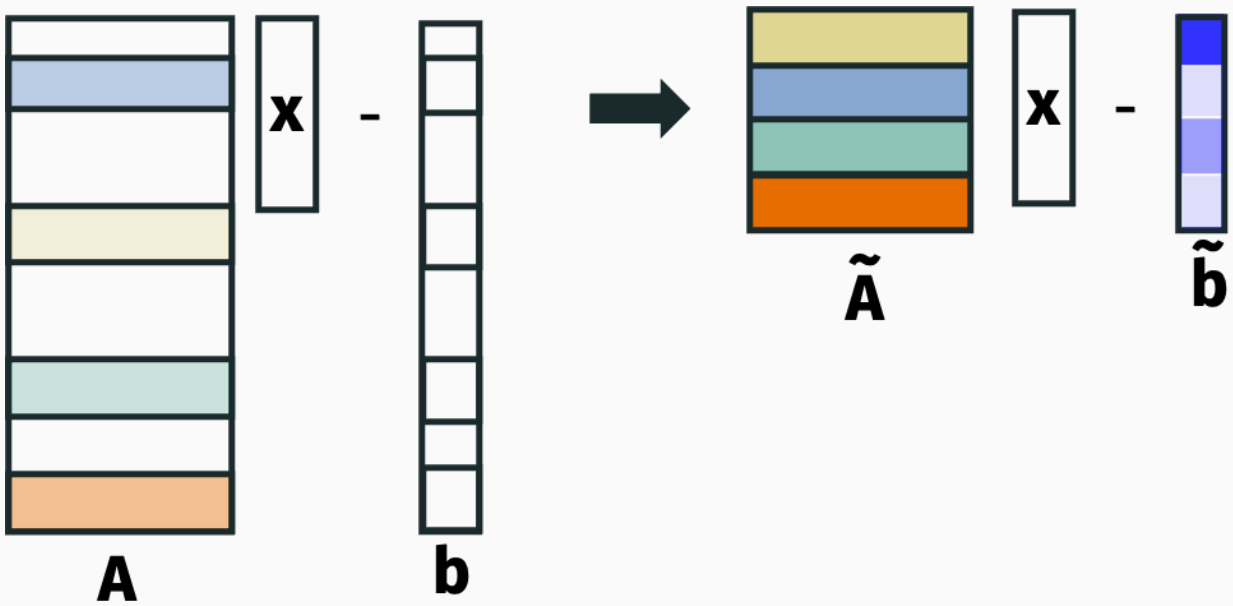
Application 3: Active learning.

Given data points $\mathbf{g}_1, \ldots, \mathbf{g}_n$ and ability to query the values of a function b_i for any i you want. Want to fit a simple function $f \in \mathcal{F}$ to the data using as few value queries as possible.



E.g. in applications to parametric PDEs, you want to fit a quantity of interest surface for use in uncertainty quantification. Each point sampled requires solving the PDE.

When your function class f is linear (e.g. polynomials) you can solve this problem with subspace embeddings. Set $\mathbf{a}_i = [\phi_1(\mathbf{g}_i), \ldots, \phi_d(\mathbf{g}_i)]$. Minimize $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$.



Only need to observe data label for entries of $\mathbf{b}$ where the corresponding row of $\mathbf{A}$ was sampled!

