COMPSCI 690RA: Randomized Algorithms and Probabilistic Data Analysis

Prof. Cameron Musco

University of Massachusetts Amherst. Spring 2022.

Lecture 5

Logistics

- Project guidelines and suggested topics have been posted on the Assignments Tab of the course page.
- One page proposal due Monday 3/7.
- Problem Set 2 was posted last Friday – due next Thursday, 3/3. We will not have a quiz this week – focus on the problem set and project brainstorming instead.

Summary

Last Time:

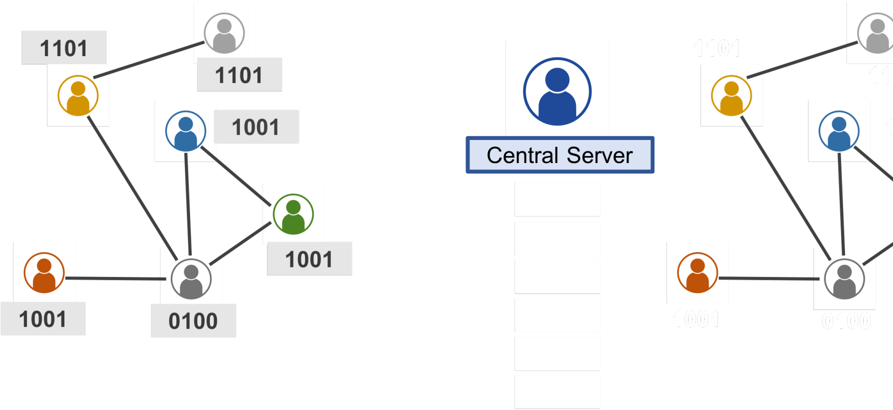
- Rabin fingerprint analysis. Applications to pattern matching (Rabin-Karp algorithm) and communication complexity (testing equality of n-bit strings using $O(\log n)$ bits).
- ℓ_0 sampling and low-communication graph connectivity.

Today:

- Quickly finish up graph sketching and streaming.
- Start on randomized methods for linear algebraic computation.
- Approximate matrix multiplication via sampling.
- Stochastic trace estimation.

A Graph Communication Problem

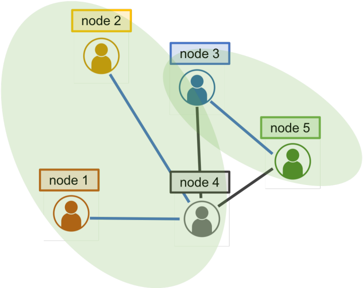
Consider n nodes, each only knows its own neighborhood. They want to send messages to a central server, who must then determine if the graph is connected.



Saw how to solve the problem with high probability using just $O(\log^c n)$ sized messages.

Simulating Boruvka's Algorithm via Sketches

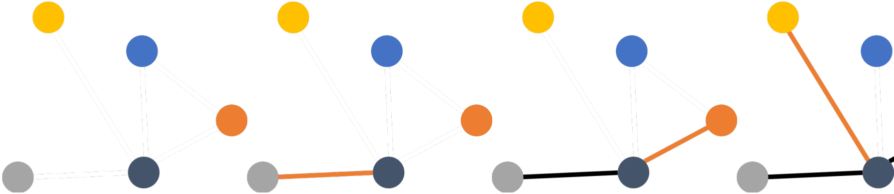
- For independent ℓ_0 sampling matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\log_2 n}$, each node computes $\mathbf{A}_j v_i$ and sends these sketches to the central server. $O(\log^c n)$ bits in total.
- The central server uses $\mathbf{A}_j v_1, \ldots, \mathbf{A}_j v_n$ to simulate the j^{th} step of Boruvka's algorithm – the sketch allows the server to recover one outgoing edge from each connected component.



v_3		v_5			
0		0		0	(1,2)
0		0		0	(1,3)
0		0		0	(1,4)
0		0		0	(1,5)
0	+	0	=	0	(2,3)
0		0		0	(2,4)
0		0		0	(2,5)
1		0		1	(3,4)
1		-1		0	(3,5)
0		-1		-1	(4,5)

A Graph Streaming Problem

Consider a setting where an algorithm must process a stream of edge insertions or deletions, which define a graph. At the end of the stream, the algorithm should output whether that graph is connected or not.



Algorithmic Question: How much memory must an algorithm use to solve this problem with high probability?

