Universal Matrix Sparsifiers and Fast Deterministic Algorithms for Linear Algebra

Cameron Musco
University of Massachusetts Amherst

With: Rajarshi Bhattacharjee (UMass), Gregory Dexter (Purdue), Archan Ray (UMass), Sushant Sachdeva (Univ. Toronto) and David Woodruff (CMU)

Research Question

What non-trivial linear algebraic problems can be solved deterministically in less than n^ω time on general $n \times n$ input matrices?[1]

[1] Here $\omega \approx 2.37$ is the matrix multiplication exponent.

Research Question

What non-trivial linear algebraic problems can be solved deterministically in less than n^ω time on general $n \times n$ input matrices?[1]

- For structured matrices (Toeplitz, low-rank, graph structured, etc.) many fast deterministic methods are known.
- Randomized methods give fast approximation methods for general input matrices for many problems (e.g., singular value and eigenvalue estimation, low-rank approximation, etc.).
- But what about deterministic methods for unstructured input matrices? Very little is known.

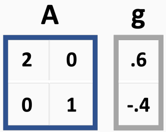
[1] Here $\omega \approx 2.37$ is the matrix multiplication exponent.

Example 1: Computing the Spectral Norm

Problem: Given symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ compute an approximation to the spectral norm $\|\mathbf{A}\|_2 = \max_{\mathbf{x} \in \mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$.

Problem: Given symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ compute an approximation to the spectral norm $\|\mathbf{A}\|_2 = \max_{\mathbf{x} \in \mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$.

- Can compute $\|\mathbf{A}\|_2$ up to small relative error in $O(n^2 \cdot \log n)$ time by applying the power method (or Krylov methods) for $O(\log n)$ iterations with a random start vector $g \in \mathbb{R}^n$.

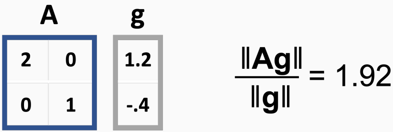


$$\frac{\|\mathbf{Ag}\|}{\|\mathbf{g}\|} = 1.75$$

Example 1: Computing the Spectral Norm

Problem: Given symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$ compute an approximation to the spectral norm $\|\mathbf{A}\|_2 = \max_{\mathbf{x}\in\mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$.

- Can compute $\|\mathbf{A}\|_2$ up to small relative error in $O(n^2 \cdot \log n)$ time by applying the power method (or Krylov methods) for $O(\log n)$ iterations with a random start vector $g \in \mathbb{R}^n$.



Example 1: Computing the Spectral Norm

Problem: Given symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ compute an approximation to the spectral norm $\|\mathbf{A}\|_2 = \max_{\mathbf{x} \in \mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$.

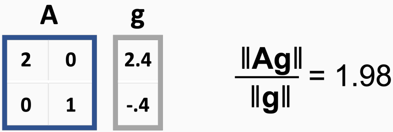
- Can compute $\|\mathbf{A}\|_2$ up to small relative error in $O(n^2 \cdot \log n)$ time by applying the power method (or Krylov methods) for $O(\log n)$ iterations with a random start vector $g \in \mathbb{R}^n$.

$$\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{g} = \begin{bmatrix} 2.4 \\ -.4 \end{bmatrix} \qquad \frac{\|\mathbf{Ag}\|}{\|\mathbf{g}\|} = 1.98$$

Example 1: Computing the Spectral Norm

Problem: Given symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ compute an approximation to the spectral norm $\|\mathbf{A}\|_2 = \max_{\mathbf{x} \in \mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$.

- Can compute $\|\mathbf{A}\|_2$ up to small relative error in $O(n^2 \cdot \log n)$ time by applying the power method (or Krylov methods) for $O(\log n)$ iterations with a random start vector $g \in \mathbb{R}^n$.



- Randomness is crucial here!
- If we pick g deterministically, in the worst-case it could be orthogonal to $\mathbf{A}$'s top singular vector(s), and we cannot ensure any approximation guarantee.