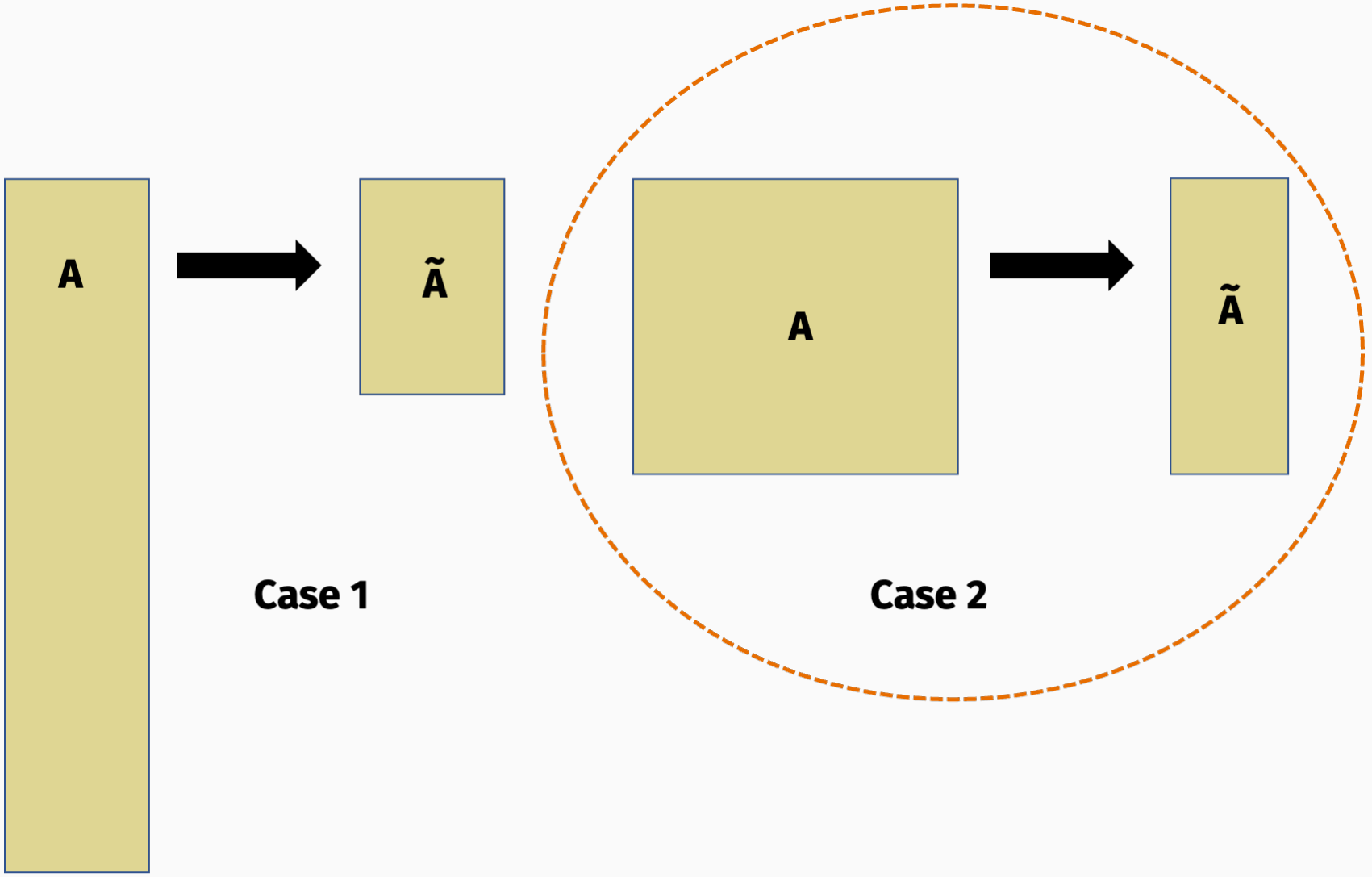
Punchline: For a function class $\mathcal{F}$ which is linear with d features, learn $\tilde{f}$ such that:

$$\sum_{i=1}^{n}(\tilde{f}(\mathbf{g}_i) - b_i)^2 \leq (1+\epsilon)\min_{f\in\mathcal{F}}\sum_{i=1}^{n}(f(\mathbf{g}_i) - b_i)^2$$

Use just $O(d \log d/\epsilon)$ function queries. Computing the optimal f would have required n.

