Hutch++

Optimal Stochastic Trace Estimation

Raphael A. Meyer (New York University)

With Christopher Musco (New York University), Cameron Musco (University of Massachusetts Amherst), and David P. Woodruff (Carnegie Mellon University)

Overview

1. Introduction
   - What problems am I solving?
   - Why are these problems interesting?
   - How am I solving them?
2. Trace Estimation *(SOSA 2021)*
3. Trace Monomial Estimation *(Ongoing Research)*

Numerical Linear Algebra

- Scientific Computing relies on Numerical Linear Algebra
- We spent decades building better algorithms

Numerical Linear Algebra

- Scientific Computing relies on Numerical Linear Algebra
- We spent decades building better algorithms
- We don't know which algorithms are *optimal*
  - Krylov Iteration is optimal for top eigenvalue
  - Hutchinson's Estimator is suboptimal for trace estimation

Numerical Linear Algebra

- Scientific Computing relies on Numerical Linear Algebra
- We spent decades building better algorithms
- We don't know which algorithms are *optimal*
  - Krylov Iteration is optimal for top eigenvalue
  - Hutchinson's Estimator is suboptimal for trace estimation
- My goal: Prove the optimality of linear algebra algorithms
  - Emphasis on building lower bounds

Trace Estimation

- Goal: Estimate trace of $d \times d$ matrix $\boldsymbol{A}$:

$$\mathrm{tr}(\boldsymbol{A}) = \sum_{i=1}^{d} \boldsymbol{A}_{ii} = \sum_{i=1}^{d} \lambda_i$$

Trace Estimation

- Goal: Estimate trace of $d \times d$ matrix $\boldsymbol{A}$:

$$\mathrm{tr}(\boldsymbol{A}) = \sum_{i=1}^{d} \boldsymbol{A}_{ii} = \sum_{i=1}^{d} \lambda_i$$

- In Downstream Applications, $\boldsymbol{A}$ is not stored in memory.
- Instead, $\boldsymbol{B}$ is in memory and $\boldsymbol{A} = f(\boldsymbol{B})$:

No. Triangles	Estrada Index	Log-Determinant
$\mathrm{tr}(\frac{1}{6}\boldsymbol{B}^3)$	$\mathrm{tr}(e^{\boldsymbol{B}})$	$\mathrm{tr}(\ln(\boldsymbol{B}))$

Trace Estimation

- Goal: Estimate trace of $d \times d$ matrix $\boldsymbol{A}$:

$$\text{tr}(\boldsymbol{A}) = \sum_{i=1}^{d} \boldsymbol{A}_{ii} = \sum_{i=1}^{d} \lambda_i$$

- In Downstream Applications, $\boldsymbol{A}$ is not stored in memory.
- Instead, $\boldsymbol{B}$ is in memory and $\boldsymbol{A} = f(\boldsymbol{B})$:

No. Triangles	Estrada Index	Log-Determinant
$\text{tr}(\frac{1}{6}\boldsymbol{B}^3)$	$\text{tr}(e^{\boldsymbol{B}})$	$\text{tr}(\ln(\boldsymbol{B}))$

- Computing $\boldsymbol{A} = \frac{1}{6}\boldsymbol{B}^3$ takes $O(n^3)$ time
- Computing $\boldsymbol{A}\mathrm{x} = \frac{1}{6}\boldsymbol{B}(\boldsymbol{B}(\boldsymbol{B}\mathrm{x}))$ takes $O(n^2)$ time
- If $\boldsymbol{A} = f(\boldsymbol{B})$, then we can often compute $\boldsymbol{A}\mathrm{x}$ quickly

Trace Estimation

- Goal: Estimate trace of $d \times d$ matrix $\boldsymbol{A}$:

$$\mathrm{tr}(\boldsymbol{A}) = \sum_{i=1}^{d} \boldsymbol{A}_{ii} = \sum_{i=1}^{d} \lambda_i$$

- In Downstream Applications, $\boldsymbol{A}$ is not stored in memory.
- Instead, $\boldsymbol{B}$ is in memory and $\boldsymbol{A} = f(\boldsymbol{B})$:

No. Triangles	Estrada Index	Log-Determinant
$\mathrm{tr}(\frac{1}{6}\boldsymbol{B}^3)$	$\mathrm{tr}(e^{\boldsymbol{B}})$	$\mathrm{tr}(\ln(\boldsymbol{B}))$

- Computing $\boldsymbol{A} = \frac{1}{6}\boldsymbol{B}^3$ takes $O(n^3)$ time
- Computing $\boldsymbol{A}\mathbf{x} = \frac{1}{6}\boldsymbol{B}(\boldsymbol{B}(\boldsymbol{B}\mathbf{x}))$ takes $O(n^2)$ time
- If $\boldsymbol{A} = f(\boldsymbol{B})$, then we can often compute $\boldsymbol{A}\mathbf{x}$ quickly
- Goal: Estimate $\mathrm{tr}(\boldsymbol{A})$ by computing $\boldsymbol{A}\mathbf{x}_1, \ldots \boldsymbol{A}\mathbf{x}_k$