Example 1: Computing the Spectral Norm

Basic Open Question: Can any non-trivial approximation to $\|\mathbf{A}\|_2$ be computed deterministically in $o(n^\omega)$ time?

- What would a fast deterministic algorithm for this problem even look like?
- We know e.g., that, unlike power method or Krylov methods, it cannot be based solely on computing matrix-vector products with $\mathbf{A}$.

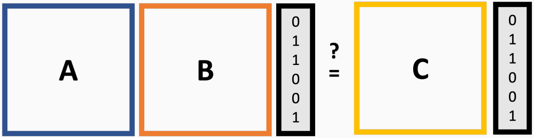
Example 2: Matrix Product Testing

Problem: Given $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times n}$ check if $\mathbf{AB} = \mathbf{C}$.

Example 2: Matrix Product Testing

Problem: Given $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times n}$ check if $\mathbf{AB} = \mathbf{C}$.

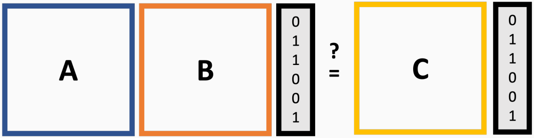
- Can answer with high probability in $O(n^2)$ time via Freivald's algorithm. Pick random $\mathbf{g} \in \mathbb{R}^n$ and check if $\mathbf{ABg} = \mathbf{Cg}$.
- Repeat multiple trials to boost confidence.



Example 2: Matrix Product Testing

Problem: Given $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times n}$ check if $\mathbf{AB} = \mathbf{C}$.

- Can answer with high probability in $O(n^2)$ time via Freivald's algorithm. Pick random $\mathbf{g} \in \mathbb{R}^n$ and check if $\mathbf{ABg} = \mathbf{Cg}$.
- Repeat multiple trials to boost confidence.



- No deterministic approach is known beyond directly computing $\mathbf{AB}$ in n^ω time and comparing the output with $\mathbf{C}$.
- As far as I am aware, no strong complexity theoretic implications of derandomizing Freivald's algorithm are known, and thus doing so remains plausible.

Broader Context

Understanding the role of randomness in fast computation is central in theoretical computer science.

Broader Context

Understanding the role of randomness in fast computation is central in theoretical computer science.

- In many settings (e.g., sublinear time algorithms, communication efficient algorithms), provable separations between randomized and deterministic algorithms exist.

Broader Context

Understanding the role of randomness in fast computation is central in theoretical computer science.

- In many settings (e.g., sublinear time algorithms, communication efficient algorithms), provable separations between randomized and deterministic algorithms exist.
- For running time complexity, no known separations exist. In fact, most complexity theorists believe that polynomial time deterministic algorithms are just as powerful as polynomial time randomized algorithms (i.e., that BQP = P).

Broader Context

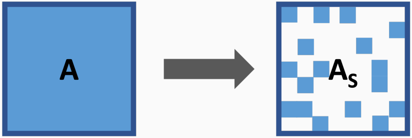
Understanding the role of randomness in fast computation is central in theoretical computer science.

- In many settings (e.g., sublinear time algorithms, communication efficient algorithms), provable separations between randomized and deterministic algorithms exist.
- For running time complexity, no known separations exist. In fact, most complexity theorists believe that polynomial time deterministic algorithms are just as powerful as polynomial time randomized algorithms (i.e., that BQP = P).
- Given the prevalence of randomized methods in numerical linear algebra today, it seems worth thinking about where/why they are needed, and if they can be replaced with clever enough deterministic approaches.

Our Contributions

Matrix Sparsification

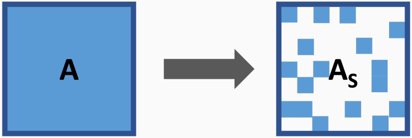
- Matrix sparsification is a key tool in randomized numerical linear algebra [Achlioptas McSherry '07, Drineas Zouzias '11].
- For any $\mathbf{A} \in \mathbb{R}^{n \times n}$, with $\|\mathbf{A}\|_\infty \leq 1$, if we form $\mathbf{A}_S$ by randomly sampling $s = O(\frac{n \log n}{\epsilon^2})$ entries of $\mathbf{A}$ and scaling the sampled entries by n^2/s, then with high probability, $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon \cdot n$.