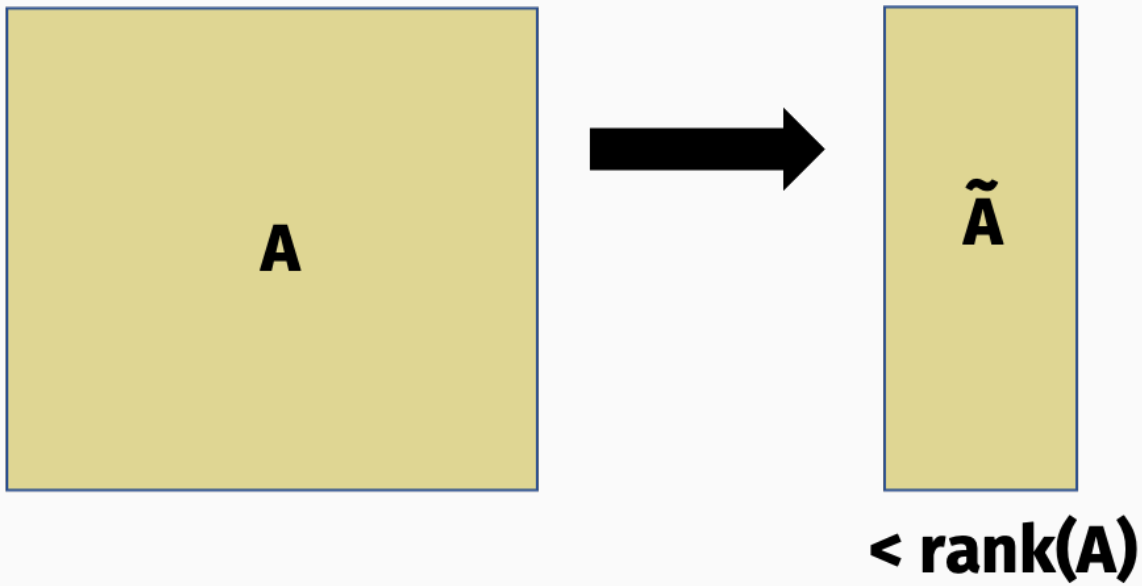
Matrix sketching was "rediscovered" by the parametric PDE/UQ community [Hampton, Doostan 2014], [Cohen, Migliorati 2016] for use in fitting polynomials, sparse polynomials, sparse Fourier functions, etc.

SECOND CASE



SECOND CASE

Cannot hope for a guarantee as strong as subspace embedding.



Usually hope to preserve information about the top singular vector subspaces of A.

PROJECTION-COST PRESERVING SKETCHES

[Feldman, Schmidt, Sohler 2013] [Cohen, Elder, Musco, Musco, Persu 2015]

Definition (Projection-Cost Preserving Sketching)

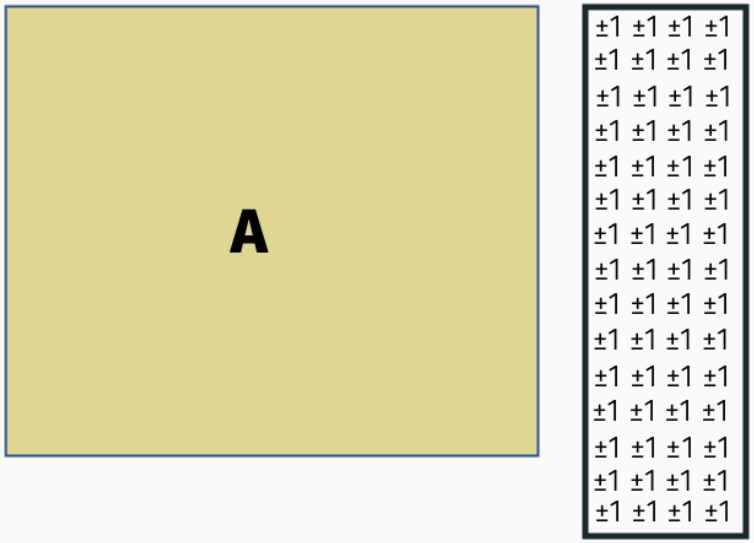
A matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times m}$ is a projection-cost preserving sketch for $\mathbf{A} \in \mathbb{R}^{n \times d}$ if for all rank k projection matrices $\mathbf{P} \in \mathbb{R}^{n \times n}$,

$$(1-\epsilon)\|\mathbf{A}-\mathbf{PA}\|_F^2 \leq \|\tilde{\mathbf{A}}-\mathbf{P}\tilde{\mathbf{A}}\|_F^2 \leq (1+\epsilon)\|\mathbf{A}-\mathbf{PA}\|_F^2.$$

Just one of many possible ways of dealing with low-rank problems. I think this definition is conceptually easy to wrap your head around, but doesn't lead to the tightest results.