Matrix-Vector Oracle Model

Formally: Matrix-Vector Product as a Computational Primitive

Matrix-Vector Oracle Model

Formally: Matrix-Vector Product as a Computational Primitive

- Given access to a $d \times d$ matrix $\boldsymbol{A}$ only through a Matrix-Vector Multiplication Oracle

$$\mathbf{x} \quad \xrightarrow{\textit{input}} \quad \text{ORACLE} \quad \xrightarrow{\textit{output}} \quad \boldsymbol{A}\mathbf{x}$$

- e.g. Krylov Methods, Sketching, Streaming, ...
- Very few existing lower bounds

Matrix-Vector Oracle Model

Formally: Matrix-Vector Product as a Computational Primitive

- Given access to a $d \times d$ matrix $\boldsymbol{A}$ only through a Matrix-Vector Multiplication Oracle

$$\mathbf{x} \xrightarrow{\textit{input}} \text{ORACLE} \xrightarrow{\textit{output}} \boldsymbol{A}\mathbf{x}$$

- e.g. Krylov Methods, Sketching, Streaming, ...
- Very few existing lower bounds

Trace Estimation: Estimate $\text{tr}(\boldsymbol{A})$ with as few Matrix-Vector products $\boldsymbol{A}\mathbf{x}_1, \ldots, \boldsymbol{A}\mathbf{x}_k$ as possible.

$$|\tilde{\text{tr}}(\boldsymbol{A}) - \text{tr}(\boldsymbol{A})| \leq \varepsilon \, \text{tr}(\boldsymbol{A})$$

Our Contributions

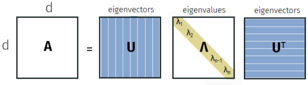
Prior Work:

- Hutchinson's Estimator: $O(\frac{1}{\varepsilon^2})$ products suffice [AT11]
  - 2 Lines of MATLAB code
- Lower Bound: Hutchinson's Estimator needs $\Omega(\frac{1}{\varepsilon^2})$ products [WWZ14]

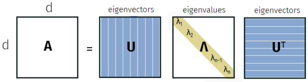
Our Results:

- Hutch++ Estimator: $O(\frac{1}{\varepsilon})$ products suffice
  - 5 Lines of MATLAB code
- Lower Bound: Any estimator needs $\Omega(\frac{1}{\varepsilon})$ products

Linear Algebra Review

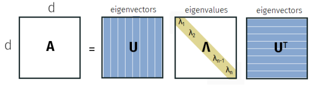


- Symmetric $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ has $\boldsymbol{A} = \boldsymbol{U} \Lambda \boldsymbol{U}^\top$
- $\boldsymbol{U}$ is a rotation matrix: $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$
- Eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$



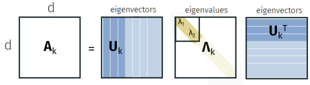
- Symmetric $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ has $\boldsymbol{A} = \boldsymbol{U} \Lambda \boldsymbol{U}^\top$
- $\boldsymbol{U}$ is a rotation matrix: $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$
- Eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
- $\|\boldsymbol{A}\|_F^2 = \sum_{i,j} \boldsymbol{A}_{i,j}^2 = \sum_i \lambda_i^2$
- $\text{tr}(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{i,i} = \sum_i \lambda_i$

Linear Algebra Review



- Symmetric $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ has $\boldsymbol{A} = \boldsymbol{U} \Lambda \boldsymbol{U}^\top$
- $\boldsymbol{U}$ is a rotation matrix: $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$
- Eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
- $\|\boldsymbol{A}\|_F^2 = \sum_{i,j} \boldsymbol{A}_{i,j}^2 = \sum_i \lambda_i^2$
- $\mathrm{tr}(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{i,i} = \sum_i \lambda_i$
- Positive Semi-Definite (PSD) $\boldsymbol{A}$ has $\lambda_i \geq 0$ for all i
  - $\|\boldsymbol{A}\|_F = \|\boldsymbol{\lambda}\|_2 \leq \|\boldsymbol{\lambda}\|_1 = \mathrm{tr}(\boldsymbol{A})$

Linear Algebra Review



- Symmetric $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ has $\boldsymbol{A} = \boldsymbol{U} \Lambda \boldsymbol{U}^\intercal$
- $\boldsymbol{U}$ is a rotation matrix: $\boldsymbol{U}^\intercal \boldsymbol{U} = \boldsymbol{I}$
- Eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
- $\|\boldsymbol{A}\|_F^2 = \sum_{i,j} \boldsymbol{A}_{i,j}^2 = \sum_i \lambda_i^2$
- $\mathrm{tr}(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{i,i} = \sum_i \lambda_i$
- Positive Semi-Definite (PSD) $\boldsymbol{A}$ has $\lambda_i \geq 0$ for all i
  - $\|\boldsymbol{A}\|_F = \|\boldsymbol{\lambda}\|_2 \leq \|\boldsymbol{\lambda}\|_1 = \mathrm{tr}(\boldsymbol{A})$
- Low Rank Approximation:

  $\boldsymbol{A}_k = \boldsymbol{U}_k \Lambda_k \boldsymbol{U}_k^\intercal = \mathrm{argmin}_{rank(\boldsymbol{B})=k} \|\boldsymbol{A} - \boldsymbol{B}\|_F$