- Proven via standard matrix concentration bounds.

Matrix Sparsification

- Matrix sparsification is a key tool in randomized numerical linear algebra [Achlioptas McSherry '07, Drineas Zouzias '11].
- For any $\mathbf{A} \in \mathbb{R}^{n \times n}$, with $\|\mathbf{A}\|_\infty \leq 1$, if we form $\mathbf{A}_S$ by randomly sampling $s = O(\frac{n \log n}{\epsilon^2})$ entries of $\mathbf{A}$ and scaling the sampled entries by n^2/s, then with high probability, $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon \cdot n$.



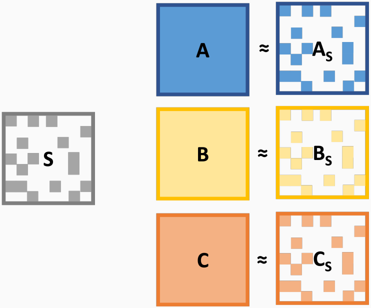
- Proven via standard matrix concentration bounds.
- $\mathbf{A}_S$ can be used in place of $\mathbf{A}$ to efficiently approximate singular values, compute a low-rank approximation, etc.

Universal Sparsifiers for PSD Matrices

Main Result: The above matrix sparsification result can be fully derandomized when A is positive semidefinite (PSD).

Main Result: The above matrix sparsification result can be fully derandomized when $\mathbf{A}$ is positive semidefinite (PSD).

- There exists a fixed set $S \subset [n] \times [n]$ with $|S| = O(\frac{n}{\epsilon^2})$ such that simultaneously for all PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $\|\mathbf{A}\|_\infty \leq 1$, $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon \cdot n$.



Main Result: The above matrix sparsification result can be fully derandomized when $\mathbf{A}$ is positive semidefinite (PSD).

- There exists a fixed set $S \subset [n] \times [n]$ with $|S| = O(\frac{n}{\epsilon^2})$ such that simultaneously for all PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $\|\mathbf{A}\|_\infty \leq 1$, $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon \cdot n$.
- We call S a universal sparsifier.

Main Result: The above matrix sparsification result can be fully derandomized when $\mathbf{A}$ is positive semidefinite (PSD).

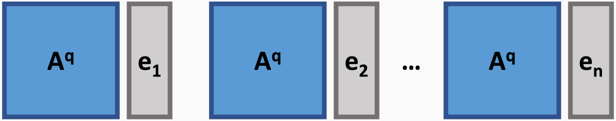
- There exists a fixed set $S \subset [n] \times [n]$ with $|S| = O(\frac{n}{\epsilon^2})$ such that simultaneously for all PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $\|\mathbf{A}\|_\infty \leq 1$, $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon \cdot n$.
- We call S a universal sparsifier.
- The above result gives an $O(\frac{n}{\epsilon^2})$ time deterministic algorithm for constructing $\mathbf{A}_S$ satisfying $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon n$ given any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$. The algorithm simply reads the entries of $\mathbf{A}$ corresponding to the elements of S.
- These elements are fixed (and independent of $\mathbf{A}$) and thus the algorithm is deterministic.

Corollary: There exists a $O(\frac{n^2 \log n}{\epsilon^3})$ time deterministic algorithm that, given PSD $\mathbf{A}$ with $\|\mathbf{A}\|_\infty \leq 1$, approximates $\|\mathbf{A}\|_2$ to $\pm\epsilon n$ error .

Application to Deterministic Singular Value Approximation

Corollary: There exists a $O(\frac{n^2 \log n}{\epsilon^3})$ time deterministic algorithm that, given PSD $\mathbf{A}$ with $\|\mathbf{A}\|_\infty \leq 1$, approximates $\|\mathbf{A}\|_2$ to $\pm\epsilon n$ error .