What is the worst-case memory required by a naive deterministic algorithm that just stores the current state of the graph? How can you improve on this when there are no edge deletions?

Solution via ℓ_0 sampling

- The algorithm samples independent ℓ_0 sampling matrices $\mathsf{A}_1, \ldots, \mathsf{A}_{\log_2 n}$ and maintains $\mathsf{A}_j v_u$ for all j and all $u \in [n]$, where $v_u \in \mathbb{R}^{\binom{n}{2}}$ is the incidence vector for node u.
- $O(n \log^c n)$ bits of storage in total.
- When an edge (u, v) is inserted or deleted, one entry is either incremented or decremented in each of v_u, v_v. The algorithm can update $\mathsf{A}_j v_u$ and $\mathsf{A}_j v_v$ in $O(\log^c n)$ time – simply set $\mathsf{A}_j v_u = \mathsf{A}_j v_u \pm \mathsf{A}_{j,k}$.

l_0 sampling matrix $\mathbf{A_j}$

1	-1	0	0	1	-1	0	1
-1	0	1	1	0	0	-1	0
1	1	-1	0	-1	-1	0	1
0	-1	-1	-1	1	1	1	0

v_u

1
0
0
-1
0
0
0
0

=

$\mathbf{A_j}v_u$

1
-2
1
1

l_0 sampling matrix $\mathbf{A_j}$

1	-1	0	0	1	-1	0	1
-1	0	1	1	0	0	-1	0
1	1	-1	0	-1	-1	0	1
0	-1	-1	-1	1	1	1	0

Approximate Matrix Multiplication

Matrix Multiplication Problem

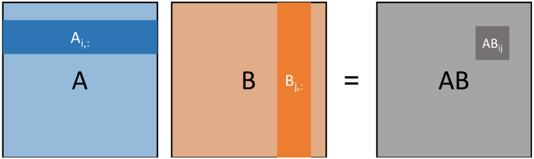
Given $A, B \in \mathbb{R}^{n \times n}$ would like to compute $C = AB$. Requires n^ω time where $\omega \approx 2.373$ in theory.

Today: We'll see how to compute an approximation in $O(n^2)$ time via a simple sampling approach.

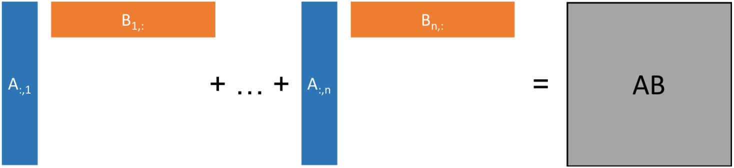
- One of the most fundamental algorithms in randomized numerical linear algebra. Forms the building block for many other algorithms.

Outer Product View of Matrix Multiplication

Inner Product View: $[AB]_{ij} = \langle A_{i,:}, B_{j,:} \rangle = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$.



Outer Product View: Observe that $C_k = A_{:,k} B_{k,:}$ is an $n \times n$ matrix with $[C_k]_{ij} = A_{jk} \cdot B_{kj}$. So $AB = \sum_{k=1}^{n} A_{:,k} B_{k,:}$



Basic Idea: Approximate **AB** by sampling terms of this sum.

Canonical AMM Algorithm

Approximate Matrix Multiplication (AMM):

- Fix sampling probabilities $p_1, \ldots, p_n$ with $p_i \geq 0$ and $\sum_{[n]} p_i = 1$.
- Select $\mathbf{i}_1, \ldots, \mathbf{i}_t \in [n]$ independently, according to the distribution $\Pr[\mathbf{i}_j = k] = p_k$.
- Let $\bar{\mathbf{C}} = \frac{1}{t} \cdot \sum_{j=1}^{t} \frac{1}{p_{\mathbf{i}_j}} \cdot A_{:,\mathbf{i}_j} B_{\mathbf{i}_j,:}$.

Claim 1: $\mathbb{E}[\bar{\mathbf{C}}] = AB$

$$\mathbb{E}[\bar{\mathbf{C}}] = \frac{1}{t} \sum_{j=1}^{t} \mathbb{E}\left[\frac{1}{p_{\mathbf{i}_j}} \cdot A_{:,\mathbf{i}_j} B_{\mathbf{i}_j,:}\right] = \frac{1}{t} \sum_{j=1}^{t} \sum_{k=1}^{n} p_k \cdot \frac{1}{p_k} \cdot A_{:,k} B_{k,:} = \frac{1}{t} \sum_{j=1}^{t} AB = AB$$

Weighting by $\frac{1}{p_{\mathbf{i}_j}}$ keeps the expectation correct.