Probability Review

- If $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, then $\boldsymbol{A}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{A}\boldsymbol{A}^\intercal)$
- If $X_1, \ldots, X_n \sim \mathcal{N}(0, 1)$, then $S := \sum_i X_i^2 \sim \chi_n^2$, $\mathbb{E}[S] = n$, $\mathrm{Var}[S] = 2n$

Probability Review

- If $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, then $\boldsymbol{A}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{A}\boldsymbol{A}^\top)$
- If $X_1, \ldots, X_n \sim \mathcal{N}(0, 1)$, then $S := \sum_i X_i^2 \sim \chi_n^2$, $\mathbb{E}[S] = n$, $\mathrm{Var}[S] = 2n$
- Chebyshev's Ineq: $|X - \mathbb{E}[X]| \leq \frac{1}{\sqrt{\delta}}\sqrt{\mathrm{Var}[X]}$ w.p. $\geq 1 - \delta$

Probability Review

- If $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, then $\boldsymbol{A}\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{A}\boldsymbol{A}^\top)$
- If $X_1, \ldots, X_n \sim \mathcal{N}(0, 1)$, then $S := \sum_i X_i^2 \sim \chi_n^2$, $\mathbb{E}[S] = n$, $\mathrm{Var}[S] = 2n$
- Chebyshev's Ineq: $|X - \mathbb{E}[X]| \leq \frac{1}{\sqrt{\delta}}\sqrt{\mathrm{Var}[X]}$ w.p. $\geq 1 - \delta$
- Chebyshev's Ineq: $|X - \mathbb{E}[X]| \leq O(\sqrt{\mathrm{Var}[X]})$ w.p. $\geq \frac{2}{3}$

Towards Optimal

Trace Estimation in the

Matrix-Vector Oracle Model

Hutchinson's Estimator

- If $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{I})$, then

$$\mathbb{E}[\mathbf{x}^\top \boldsymbol{A} \mathbf{x}] = \mathrm{tr}(\boldsymbol{A}) \qquad \mathrm{Var}[\mathbf{x}^\top \boldsymbol{A} \mathbf{x}] = 2\|\boldsymbol{A}\|_F^2$$

Hutchinson's Estimator

- If $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{I})$, then

$$\mathbb{E}[\mathbf{x}^\intercal \boldsymbol{A} \mathbf{x}] = \mathrm{tr}(\boldsymbol{A}) \qquad \mathrm{Var}[\mathbf{x}^\intercal \boldsymbol{A} \mathbf{x}] = 2\|\boldsymbol{A}\|_F^2$$

- Hutchinson's Estimator: $\mathrm{H}_\ell(\boldsymbol{A}) := \frac{1}{\ell}\sum_{i=1}^{\ell} \mathbf{x}_i^\intercal \boldsymbol{A} \mathbf{x}_i$

$$\mathbb{E}[\mathrm{H}_\ell(\boldsymbol{A})] = \mathrm{tr}(\boldsymbol{A}) \qquad \mathrm{Var}[\mathrm{H}_\ell(\boldsymbol{A})] = \frac{2}{\ell}\|\boldsymbol{A}\|_F^2$$

Hutchinson's Estimator

- If $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{I})$, then

$$\mathbb{E}[\mathbf{x}^\top \boldsymbol{A} \mathbf{x}] = \mathrm{tr}(\boldsymbol{A}) \qquad \mathrm{Var}[\mathbf{x}^\top \boldsymbol{A} \mathbf{x}] = 2\|\boldsymbol{A}\|_F^2$$

- Hutchinson's Estimator: $\mathrm{H}_\ell(\boldsymbol{A}) := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i^\top \boldsymbol{A} \mathbf{x}_i$

$$\mathbb{E}[\mathrm{H}_\ell(\boldsymbol{A})] = \mathrm{tr}(\boldsymbol{A}) \qquad \mathrm{Var}[\mathrm{H}_\ell(\boldsymbol{A})] = \frac{2}{\ell}\|\boldsymbol{A}\|_F^2$$

Proof: $\mathrm{H}_\ell(\boldsymbol{A})$ needs $\ell = O(\frac{1}{\varepsilon^2})$ for PSD $\boldsymbol{A}$

- For PSD $\boldsymbol{A}$, we have $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(\boldsymbol{A})$, so that

Hutchinson's Estimator

- If $\mathbf{x} \sim \mathcal{N}(0, \boldsymbol{I})$, then

$$\mathbb{E}[\mathbf{x}^\top \boldsymbol{A} \mathbf{x}] = \operatorname{tr}(\boldsymbol{A}) \qquad \operatorname{Var}[\mathbf{x}^\top \boldsymbol{A} \mathbf{x}] = 2\|\boldsymbol{A}\|_F^2$$

- Hutchinson's Estimator: $\mathrm{H}_\ell(\boldsymbol{A}) := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i^\top \boldsymbol{A} \mathbf{x}_i$

$$\mathbb{E}[\mathrm{H}_\ell(\boldsymbol{A})] = \operatorname{tr}(\boldsymbol{A}) \qquad \operatorname{Var}[\mathrm{H}_\ell(\boldsymbol{A})] = \frac{2}{\ell}\|\boldsymbol{A}\|_F^2$$