- We can deterministically compute $\mathbf{A}_S$ with just $O(\frac{n}{\epsilon^2})$ entries such that $\|\mathbf{A} - \mathbf{A}_S\| \leq \epsilon n$.
- To approximate $\|\mathbf{A}\|_2$ up to error $\pm\epsilon n$, it suffices to approximate $\|\mathbf{A}_S\|_2$ up to error $\pm\epsilon n$.
- Apply power method on $\mathbf{A}_S$ with q iterations and a full orthogonal basis of starting vectors in just nq mat-vecs $= O(\frac{n^2 q}{\epsilon^2})$ time.



Application to Deterministic Singular Value Approximation

Corollary: There exists a $O(\frac{n^2 \log n}{\epsilon^3})$ time deterministic algorithm that, given PSD $\mathbf{A}$ with $\|\mathbf{A}\|_\infty \leq 1$, approximates $\|\mathbf{A}\|_2$ to $\pm \epsilon n$ error .

- We can deterministically compute $\mathbf{A}_S$ with just $O(\frac{n}{\epsilon^2})$ entries such that $\|\mathbf{A} - \mathbf{A}_S\| \leq \epsilon n$.
- To approximate $\|\mathbf{A}\|_2$ up to error $\pm \epsilon n$, it suffices to approximate $\|\mathbf{A}_S\|_2$ up to error $\pm \epsilon n$.
- Apply power method on $\mathbf{A}_S$ with q iterations and a full orthogonal basis of starting vectors in just nq mat-vecs $= O(\frac{n^2 q}{\epsilon^2})$ time.
- At least one of the starting vectors will converge in $q = O(\frac{\log n}{\epsilon})$ iterations. So we obtain a $O(\frac{n^2 \log n}{\epsilon^3})$ time deterministic algorithm for approximating $\|\mathbf{A}_S\|_2$, and in turn $\|\mathbf{A}\|_2$, to $\pm \epsilon n$ error.

Application to Deterministic Singular Value Approximation

Corollary: There exists a $O(\frac{n^2 \log n}{\epsilon^3})$ time deterministic algorithm that, given PSD $\mathbf{A}$ with $\|\mathbf{A}\|_\infty \leq 1$, approximates $\|\mathbf{A}\|_2$ to $\pm\epsilon n$ error .

- We can deterministically compute $\mathbf{A}_S$ with just $O(\frac{n}{\epsilon^2})$ entries such that $\|\mathbf{A} - \mathbf{A}_S\| \leq \epsilon n$.
- To approximate $\|\mathbf{A}\|_2$ up to error $\pm\epsilon n$, it suffices to approximate $\|\mathbf{A}_S\|_2$ up to error $\pm\epsilon n$.
- Apply power method on $\mathbf{A}_S$ with q iterations and a full orthogonal basis of starting vectors in just nq mat-vecs $= O(\frac{n^2 q}{\epsilon^2})$ time.
- At least one of the starting vectors will converge in $q = O(\frac{\log n}{\epsilon})$ iterations. So we obtain a $O(\frac{n^2 \log n}{\epsilon^3})$ time deterministic algorithm for approximating $\|\mathbf{A}_S\|_2$, and in turn $\|\mathbf{A}\|_2$, to $\pm\epsilon n$ error.
- We actually show that we can approximate all singular values of $\mathbf{A}$ to $\pm\epsilon n$ error deterministically in $O(\frac{n^2 \log n}{\epsilon^6})$ time.

For non-PSD matrices, we show the existence of a universal sparsifier with $|S| = O(\frac{n}{\epsilon^4})$ entries achieving $\|\mathsf{A} - \mathsf{A}_S\|_2 \leq \epsilon \cdot \max(n, \|\mathsf{A}\|_1)$, where $\|\mathsf{A}\|_1$ is the trace norm (the sum of A's singular values).

Universal Sparsifiers for Non-PSD Matrices

For non-PSD matrices, we show the existence of a universal sparsifier with $|S| = O(\frac{n}{\epsilon^4})$ entries achieving $\|\mathsf{A} - \mathsf{A}_S\|_2 \leq \epsilon \cdot \max(n, \|\mathsf{A}\|_1)$, where $\|\mathsf{A}\|_1$ is the trace norm (the sum of A's singular values).