AMM Error Analysis

Claim 2: $\mathbb{E}[\|AB - \overline{C}\|_F^2] = \frac{1}{t}\sum_{m=1}^{n} \frac{\|A_{:,m}\|_2^2 \cdot \|B_{m,:}\|_2^2}{p_m}$.

$$\mathbb{E}[\|AB - \overline{C}\|_F^2] = \sum_{k,\ell} \mathbb{E}[([AB]_{k\ell} - \overline{C}_{k\ell})^2] = \sum_{k,\ell} \mathrm{Var}[\overline{C}_{k\ell}].$$

$$\begin{aligned}
\mathrm{Var}[\overline{C}_{k\ell}] = \mathrm{Var}\left[\frac{1}{t}\sum_{j=1}^{t} \frac{1}{p_{\mathbf{i}_j}} A_{k,\mathbf{i}_j} B_{\mathbf{i}_j,\ell}\right] &= \frac{1}{t}\mathrm{Var}\left[\frac{1}{p_{\mathbf{i}_j}} A_{k,\mathbf{i}_j} B_{\mathbf{i}_j,\ell}\right] \\
&\leq \frac{1}{t}\sum_{m=1}^{n} p_m \cdot \frac{1}{p_m^2} \cdot A_{k,m}^2 \cdot B_{m,\ell}^2 \\
&= \frac{1}{t}\sum_{m=1}^{n} \frac{A_{k,m}^2 \cdot B_{m,\ell}^2}{p_m}
\end{aligned}$$

$$\mathbb{E}[\|AB - \overline{C}\|_F^2] \leq \frac{1}{t} \cdot \sum_{m=1}^{n}\sum_{k=1}^{n}\sum_{\ell=1}^{n} \frac{A_{k,m}^2 \cdot B_{m,\ell}^2}{p_m} = \sum_{m=1}^{n} \frac{\|A_{:,m}\|_2^2 \cdot \|B_{m,:}\|_2^2}{p_m}$$

Optimal Sampling Probabilities

Claim 2: $\mathbb{E}[\|AB - \overline{C}\|_F^2] = \frac{1}{t}\sum_{m=1}^n \frac{\|A_{:,m}\|_2^2 \cdot \|B_{m,:}\|_2^2}{p_m}$.

How should we set $p_1, \ldots, p_n$ to minimize this expected error?

Set $p_m = \frac{\|A_{:,m}\|_2 \cdot \|B_{m,:}\|_2}{\sum_{k=1}^n \|A_{:,k}\|_2 \cdot \|B_{k,:}\|_2}$, giving:

$$\mathbb{E}[\|AB - \overline{C}\|_F^2] = \frac{1}{t}\sum_{m=1}^n \|A_{:,m}\|_2 \cdot \|B_{m,:}\|_2 \cdot \left(\sum_{k=1}^n \|A_{:,k}\|_2 \cdot \|B_{k,:}\|_2\right)$$

$$= \frac{1}{t}\left(\sum_{m=1}^n \|A_{:,k}\|_2 \cdot \|B_{k,:}\|_2\right)^2$$

By the Cauchy-Schwarz inequality,

$$\sum_{m=1}^n \|A_{:,k}\|_2 \cdot \|B_{k,:}\|_2 \leq \sqrt{\sum_{m=1}^n \|A_{:,k}\|_2^2} \cdot \sqrt{\sum_{m=1}^n \|B_{k,:}\|_2^2} = \|A\|_F \cdot \|B\|_F$$

Overall: $\mathbb{E}[\|AB - \overline{C}\|_F^2] \leq \frac{\|A\|_F^2 \cdot \|B\|_F^2}{t}$. Setting $t = \frac{1}{\epsilon^2 \sqrt{\delta}}$, by Chebyshev's inequality:

$$\Pr[\|AB - \overline{C}\|_F \geq \epsilon \cdot \|A\|_F \cdot \|B\|_F] \leq \delta.$$