Proof: $\mathrm{H}_\ell(\boldsymbol{A})$ needs $\ell = O(\frac{1}{\varepsilon^2})$ for PSD $\boldsymbol{A}$

- For PSD $\boldsymbol{A}$, we have $\|\boldsymbol{A}\|_F \le \operatorname{tr}(\boldsymbol{A})$, so that

$$\begin{aligned} |\mathrm{H}_\ell(\boldsymbol{A}) - \operatorname{tr}(\boldsymbol{A})| &\le O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F && \text{(Chebyshev Ineq.)} \\ &\le O(\tfrac{1}{\sqrt{\ell}}) \operatorname{tr}(\boldsymbol{A}) && (\|\boldsymbol{A}\|_F \le \operatorname{tr}(\boldsymbol{A})) \\ &= \varepsilon \operatorname{tr}(\boldsymbol{A}) && (\ell = O(\tfrac{1}{\varepsilon^2})) \end{aligned}$$

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

$$\begin{aligned}
|\mathrm{H}_\ell(\boldsymbol{A}) - \operatorname{tr}(\boldsymbol{A})| &\leq O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\
&\leq O(\tfrac{1}{\sqrt{\ell}}) \operatorname{tr}(\boldsymbol{A}) \\
&= \varepsilon \operatorname{tr}(\boldsymbol{A})
\end{aligned}$$

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

$$\begin{aligned} |\mathrm{H}_\ell(\boldsymbol{A}) - \operatorname{tr}(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\ &\leq O(\tfrac{1}{\sqrt{\ell}})\operatorname{tr}(\boldsymbol{A}) \\ &= \varepsilon \operatorname{tr}(\boldsymbol{A}) \end{aligned}$$

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

$$
\begin{aligned}
|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\
&\leq O(\tfrac{1}{\sqrt{\ell}})\,\mathrm{tr}(\boldsymbol{A}) \\
&= \varepsilon\,\mathrm{tr}(\boldsymbol{A})
\end{aligned}
$$

- When is the bound $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(A)$ tight?

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

$$\begin{aligned}|\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\ &\leq O(\tfrac{1}{\sqrt{\ell}})\,\mathrm{tr}(\boldsymbol{A}) \\ &= \varepsilon\,\mathrm{tr}(\boldsymbol{A})\end{aligned}$$

- When is the bound $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(A)$ tight?
- Let $\mathbf{v} = \begin{bmatrix}\lambda_1 & \dots & \lambda_n\end{bmatrix}$ be the eigenvalues of PSD $\boldsymbol{A}$

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

$$\begin{aligned} |\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\ &\leq O(\tfrac{1}{\sqrt{\ell}})\,\mathrm{tr}(\boldsymbol{A}) \\ &= \varepsilon\,\mathrm{tr}(\boldsymbol{A}) \end{aligned}$$

- When is the bound $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(A)$ tight?
- Let $\mathbf{v} = \begin{bmatrix} \lambda_1 & \dots & \lambda_n \end{bmatrix}$ be the eigenvalues of PSD $\boldsymbol{A}$
- When is the bound $\|\mathbf{v}\|_2 \leq \|\mathbf{v}\|_1$ tight?

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

$$\begin{aligned} |\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\ &\leq O(\tfrac{1}{\sqrt{\ell}})\,\mathrm{tr}(\boldsymbol{A}) \\ &= \varepsilon\,\mathrm{tr}(\boldsymbol{A}) \end{aligned}$$

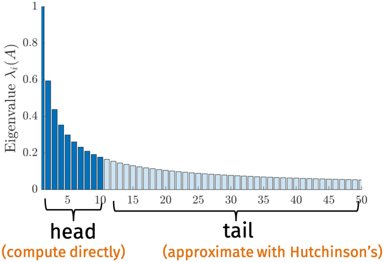
- When is the bound $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(A)$ tight?
- Let $\mathbf{v} = \begin{bmatrix} \lambda_1 & \dots & \lambda_n \end{bmatrix}$ be the eigenvalues of PSD $\boldsymbol{A}$
- When is the bound $\|\mathbf{v}\|_2 \leq \|\mathbf{v}\|_1$ tight?
  - Property of norms: $\|\mathbf{v}\|_2 \approx \|\mathbf{v}\|_1$ only if $\mathbf{v}$ is nearly sparse

Hutchinson's Estimator

For what $\boldsymbol{A}$ is this analysis tight?