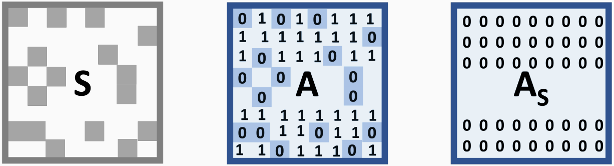
- Observe that this is weaker in both ϵ dependence and error guarantee than our result for PSD matrices, and than known randomized results for non-PSD matrices.
- We show that this loss is neccesary for deterministic algorithms – both the $1/\epsilon^4$ dependence and $\max(n, \|\mathsf{A}\|_1)$ scaling in the error cannot be improved.

Universal Sparsifiers for Non-PSD Matrices

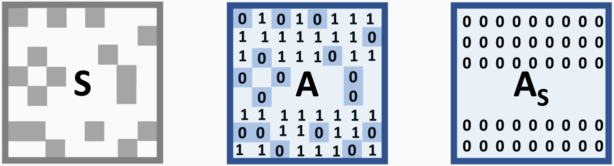
For non-PSD matrices, we show the existence of a universal sparsifier with $|S| = O(\frac{n}{\epsilon^4})$ entries achieving $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon \cdot \max(n, \|\mathbf{A}\|_1)$, where $\|\mathbf{A}\|_1$ is the trace norm (the sum of $\mathbf{A}$'s singular values).

- Observe that this is weaker in both ϵ dependence and error guarantee than our result for PSD matrices, and than known randomized results for non-PSD matrices.
- We show that this loss is neccesary for deterministic algorithms – both the $1/\epsilon^4$ dependence and $\max(n, \|\mathbf{A}\|_1)$ scaling in the error cannot be improved.

Wait....how is this even possible...

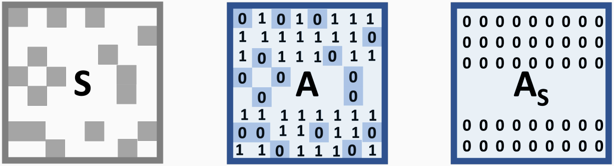


- Consider **A** which is 0 on all entries sampled by S and 1 everywhere else.
- $\|\mathbf{A} - \mathbf{A}_S\|_2 = \|\mathbf{A}\|_2 \approx n$. Very bad approximation.



- Consider $\mathbf{A}$ which is 0 on all entries sampled by S and 1 everywhere else.
- $\|\mathbf{A} - \mathbf{A}_S\|_2 = \|\mathbf{A}\|_2 \approx n$. Very bad approximation.
- Since we have claimed error bounded by $\epsilon \max(n, \|\mathbf{A}\|_1)$, this means that we must have $n \leq \epsilon \max(n, \|\mathbf{A}\|_1)$, so $\|\mathbf{A}\|_1 \geq n/\epsilon$.

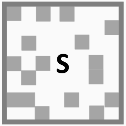
Hard Case



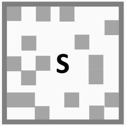
- Consider **A** which is 0 on all entries sampled by S and 1 everywhere else.
- $\|\mathbf{A} - \mathbf{A}_S\|_2 = \|\mathbf{A}\|_2 \approx n$. Very bad approximation.
- Since we have claimed error bounded by $\epsilon \max(n, \|\mathbf{A}\|_1)$, this means that we must have $n \leq \epsilon \max(n, \|\mathbf{A}\|_1)$, so $\|\mathbf{A}\|_1 \geq n/\epsilon$.
- One can check that this will indeed be the case. If the 0's in **A** are sufficiently well-spread, **A** will have one singular value near n, but must also many small singular values and thus large nuclear norm.

Proof of the PSD Case

- Intuitively, our universal sparsifier S should be 'well-spread'. It should 'look random'.

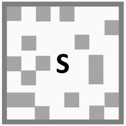


- Intuitively, our universal sparsifier S should be 'well-spread'. It should 'look random'.