AMM Upshot

Upshot: Sampling $t = O(1/\epsilon^2)$ columns/rows of A, B with probabilities proportional to $\|A_{:,k}\|_2 \cdot \|B_{k,:}\|_2$ yields, with good probability, an approximation $\overline{\mathsf{C}}$ with

$$\|AB - \overline{\mathsf{C}}\|_F \leq \epsilon \cdot \|A\|_F \cdot \|B\|_F.$$

- Probabilities take $O(n^2)$ time to compute. After sampling, $\overline{\mathsf{C}}$ takes $O(t \cdot n^2)$ time to compute.
- Can derive related bounds when probabilities are just approximate – i.e. $p_k \geq \beta \cdot \frac{\|A_{:,k}\|_2 \cdot \|B_{k,:}\|_2}{\sum_{m=1}^{n} \|A_{:,m}\|_2 \cdot \|B_{m,:}\|_2}$ for some $\beta > 0$.
- Can also give bounds on $\|AB - \overline{\mathsf{C}}\|_2$, but analysis is much more complex. Will see tools in the coming weeks that let us do this.
- A classic example of using weighted sampling to decrease variance and in turn, sample complexity.

AMM Upshot

Think-Pair-Share 1: Ideally we would have *relative error*, $\|AB - \overline{\mathbf{C}}\|_F \leq \epsilon \|AB\|_F$. Could we get this via a tighter analysis or better sampling distribution?

Think-Pair-Share 2: What if we just uniformly sampled rows/columns? Recall that $\mathbb{E}[\|AB - \overline{\mathbf{C}}\|_F^2] = \frac{1}{t}\sum_{m=1}^{n} \frac{\|A_{:,m}\|_2^2 \cdot \|B_{m,:}\|_2^2}{p_m}$.

Stochastic Trace Estimation

Matrix Trace

The trace of a matrix $A \in \mathbb{R}^{n \times n}$ is the sum of it diagonal entries.

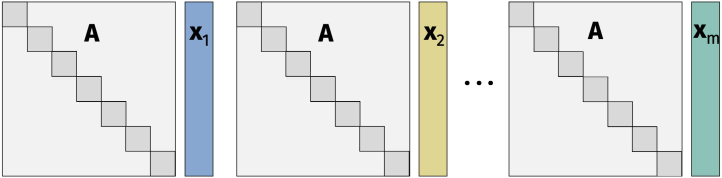
$$\text{tr}(A) = \sum_{i=1}^{n} A_{ii}.$$

When A is diagonalizable (e.g., when it is symmetric) with eigenvalues $\lambda_1, \ldots, \lambda_n$, $\text{tr}(A) = \sum_{i=1}^{n} \lambda_i$.

How many operations does it take to compute $\text{tr}(A)$ given explicit access to A?

Implicit Trace Estimation

- Given implicit access to $A \in \mathbb{R}^{n\times n}$ through **matrix-vector multiplication**.
- Goal is to approximate $\text{tr}(A) = \sum_{i=1}^{n} A_{ii}$.



Main question: How many matrix-vector multiplication "queries" $Ax_1, \ldots, Ax_m$ are required to approximate $\text{tr}(A)$?

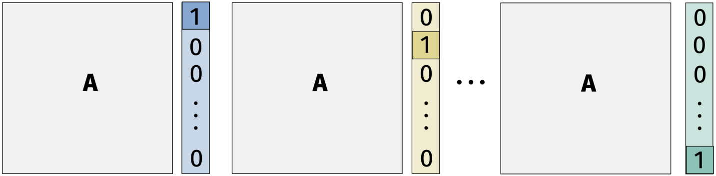
Algorithms in this model are called matrix-free methods. Useful when A is not given explicitly, but we have an efficient algorithm for multiplying A by a vector (examples to come).

What other matrix free method have we studied in this class?

Naive Exact Algorithm

Naive solution:

- Set $x_i = e_i$ for $i = 1, \ldots, n$.
- Return $\text{tr}(A) = \sum_{i=1}^{n} x_i^T A x_i$.

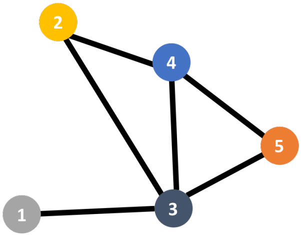


Returns exact solution, but requires n matrix-vector multiplies.

We will see how to use $m \ll n$ multiplies by using randomness and allowing for small approximation error.

Motivating Example

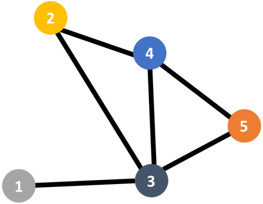
The number of triangles or other small 'motifs' is an important metric of network connectivity. E.g., important in computing the network clustering coefficient



How long does it take to exactly compute the number of triangles in the graph?