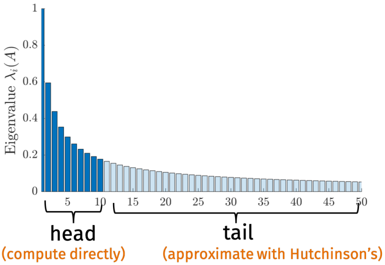
$$\begin{aligned} |\mathrm{H}_\ell(\boldsymbol{A}) - \mathrm{tr}(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F \\ &\leq O(\tfrac{1}{\sqrt{\ell}})\,\mathrm{tr}(\boldsymbol{A}) \\ &= \varepsilon\,\mathrm{tr}(\boldsymbol{A}) \end{aligned}$$

- When is the bound $\|\boldsymbol{A}\|_F \leq \mathrm{tr}(A)$ tight?
- Let $\mathbf{v} = \begin{bmatrix} \lambda_1 & \ldots & \lambda_n \end{bmatrix}$ be the eigenvalues of PSD $\boldsymbol{A}$
- When is the bound $\|\mathbf{v}\|_2 \leq \|\mathbf{v}\|_1$ tight?
  - Property of norms: $\|\mathbf{v}\|_2 \approx \|\mathbf{v}\|_1$ only if $\mathbf{v}$ is nearly sparse
- Hutchinson only requires $O(\frac{1}{\varepsilon^2})$ queries if $\boldsymbol{A}$ has a few large eigenvalues



Idea: Explicitly estimate the top few eigenvalues of $\boldsymbol{A}$. Use Hutchinson's for the rest.

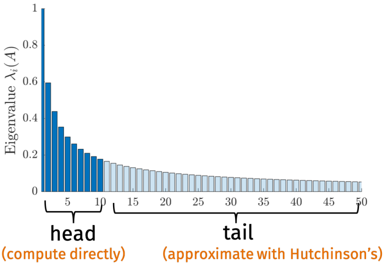
Helping Hutchinson's Estimator



Idea: Explicitly estimate the top few eigenvalues of $\boldsymbol{A}$. Use Hutchinson's for the rest.

1. Find a good rank-k approximation $\tilde{\boldsymbol{A}}_k$
2. Notice that $\text{tr}(\boldsymbol{A}) = \text{tr}(\tilde{\boldsymbol{A}}_k) + \text{tr}(\boldsymbol{A} - \tilde{\boldsymbol{A}}_k)$
3. Compute $\text{tr}(\tilde{\boldsymbol{A}}_k)$ exactly
4. Return $\text{Hutch++}(\boldsymbol{A}) = \text{tr}(\tilde{\boldsymbol{A}}_k) + \text{H}_\ell(\boldsymbol{A} - \tilde{\boldsymbol{A}}_k)$

Helping Hutchinson's Estimator



Idea: Explicitly estimate the top few eigenvalues of $\boldsymbol{A}$. Use Hutchinson's for the rest.

1. Find a good rank-k approximation $\tilde{\boldsymbol{A}}_k$
2. Notice that $\text{tr}(\boldsymbol{A}) = \text{tr}(\tilde{\boldsymbol{A}}_k) + \text{tr}(\boldsymbol{A} - \tilde{\boldsymbol{A}}_k)$
3. Compute $\text{tr}(\tilde{\boldsymbol{A}}_k)$ exactly
4. Return $\text{Hutch++}(\boldsymbol{A}) = \text{tr}(\tilde{\boldsymbol{A}}_k) + \text{H}_\ell(\boldsymbol{A} - \tilde{\boldsymbol{A}}_k)$

If $k = \ell = O(\frac{1}{\varepsilon})$, then $|\text{Hutch++}(\boldsymbol{A}) - \text{tr}(\boldsymbol{A})| \leq \varepsilon \, \text{tr}(\boldsymbol{A})$.

(Whiteboard)

Finding a Good Low-Rank Approximation

Let $\boldsymbol{A}_k$ be the best rank-k approximation of $\boldsymbol{A}$.

Lemma [Sar06, Woo14]

Let $\boldsymbol{S} \in \mathbb{R}^{d \times k}$ have i.i.d. uniform ± 1 entries, $\boldsymbol{Q} = \text{orth}(\boldsymbol{AS})$, and $\tilde{\boldsymbol{A}}_k = \boldsymbol{AQQ}^\intercal$. Then, with probability $1 - \delta$,

$$\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_k\|_F \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$$

so long as $\boldsymbol{S}$ has $m = O(k + \log(1/\delta))$ columns.

Finding a Good Low-Rank Approximation

Let $\boldsymbol{A}_k$ be the best rank-k approximation of $\boldsymbol{A}$.

Lemma [Sar06, Woo14]

Let $\boldsymbol{S} \in \mathbb{R}^{d \times k}$ have i.i.d. uniform ± 1 entries, $\boldsymbol{Q} = \text{orth}(\boldsymbol{AS})$, and $\tilde{\boldsymbol{A}}_k = \boldsymbol{AQQ}^\intercal$. Then, with probability $1 - \delta$,

$$\|\boldsymbol{A} - \tilde{\boldsymbol{A}}_k\|_F \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F$$

so long as $\boldsymbol{S}$ has $m = O(k + \log(1/\delta))$ columns.