- Concretely, in order for $\mathbf{A}_S$ to approximate any PSD $\mathbf{A}$, S should at least place roughly $\frac{s}{n^2} \cdot RC$ samples in any $R \times C$ submatrix.

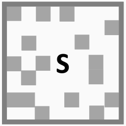
- Intuitively, our universal sparsifier S should be 'well-spread'. It should 'look random'.



- Concretely, in order for $\mathbf{A}_S$ to approximate any PSD $\mathbf{A}$, S should at least place roughly $\frac{s}{n^2} \cdot RC$ samples in any $R \times C$ submatrix.
- We will let S be the edge set of a Ramanujan expander graph. I.e., a d-regular graph whose second largest adjacency matrix eigenvalue is bounded by $2\sqrt{d-1}$.

Deterministic Sampling Via Expander Graphs

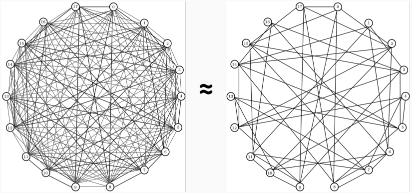
- Intuitively, our universal sparsifier S should be 'well-spread'. It should 'look random'.



- Concretely, in order for $\mathbf{A}_S$ to approximate any PSD $\mathbf{A}$, S should at least place roughly $\frac{s}{n^2} \cdot RC$ samples in any $R \times C$ submatrix.
- We will let S be the edge set of a Ramanujan expander graph. I.e., a d-regular graph whose second largest adjacency matrix eigenvalue is bounded by $2\sqrt{d-1}$.
- These graphs have the fastest random walk mixing times amongst all d-regular graphs and are important tools in spectral graph theory and pseudorandomness/derandomization.
- Efficient algebraic constructions have been known since the 80s [Lubotzky, Phillips, Sarnak '88 and Margulis '88].

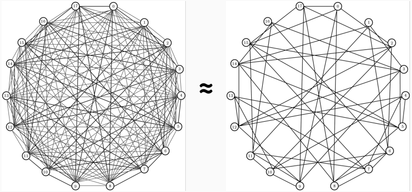
Expander Graphs as Approximations to the Complete Graph

Expander graphs can be thought of as optimal sparse approximations to the complete graph.



Expander Graphs as Approximations to the Complete Graph

Expander graphs can be thought of as optimal sparse approximations to the complete graph.

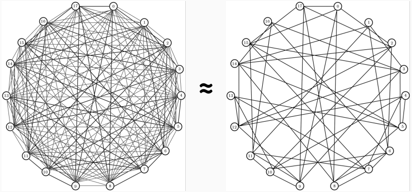


- Algebraically, letting $\mathbf{G}$ be the adjacency matrix of a Ramanujan graph scaled by $\frac{n}{d}$, and $\mathbf{1}$ be the all ones matrix,

$$\|\mathbf{1} - \mathbf{G}\|_2 \leq \frac{n}{d} \cdot 2\sqrt{d-1}.$$

Expander Graphs as Approximations to the Complete Graph

Expander graphs can be thought of as optimal sparse approximations to the complete graph.



- Algebraically, letting $\mathbf{G}$ be the adjacency matrix of a Ramanujan graph scaled by $\frac{n}{d}$, and $\mathbf{1}$ be the all ones matrix,

$$\|\mathbf{1} - \mathbf{G}\|_2 \leq \frac{n}{d} \cdot 2\sqrt{d-1}.$$

- Setting $d = O(1/\epsilon^2)$, we have $\|\mathbf{1} - \mathbf{G}\|_2 \leq \epsilon n$.

For any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can write our deterministically sparsified matrix as $\mathbf{A}_S = \mathbf{A} \circ \mathbf{G}$.

For any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can write our deterministically sparsified matrix as $\mathbf{A}_S = \mathbf{A} \circ \mathbf{G}$.

- Thus, we can write $\mathbf{A} - \mathbf{A}_S = \mathbf{A} \circ (\mathbf{1} - \mathbf{G})$.