PROJECTION-COST PRESERVING SKETCHES

How do you compute a projection-cost preserving sketch?



Multiply by a random matrix with $O(k/\epsilon^2)$ columns. Or a sparse random matrix, subsampling matrix, etc. Everything that worked for subspace embeddings works here.

Application 1: Low-rank approximation / Randomized SVD.

Let $\tilde{\mathbf{Q}} = \arg\min_{\mathbf{Q} \in \mathbb{R}^{n \times k}} \|\tilde{\mathbf{A}} - \mathbf{Q}\mathbf{Q}^T\tilde{\mathbf{A}}\|_F$. Then:

$$\|\mathbf{A} - \tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T\tilde{\mathbf{A}}\|_F \leq (1+\epsilon) \min_{\mathbf{Q} \in \mathbb{R}^{n \times k}} \|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T\tilde{\mathbf{A}}\|_F$$

In other words, $\tilde{\mathbf{Q}}$ gives a near-optimal low-rank approximation. Can be computed in $O(ndk/\epsilon^2)$ time, even if you are using a dense sketching matrix.

This result can be improved. See work by [Sarlos 2006], [Halko, Martinsson, Tropp 2011] [Clarkson, Woodruff 2013], many more. Notably, the ϵ dependence can be improved to $1/\epsilon$.

Constrained low-rank approximation

Application 1a: Constrained low-rank approximation.

For any constraint set $\mathcal{S}$, $\tilde{Q} = \arg\min_{Q \in \mathbb{R}^{n \times k}, Q \in \mathcal{S}} \|\tilde{A} - QQ^T\tilde{A}\|_F$. Then:

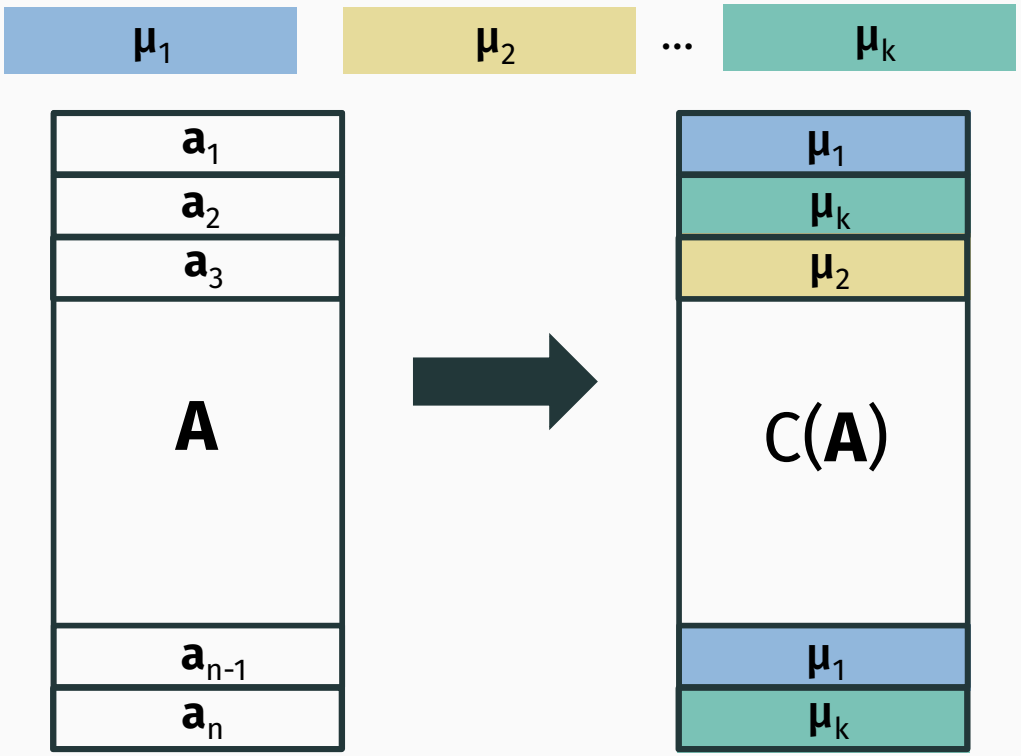
$$\|A - \tilde{Q}\tilde{Q}^T\tilde{A}\|_F \leq (1+\epsilon) \min_{Q \in \mathbb{R}^{n \times k}, Q \in \mathcal{S}} \|A - QQ^T\tilde{A}\|_F$$