We can compute the trace of $\tilde{\boldsymbol{A}}_k$ with m queries and $O(mn)$ space:

$$\text{tr}(\tilde{\boldsymbol{A}}_k) = \text{tr}(\boldsymbol{AQQ}^\intercal) = \text{tr}(\boldsymbol{Q}^\intercal(\boldsymbol{AQ}))$$

Hutch++ Algorithm:

- Input: Number of matrix-vector queries m, matrix $\boldsymbol{A}$

1. Sample $\boldsymbol{S} \in \mathbb{R}^{d \times \frac{m}{3}}$ and $\boldsymbol{G} \in \mathbb{R}^{d \times \frac{m}{3}}$ with i.i.d. $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ entries
2. Compute $\boldsymbol{Q} = \text{qr}(\boldsymbol{AS})$
3. Return $\text{tr}(\boldsymbol{Q}^T\boldsymbol{AQ}) + \frac{3}{m}\text{tr}(\boldsymbol{G}^T(\boldsymbol{I} - \boldsymbol{QQ}^\mathsf{T})\boldsymbol{A}(\boldsymbol{I} - \boldsymbol{QQ}^\mathsf{T})\boldsymbol{G})$

Hutch++

Hutch++ Algorithm:

- Input: Number of matrix-vector queries m, matrix $\boldsymbol{A}$

1. Sample $\boldsymbol{S} \in \mathbb{R}^{d \times \frac{m}{3}}$ and $\boldsymbol{G} \in \mathbb{R}^{d \times \frac{m}{3}}$ with i.i.d. $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ entries
2. Compute $\boldsymbol{Q} = \mathrm{qr}(\boldsymbol{A}\boldsymbol{S})$
3. Return $\mathrm{tr}(\boldsymbol{Q}^T \boldsymbol{A} \boldsymbol{Q}) + \frac{3}{m} \mathrm{tr}(\boldsymbol{G}^T (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\intercal) \boldsymbol{A} (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\intercal) \boldsymbol{G})$

This algorithm is adaptive:

Hutch++

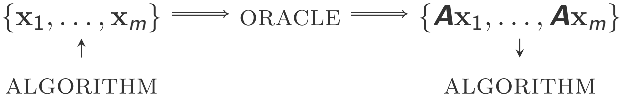
Hutch++ Algorithm:

- Input: Number of matrix-vector queries m, matrix $\boldsymbol{A}$

1. Sample $\boldsymbol{S} \in \mathbb{R}^{d \times \frac{m}{3}}$ and $\boldsymbol{G} \in \mathbb{R}^{d \times \frac{m}{3}}$ with i.i.d. $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ entries
2. Compute $\boldsymbol{Q} = \mathrm{qr}(\boldsymbol{AS})$
3. Return $\mathrm{tr}(\boldsymbol{Q}^T \boldsymbol{A} \boldsymbol{Q}) + \frac{3}{m} \mathrm{tr}(\boldsymbol{G}^T (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\top) \boldsymbol{A} (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\top) \boldsymbol{G})$

This algorithm is adaptive:

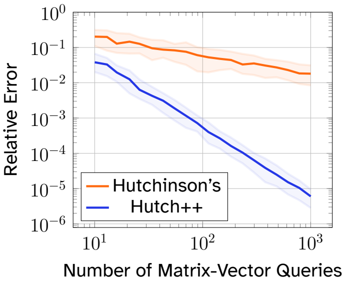
There is a non-adaptive variant of Hutch++:



When $\|\boldsymbol{A}\|_F \approx \text{tr}(\boldsymbol{A})$, Hutch++ is much faster than H_ℓ:

Fast Eig. Decay

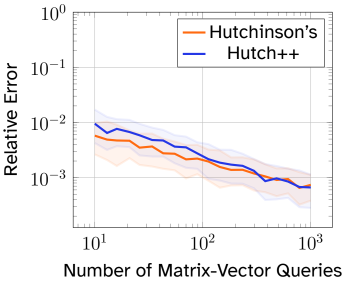
Decay Plot.pdf Decay Plot.bb



(a) $\|\boldsymbol{A}\|_F = 0.63\,\text{tr}(\boldsymbol{A})$

Slow Eig. Decay

Decay Rate.pdf Decay Rate.bb



(b) $\|\boldsymbol{A}\|_F = 0.02\,\text{tr}(\boldsymbol{A})$

```
function T = hutchplusplus(A, m)
    S = 2*randi(2,size(A,1),m/3);
    G = 2*randi(2,size(A,1),m/3);
    [Q,~] = qr(A*S,0);
    G = G - Q*(Q'*G);
    T = trace(Q'*A*Q) + 1/size(G,2)*trace(G'*A*G);
end
```

Trace Estimation Lower Bounds

Super Rough Intuition

$$\mathbf{x} \xrightarrow[]{\textit{input}} \text{ORACLE} \xrightarrow[]{\textit{output}} \boldsymbol{A}\mathbf{x}$$

View oracle as a limit on information about $\boldsymbol{A}$:

1. Suppose $\boldsymbol{A} \sim \mathcal{D}$ is a random matrix
2. Then $\mathrm{tr}(\boldsymbol{A})$ is a random variable with variance
3. If an algorithm computes few queries, it has little information about $\mathrm{tr}(\boldsymbol{A})$
4. Then the algorithm cannot predict $\mathrm{tr}(\boldsymbol{A})$ well

Super Rough Intuition

$$\mathbf{x} \xrightarrow{\textit{input}} \text{ORACLE} \xrightarrow{\textit{output}} \boldsymbol{A}\mathbf{x}$$

View oracle as a limit on information about $\boldsymbol{A}$:

1. Suppose $\boldsymbol{A} \sim \mathcal{D}$ is a random matrix
2. Then $\text{tr}(\boldsymbol{A})$ is a random variable with variance
3. If an algorithm computes few queries, it has little information about $\text{tr}(\boldsymbol{A})$
4. Then the algorithm cannot predict $\text{tr}(\boldsymbol{A})$ well

Removing the Algorithm's Agency

- **Problem:** The user can pick many different query vectors $\mathbf{x}$.
- If the user had no freedom, we could use **statistics** to make lower bounds.