Expander Graphs as Universal Sparsifiers

For any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can write our deterministically sparsified matrix as $\mathbf{A}_S = \mathbf{A} \circ \mathbf{G}$.

- Thus, we can write $\mathbf{A} - \mathbf{A}_S = \mathbf{A} \circ (\mathbf{1} - \mathbf{G})$.
- We want to show that $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon n$. I.e., for any unit $\mathbf{x}$, $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} \leq \epsilon n$.

Expander Graphs as Universal Sparsifiers

For any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can write our deterministically sparsified matrix as $\mathbf{A}_S = \mathbf{A} \circ \mathbf{G}$.

- Thus, we can write $\mathbf{A} - \mathbf{A}_S = \mathbf{A} \circ (\mathbf{1} - \mathbf{G})$.
- We want to show that $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon n$. I.e., for any unit $\mathbf{x}$, $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} \leq \epsilon n$.
- $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} = \mathbf{x}^T(\mathbf{A} \circ (\mathbf{1} - \mathbf{G}))\mathbf{x}$

Expander Graphs as Universal Sparsifiers

For any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can write our deterministically sparsified matrix as $\mathbf{A}_S = \mathbf{A} \circ \mathbf{G}$.

- Thus, we can write $\mathbf{A} - \mathbf{A}_S = \mathbf{A} \circ (\mathbf{1} - \mathbf{G})$.
- We want to show that $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon n$. I.e., for any unit $\mathbf{x}$, $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} \leq \epsilon n$.
- $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} = \mathbf{x}^T(\mathbf{A} \circ (\mathbf{1} - \mathbf{G}))\mathbf{x} = \sum_{i=1}^n \lambda_i \mathbf{x}^T(\mathbf{v}_i\mathbf{v}_i^T \circ (\mathbf{1} - \mathbf{G}))\mathbf{x}$.

For any PSD $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can write our deterministically sparsified matrix as $\mathbf{A}_S = \mathbf{A} \circ \mathbf{G}$.

- Thus, we can write $\mathbf{A} - \mathbf{A}_S = \mathbf{A} \circ (\mathbf{1} - \mathbf{G})$.
- We want to show that $\|\mathbf{A} - \mathbf{A}_S\|_2 \leq \epsilon n$. I.e., for any unit $\mathbf{x}$, $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} \leq \epsilon n$.
- $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} = \mathbf{x}^T(\mathbf{A} \circ (\mathbf{1} - \mathbf{G}))\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{x}^T(\mathbf{v}_i\mathbf{v}_i^T \circ (\mathbf{1} - \mathbf{G}))\mathbf{x}$.
- We observe that, letting $\mathbf{D}_i \in \mathbb{R}^{n \times n}$ be diagonal with entries corresponding to $\mathbf{v}_i$, we can rewrite the above as:

$$\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{x}^T\mathbf{D}_i(\mathbf{1} - \mathbf{G})\mathbf{D}_i\mathbf{x} \leq \epsilon n \cdot \sum_{i=1}^{n} \lambda_i \mathbf{x}^T\mathbf{D}_i^2\mathbf{x}.$$

Expander Graphs as Universal Sparsifiers

So Far: $x^T(A - A_S)x \leq \epsilon n \cdot \sum_{i=1}^{n} \lambda_i x^T D_i^2 x$.

So Far: $x^T(A - A_S)x \leq \epsilon n \cdot \sum_{i=1}^{n} \lambda_i x^T D_i^2 x$.

It suffices to bound $\sum_{i=1}^{n} \lambda_i x^T D_i^2 x \leq 1$.

So Far: $\mathbf{x}^T(\mathbf{A} - \mathbf{A}_S)\mathbf{x} \leq \epsilon n \cdot \sum_{i=1}^{n} \lambda_i \mathbf{x}^T \mathbf{D}_i^2 \mathbf{x}$.