Cool observation: k-means clustering is a constrained low rank approximation problem! [Drineas, Frieze, Kannan, Vempala, Vinay 2004].

Application 1a: Constrained low-rank approximation.

k-means clustering == low rank approximation

$$\min \sum_{i=1}^{n} \|a_i - \mu(a_i)\|_2^2 = \|A - C(A)\|_F^2$$



Punchline: Can solves k-means approximately by first projecting data pojnts to $O(k)$ dimensional space. Usually choosing ϵ to be constant is just fine since it is such a noisy problem to begin with.

There have been improvements getting this result to $O(\log k)$ that don't take the linear algebraic approach. See e.g. [Makarychev, Makarychev, Razenshteyn 2019].

LOW-RANK APPROXIMATION

For unconstrained low-rank approximation, how do sketching methods compare to iterative methods with random starts?

Sketching

- $O(k/\epsilon)$ matrix-vector multiplies. $1/\epsilon$ is "real".
- $(1+\epsilon)$ accurate low-rank approx. in Frobenius norm.
- Can do all multiplication at once.

Block (or single vector!) Krylov

- $O(k/\sqrt{\epsilon})$ matrix-vector multiplies. Often much better when you have singular value gaps.
- $(1+\epsilon)$ accurate low-rank approx. in Frobenius and spectral norms.
- Adaptive/multipass.

Sketching methods have a place in lots of applications where accuracy is not critical (e.g. $\epsilon = 1/2$).

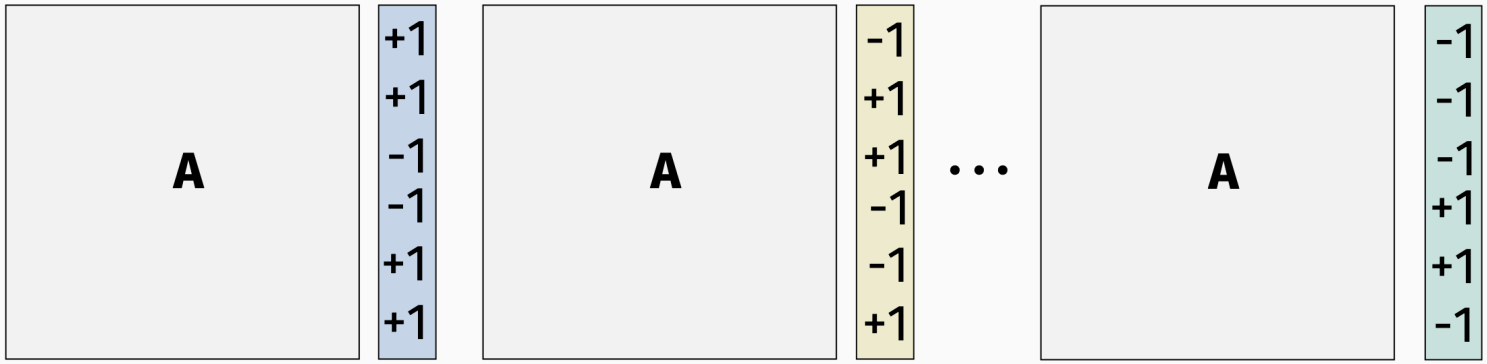
Application 2: Variance Reduction for Trace Estimation

Hutchinson's stochastic trace estimator

Goal: Approximate the trace of a PSD matrix $\mathbf{A}$ given ability to multiply $\mathbf{A}$ by vectors.