Removing the Algorithm's Agency

- **Problem:** The user can pick many different query vectors $\mathbf{x}$.
- If the user had no freedom, we could use **statistics** to make lower bounds.

Two Observations:

1. WLOG, the user submits orthonormal query vectors

Removing the Algorithm's Agency

- **Problem:** The user can pick many different query vectors $\mathbf{x}$.
- If the user had no freedom, we could use **statistics** to make lower bounds.

Two Observations:

1. WLOG, the user submits orthonormal query vectors
2. Let $\boldsymbol{G}$ be a $\mathcal{N}(0,1)$ Gaussian matrix
   Let $\boldsymbol{Q}$ be an orthogonal matrix
   Then $\boldsymbol{GQ}$ is a $\mathcal{N}(0,1)$ Gaussian matrix
   - (informal) If $\boldsymbol{A}$ uses Gaussians, the responses from the oracle are independent of the queries submitted.

Removing the Algorithm's Agency

- **Problem:** The user can pick many different query vectors $\mathbf{x}$.
- If the user had no freedom, we could use **statistics** to make lower bounds.

Two Observations:

1. WLOG, the user submits orthonormal query vectors
2. Let $\boldsymbol{G}$ be a $\mathcal{N}(0,1)$ Gaussian matrix
   Let $\boldsymbol{Q}$ be an orthogonal matrix
   Then $\boldsymbol{GQ}$ is a $\mathcal{N}(0,1)$ Gaussian matrix
   - (informal) If $\boldsymbol{A}$ uses Gaussians, the responses from the oracle are independent of the queries submitted.

- (informal) WLOG, the user observes the first k columns of $\boldsymbol{A}$.

Wigner/Wishart Anti-Concentration Method

Theorem (Wishart Case)

- Let $\boldsymbol{G} \in \mathbb{R}^{d \times d}$ be a $\mathcal{N}(0,1)$ Gaussian Matrix.
- Let $\boldsymbol{A} = \boldsymbol{G}^\intercal \boldsymbol{G}$ be a *Wishart Matrix*.
- An algorithm sends query vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$, gets responses $\mathbf{w}_1, \ldots, \mathbf{w}_k$

Wigner/Wishart Anti-Concentration Method

Theorem (Wishart Case)

- Let $\boldsymbol{G} \in \mathbb{R}^{d \times d}$ be a $\mathcal{N}(0,1)$ Gaussian Matrix.
- Let $\boldsymbol{A} = \boldsymbol{G}^\intercal \boldsymbol{G}$ be a *Wishart Matrix*.
- An algorithm sends query vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$, gets responses $\mathbf{w}_1, \ldots, \mathbf{w}_k$
- Then there exists orthogonal matrix $\boldsymbol{V}$ such that

$$\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal = \boldsymbol{\Delta} + \begin{bmatrix} 0 & 0 \\ 0 & \tilde{\boldsymbol{A}} \end{bmatrix}$$

  where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{(d-k)\times(d-k)}$ is distributed as $\tilde{A} = \tilde{\boldsymbol{G}}^\intercal \tilde{\boldsymbol{G}}$, conditioned on all observations $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_k$
- $\boldsymbol{\Delta}$ is known exactly

Wigner/Wishart Anti-Concentration Method

Theorem (Wishart Case)

- Let $\boldsymbol{G} \in \mathbb{R}^{d \times d}$ be a $\mathcal{N}(0,1)$ Gaussian Matrix.
- Let $\boldsymbol{A} = \boldsymbol{G}^\intercal \boldsymbol{G}$ be a *Wishart Matrix*.
- An algorithm sends query vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$, gets responses $\mathbf{w}_1, \ldots, \mathbf{w}_k$
- Then there exists orthogonal matrix $\boldsymbol{V}$ such that

$$\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal = \boldsymbol{\Delta} + \begin{bmatrix} 0 & 0 \\ 0 & \tilde{\boldsymbol{A}} \end{bmatrix}$$

  where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{(d-k)\times(d-k)}$ is distributed as $\tilde{A} = \tilde{\boldsymbol{G}}^\intercal \tilde{\boldsymbol{G}}$, conditioned on all observations $\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{w}_1, \ldots, \mathbf{w}_k$
- $\boldsymbol{\Delta}$ is known exactly

- Analogous holds for Wigner Matrices: $\boldsymbol{A} = \frac{1}{2}(\boldsymbol{G} + \boldsymbol{G}^\intercal)$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\text{tr}(\boldsymbol{A}) = \text{tr}(\boldsymbol{VAV}^\mathsf{T}) = \text{tr}(\boldsymbol{\Delta}) + \text{tr}(\tilde{\boldsymbol{A}})$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\text{tr}(\boldsymbol{A}) = \text{tr}(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^{\intercal}) = \text{tr}(\boldsymbol{\Delta}) + \text{tr}(\tilde{\boldsymbol{A}})$
2. Let t estimate $\text{tr}(\boldsymbol{A})$. Define $\tilde{t} := t - \text{tr}(\boldsymbol{\Delta})$.