It suffices to bound $\sum_{i=1}^{n} \lambda_i \mathbf{x}^T \mathbf{D}_i^2 \mathbf{x} \leq 1$.

$$\sum_{i=1}^{n} \lambda_i \mathbf{x}^T \mathbf{D}_i^2 \mathbf{x} = \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{n} \mathbf{x}(j)^2 \mathbf{v}_i(j)^2$$

So Far: $x^T(A - A_S)x \leq \epsilon n \cdot \sum_{i=1}^{n} \lambda_i x^T D_i^2 x$.

It suffices to bound $\sum_{i=1}^{n} \lambda_i x^T D_i^2 x \leq 1$.

$$\begin{aligned}\sum_{i=1}^{n} \lambda_i x^T D_i^2 x &= \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{n} x(j)^2 v_i(j)^2 \\ &= \sum_{j=1}^{n} x(j)^2 \sum_{i=1}^{n} \lambda_i v_i(j)^2\end{aligned}$$

So Far: $x^T(A - A_S)x \le \epsilon n \cdot \sum_{i=1}^n \lambda_i x^T D_i^2 x$.

It suffices to bound $\sum_{i=1}^n \lambda_i x^T D_i^2 x \le 1$.

$$\begin{aligned}\sum_{i=1}^n \lambda_i x^T D_i^2 x &= \sum_{i=1}^n \lambda_i \sum_{j=1}^n x(j)^2 v_i(j)^2 \\ &= \sum_{j=1}^n x(j)^2 \sum_{i=1}^n \lambda_i v_i(j)^2 \\ &= \sum_{j=1}^n x(j)^2 A_{i,i}\end{aligned}$$

So Far: $x^T(A - A_S)x \leq \epsilon n \cdot \sum_{i=1}^{n} \lambda_i x^T D_i^2 x$.

It suffices to bound $\sum_{i=1}^{n} \lambda_i x^T D_i^2 x \leq 1$.

$$\begin{aligned}
\sum_{i=1}^{n} \lambda_i x^T D_i^2 x &= \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{n} x(j)^2 v_i(j)^2 \\
&= \sum_{j=1}^{n} x(j)^2 \sum_{i=1}^{n} \lambda_i v_i(j)^2 \\
&= \sum_{j=1}^{n} x(j)^2 A_{i,i} \\
&\leq \sum_{j=1}^{n} x(j)^2 = 1.
\end{aligned}$$

Note on the Proof

- Our original proof (appearing in the current arXiv version) was *much more complex* and explicitly used the fact that G has roughly $\frac{d}{n} \cdot RC$ edges in any $R \times C$ submatrix.
- We also gave a non-constructive proof with tight ϵ dependencies based on showing that a random S simultaneously sparsifies all PSD $\mathbf{A}$ with high probability.
- We only recently came upon this much simpler proof, which gives an asymptotically tight bound of $O(n/\epsilon^2)$ samples and applies to any Ramanujan graph sampling scheme.

Next Steps

Next Steps

- Broadly, there seem to be many interesting questions related to the role of randomness in fast linear algebraic computation.
- Can we achieve a relative error approximation to $\|\mathbf{A}\|_2$ in $< n^\omega$ time deterministically?
- Our universal sparsifiers are both deterministic and data oblivious. I.e., S does not depend on the input matrix $\mathbf{A}$. Are there fast deterministic algorithms that inspect the entries of $\mathbf{A}$ to do better?
- Notably, for PSD $\mathbf{A}$ with $\|\mathbf{A}\|_\infty \leq 1$, there are randomized methods based on Nyström approximation that output $\tilde{\mathbf{A}}$ with $\|\mathbf{A} - \tilde{\mathbf{A}}\|_2 \leq \epsilon n$ using just $O(\frac{n \log n}{\epsilon})$ queries to $\mathbf{A}$ [Musco, Musco '17].
- Can we match this complexity with a deterministic algorithm?
- We have shown that we can in the very special case when $\mathbf{A}$ is binary, using a weaker (but still strong enough) family of expander graphs.

Next Steps

- Can we prove that derandomizing Freivald's algorithm is unlikely under some reasonable complexity theoretic assumption?
- We can show reductions – e.g. that derandomizing input sparsity time low-rank approximation or regression would imply derandomizing Freivald's and vice-versa.

THANKS! QUESTIONS?