Hutchinson 1991, Girard 1987:

- Draw $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ i.i.d. with random $\{+1, -1\}$ entries.
- Return $\tilde{T} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i^T \mathbf{A} \mathbf{x}_i$ as approximation to $\mathbf{tr}(\mathbf{A})$.



Claim (Avron, Toledo 2011, Cortinovis, Kressner 2020)

Let $\tilde{T}$ be the trace estimate returned by Hutchinson's method. If $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, then with probability $(1-\delta)$,

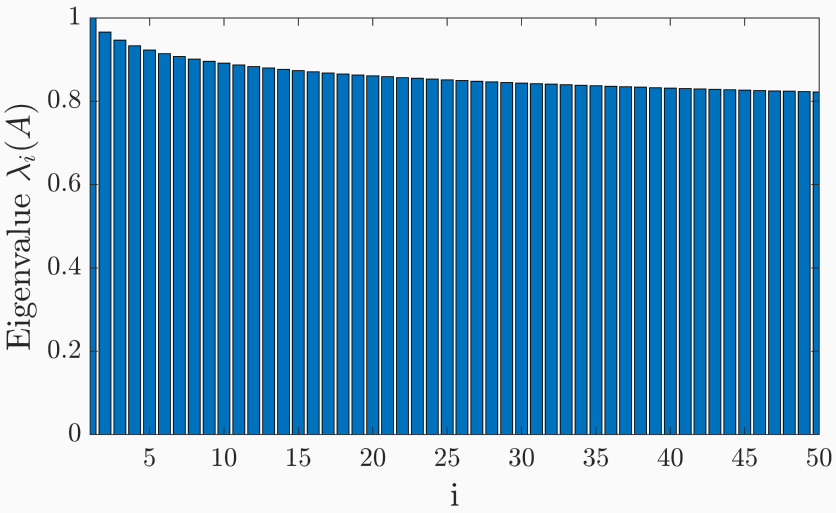
$$\left|\tilde{T} - \operatorname{tr}(\mathbf{A})\right| \leq \epsilon \|\mathbf{A}\|_F.$$

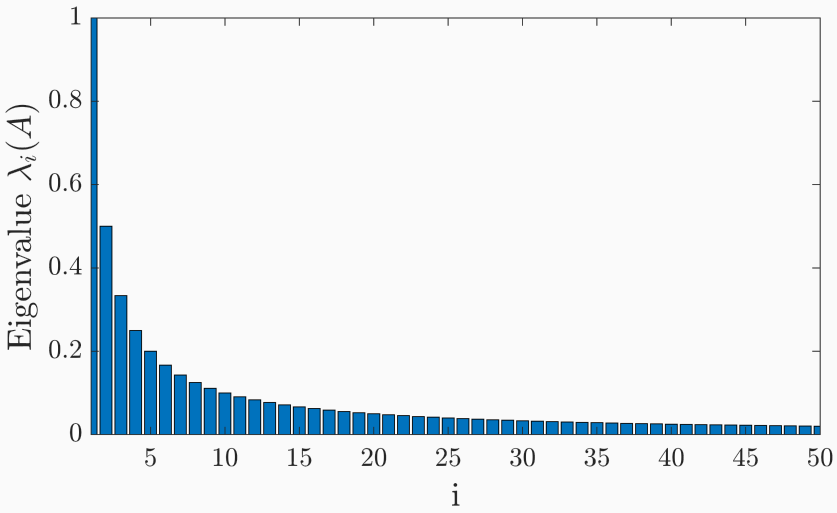
If $\mathbf{A}$ is symmetric positive semidefinite (PSD) with eigenvalues $\lambda_1, \ldots, \lambda_n$, then

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n} \lambda_i^2} \leq \sum_{i=1}^{n} \lambda_i = \operatorname{tr}(\mathbf{A}).$$

Corollary: For PSD A, $\quad (1-\epsilon)\operatorname{tr}(\mathbf{A}) \leq \tilde{T} \leq (1+\epsilon)\operatorname{tr}(\mathbf{A})$.

Observation: $\|A\|_F$ is only close to $\mathrm{tr}(A)$ if A has large eignenvalues.