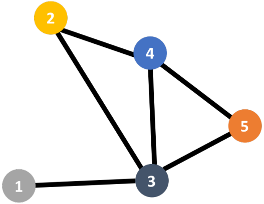
Motivating Example

Can use the adjacency matrix $B \in \{0,1\}^{n \times n}$ to write the number of triangles in a linear algebraic way.



B

0	0	1	0	0
0	0	1	1	0
1	1	0	1	1
0	1	1	0	1
0	0	1	1	0

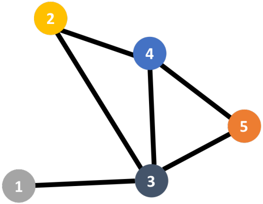


- B_{ij} indicates the number of 1-step paths (edges) from i, j
- $[B^2]_{ij}$ indicates the number of 2-step paths from i, j
- $[B^3]_{ij}$ indicates the number of 3-step paths from i, j

B_{ii} is the number of length 3-paths from i back to i. Thus,

$$\frac{1}{6}\operatorname{tr}(B^3) = \#\text{ triangles}.$$

Motivating Example



B^3

0	1	4	2	1
1	2	6	5	2
4	6	4	6	6
2	5	6	4	5
1	2	6	5	2

$$\frac{1}{6}\,\mathrm{tr}(B^3) = \#\ \text{triangles}.$$

- Explicitly forming B^3 and computing $\mathrm{tr}(B^3)$ takes $O(n^3)$ time.
- Can multiply B^3 by a vector in $3 \cdot |E| = O(n^2)$ operations.
- So a trace estimation algorithm using m queries, yields an $O(m \cdot |E|)$ time approximate triangle counting algorithm.

Other Examples

Example 2: Hessian/Jacobian matrix-vector products.

- For vector x, $\nabla \mathrm{f}(y)x$ and $\nabla^2 f(y)x$ can often be computed efficiently using finite difference methods or explicit differentiation (e.g., via backpropagation).
- Do not need to fully form $\nabla f(y)$ or $\nabla^2 f(y)$.
- Many applications, e.g., in analyzing neural network convergence.

Other Examples

Example 3: A is a function of another (explicit) matrix B, $A = f(B)$ that can be applied efficiently via an iterative method.

- Repeated multiplication to apply $A = B^3$.
- Conjugate gradient, MINRES, or any linear system solver:

$$A = B^{-1}.$$

- Lanczos method, polynomial/rational approximation:

$$A = \exp(B),\ A = \sqrt{B},\ A = \log(B),\ \text{etc.}$$

- These methods run in $n^2 \cdot C$ time, where C depends on properties of B. Typically $C \ll n$ so $n^2 \cdot C \ll n^3$.

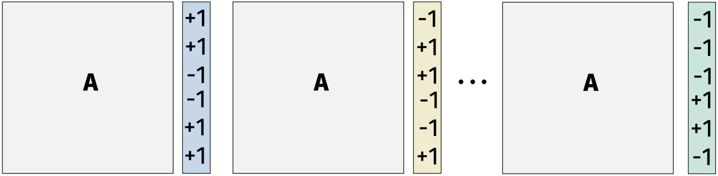
Matrix Function Examples

- Log-likelihood computation in Bayesian optimization, experimental design. $\mathrm{tr}(\log(B)) = \log\det(B)$.
- Estrada index, a measure of protein folding degree and more generally, network connectivity. $\mathrm{tr}(\exp(B))$.
- Trace inverse, which is important in uncertainty quantification and many other scientific computing applications. $\mathrm{tr}(B^{-1})$
- Information about the matrix eigenvalue spectrum, since $\mathrm{tr}(f(B)) = \sum_{i=1}^{n} f(\lambda_i)$, where λ_i is B's i^{th} eigenvalue.
- E.g., counting the number of eigenvalues in an interval, spectral density estimation, matrix norms
- See e.g., [Ubaru, and Saad 2017].

Hutchinson's Method

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 $\bar{\mathrm{T}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i^T A \mathbf{x}_i$ as an approximation to $\mathrm{tr}(A)$.



- One of the earliest examples of a randomized algorithm for linear algebraic computation.

Hutchinson's Method Error Bound

Theorem

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

$$\left|\bar{\mathsf{T}} - \mathrm{tr}(A)\right| \leq \epsilon \|A\|_F$$

If A is symmetric positive semidefinite (PSD) then

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

So for PSD A: $(1-\epsilon)\,\mathrm{tr}(A) \leq \bar{\mathsf{T}} \leq (1+\epsilon)\,\mathrm{tr}(A).$

Proof Approach

Theorem

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

$$\left|\overline{\mathsf{T}} - \mathrm{tr}(A)\right| \leq \epsilon \|A\|_F$$

1. Show that $\mathbb{E}[\overline{\mathsf{T}}] = \mathrm{tr}(A)$.
2. Bound $\mathrm{Var}[\overline{\mathsf{T}}]$.
3. Apply Chebyshev's inequality.

A tighter proof that uses the Hanson-Wright inequality, an exponential concentration inequality for quadratic forms, can improve the δ dependence to $\log(1/\delta)$.

Expectation Analysis

Hutchinson's Estimator::

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

By linearity of expectation, $\mathbb{E}[\bar{\mathsf{T}}] = \mathbb{E}[\mathbf{x}^T A \mathbf{x}]$ for a single random ± 1 vector $\mathbf{x}$.

$$\mathbb{E}[\mathbf{x}^T A \mathbf{x}] = \mathbb{E}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{x}_i \mathbf{x}_j A_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} \cdot \mathbb{E}[\mathbf{x}_i \mathbf{x}_j] = \sum_{i=1}^{n} A_{ii}.$$

- When $i \neq j$, $\mathbf{x}_i \mathbf{x}_j = 1$ with probability $1/2$ and -1 with probability $1/2$, so $\mathbb{E}[\mathbf{x}_i \mathbf{x}_j] = 0$. When $i = j$, $\mathbf{x}_i \mathbf{x}_j = 1$, so $\mathbb{E}[\mathbf{x}_i \mathbf{x}_j] = 1$.
- So the estimator is correct in expectation: $\mathbb{E}[\bar{\mathsf{T}}] = \mathrm{tr}(A)$.

Variance Bound

Hutchinson's Estimator::

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

$$\text{Var}[\bar{\mathsf{T}}] = \frac{1}{m}\text{Var}[\mathbf{x}^T A \mathbf{x}] = \frac{1}{m}\text{Var}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{x}_i \mathbf{x}_j A_{ij}\right]$$

Can we apply linearity of variance here? Almost – need to remove repeated terms, and then can use pairwise independence.

$$\text{Var}[\bar{\mathsf{T}}] = \frac{1}{m}\text{Var}\left[\sum_{i=1}^{n} A_{ii} + \sum_{i=1}^{n}\sum_{j>i} \mathbf{x}_i \mathbf{x}_j (A_{ij} + A_{ji})\right]$$

$$= \frac{1}{m}\sum_{i=1}^{n}\sum_{j>i}\text{Var}[\mathbf{x}_i \mathbf{x}_j] \cdot (A_{ij} + A_{ji})^2 \leq \frac{1}{m}\sum_{i=1}^{n}\sum_{j>i} 2A_{ij}^2 + 2A_{ji}^2 \leq \frac{2\|A\|_F^2}{m}.$$

Final Analysis

Hutchinson's Estimator::

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

Chebyshev's inequality implies that, for $m = \frac{2}{\delta\epsilon^2}$:

$$\Pr\left[\left|\bar{\mathsf{T}} - \operatorname{tr}(A)\right| \geq \epsilon\|A\|_F\right] \leq \frac{2\|A\|_F^2/m}{\epsilon^2\|A\|_F^2} = \delta.$$

Could we have gotten a better bound by applying Bernstein's inequality to $\sum_{i=1}^{n}\sum_{j>i} \mathbf{x}_i\mathbf{x}_j(A_{ij} + A_{ji})$?

Hanson-Wright is an exponential concentration bound that can be used in the specific case – improves bound to $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$.

Optimality of Hutchinson's Method

The $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ bound given by the Hanson-Wright inequality is tight.

- Any algorithm that only uses queries of the form $\mathbf{x}_i^T A \mathbf{x}_i$ requires $\Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ samples to estimate $\text{tr}(A)$ to error $\pm\epsilon\,\text{tr}(A)$ for PSD A [Wimmer, Wu, Zhang 2014].
- We recently showed that using the full power of matrix-vector queries, one can achieve $O\left(\frac{\log(1/\delta)}{\epsilon}\right)$ queries for PSD matrices – see project topics.