VARIANCE REDUCTION FOR TRACE ESTIMATION

Idea: Project off large eigenvalues directly. [Saibaba, Alexanderian, Ipsen 2017]. I.e. compute approximation to top k singular vector subspace $\mathbf{Q} \in \mathbb{R}^{n \times k}$ and write:

$$\mathrm{tr}(\mathbf{A}) = \mathrm{tr}(\mathbf{A}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)) + \mathrm{tr}(\mathbf{A}\mathbf{Q}\mathbf{Q}^T)$$

Hutch++ Algorithm: Estimate $\mathrm{tr}(\mathbf{A}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T))$ using Hutchinson's estimator. Compute $\mathrm{tr}(\mathbf{A}\mathbf{Q}\mathbf{Q}^T)$ using k matrix vector products.

Main Observation: Variance of Hutchinson's estimator in computing $\mathrm{tr}(\mathbf{A}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T))$ depends on $\|\mathbf{A} - \mathbf{A}\mathbf{Q}\mathbf{Q}^T\|_F$.

Not critical to get this close to $(1 + \epsilon)$. Even if variance is suboptimal by a 2x factor, make up for this with more iterations of Hutchinsons!

VARIANCE REDUCTION FOR TRACE ESTIMATION

Claim (Meyer, Musco, Musco, Woodruff, 2021)

Given a sketching method for computing an $O(1)$ approximate k-rank approximation using $O(k)$ matrix-vector multiplications with a given PSD matrix $\mathbf{A}$, The Hutch++ algorithm returns an estimate $\tilde{T}$ satisfying:

$$|\tilde{T} - \mathrm{tr}(\mathbf{A})| \leq \epsilon \, \mathrm{tr}(\mathbf{A})$$

using just $O(1/\epsilon)$ matrix-vector multiplications in total.

Takeaway: Quadratic improvement on the $O(1/\epsilon^2)$ required by Hutchinson's method. Doesn't pay the high ϵ dependencies we typically associate with sketching.

COMMON THEME?

Poor accuracy is an issue with matrix sketching methods. A lot of the most compelling applications combine sketching with other refinement techniques. **Two examples so far:**

1. Using constant-factor subspace embedding as a preconditioner for linear systems.
2. Using constant-factor low-rank approximation sketch as a variance reduction tool for trace estimation.

Application 3: Analyzing iterative SVD methods.

Question: How many iterations does it take for a block Krylov method to converge to a near optimal low-rank approximation? I.e. to find $\mathbf{Q} \in \mathbb{R}^{n \times k}$ such that:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T\mathbf{A}\| \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|$$

This problem is distinct from asking how long is takes for $\mathbf{Q}$ to converge the top singular vectors of $\mathbf{A}$ [Drineas, Ipsen, 2019].

Typical analysis approach: View Krylov subspace method as returning a sketch of $p(\mathbf{A})$ for some polynomial p:

$$p(\mathbf{A})\mathbf{G}.$$

Typical analysis approach: View Krylov subspace method as returning a sketch of $p(\mathsf{A})$ for some polynomial p:

$$\mathsf{B} = p(\mathsf{A})\mathsf{G}.$$

Choose sketch size G as small as possible: $n \times k$ for rank k approximation.

Sketching Guarantee: From sketch $\mathsf{BG} \in \mathbb{R}^{n \times k}$ can compute $\mathsf{Q} \in \mathbb{R}^{n \times k}$ such that:

$$\|\mathsf{B} - \mathsf{Q}\mathsf{Q}^T\mathsf{B}\|_F \leq cnk\|\mathsf{B} - \mathsf{Q}\mathsf{Q}^T\mathsf{B}\|_F.$$

This is on the surface a really weak guarantee! But leads to strong analysis of block Krylov methods. The cnk term ends up inside a log.

OPEN DIRECTIONS

- Other ways to combine sketches with iterative or high accuracy methods?
- Better abstractions for sketching guarantees?
- Sketches for active learning for more function classes.
- Applying sketches to infinitely tall matrices / linear operators.
- What are the right practical methods and why do they work? Adaptive sampling, rank revealing QR, determinantal point process sampling, etc.
- Sketches for structured matrices that don't destroy structure.

QUESTIONS?