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{VAV}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
2. Let t estimate $\mathrm{tr}(\boldsymbol{A})$. Define $\tilde{t} := t - \mathrm{tr}(\boldsymbol{\Delta})$.
3. Note $\mathrm{tr}(\boldsymbol{A}) = \|\boldsymbol{G}\|_F^2 \sim \chi^2_{d^2}$ and $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
2. Let t estimate $\mathrm{tr}(\boldsymbol{A})$. Define $\tilde{t} := t - \mathrm{tr}(\boldsymbol{\Delta})$.
3. Note $\mathrm{tr}(\boldsymbol{A}) = \|\boldsymbol{G}\|_F^2 \sim \chi^2_{d^2}$ and $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$
    - $|t - \mathrm{tr}(\boldsymbol{A})| = |\tilde{t} - \mathrm{tr}(\tilde{\boldsymbol{A}})| \geq \Omega(d-k)$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
2. Let t estimate $\mathrm{tr}(\boldsymbol{A})$. Define $\tilde{t} := t - \mathrm{tr}(\boldsymbol{\Delta})$.
3. Note $\mathrm{tr}(\boldsymbol{A}) = \|\boldsymbol{G}\|_F^2 \sim \chi^2_{d^2}$ and $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$
    - $|t - \mathrm{tr}(\boldsymbol{A})| = |\tilde{t} - \mathrm{tr}(\tilde{\boldsymbol{A}})| \geq \Omega(d-k)$
    - $\mathrm{tr}(\boldsymbol{A}) \leq O(d^2)$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
2. Let t estimate $\mathrm{tr}(\boldsymbol{A})$. Define $\tilde{t} := t - \mathrm{tr}(\boldsymbol{\Delta})$.
3. Note $\mathrm{tr}(\boldsymbol{A}) = \|\boldsymbol{G}\|_F^2 \sim \chi^2_{d^2}$ and $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$
   - $|t - \mathrm{tr}(\boldsymbol{A})| = |\tilde{t} - \mathrm{tr}(\tilde{\boldsymbol{A}})| \geq \Omega(d-k)$
   - $\mathrm{tr}(\boldsymbol{A}) \leq O(d^2)$
4. Enforce $|t - \mathrm{tr}(\boldsymbol{A})| \leq \varepsilon\,\mathrm{tr}(\boldsymbol{A})$

$$(d-k) \leq \varepsilon \cdot Cd^2$$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
2. Let t estimate $\mathrm{tr}(\boldsymbol{A})$. Define $\tilde{t} := t - \mathrm{tr}(\boldsymbol{\Delta})$.
3. Note $\mathrm{tr}(\boldsymbol{A}) = \|\boldsymbol{G}\|_F^2 \sim \chi^2_{d^2}$ and $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$
   - $|t - \mathrm{tr}(\boldsymbol{A})| = |\tilde{t} - \mathrm{tr}(\tilde{\boldsymbol{A}})| \geq \Omega(d-k)$
   - $\mathrm{tr}(\boldsymbol{A}) \leq O(d^2)$
4. Enforce $|t - \mathrm{tr}(\boldsymbol{A})| \leq \varepsilon\, \mathrm{tr}(\boldsymbol{A})$
$$(d-k) \leq \varepsilon \cdot Cd^2$$
5. Set $d = \frac{1}{2C\varepsilon}$ and simplify: $k \geq \frac{1}{4C\varepsilon}$

Statistical Hypothesis Testing

Non-Adaptive Proof Framework

Design distributions $\mathcal{P}_0$ and $\mathcal{P}_1$, for large enough n:

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}) \times d}$ Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}+1) \times d}$ Gaussian

Statistical Hypothesis Testing

Non-Adaptive Proof Framework

Design distributions $\mathcal{P}_0$ and $\mathcal{P}_1$, for large enough n:

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}) \times d}$ Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}+1) \times d}$ Gaussian

1. A trace estimator can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$
   - If $\boldsymbol{A}_0 \sim \mathcal{P}_0$ and $\boldsymbol{A}_1 \sim \mathcal{P}_1$
   - With high probability, $\text{tr}(\boldsymbol{A}_0) \leq (1-2\varepsilon)\,\text{tr}(\boldsymbol{A}_1)$

Statistical Hypothesis Testing

Non-Adaptive Proof Framework

Design distributions $\mathcal{P}_0$ and $\mathcal{P}_1$, for large enough n:

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon})\times d}$ Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}+1)\times d}$ Gaussian

1. A trace estimator can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$
   - If $\boldsymbol{A}_0 \sim \mathcal{P}_0$ and $\boldsymbol{A}_1 \sim \mathcal{P}_1$
   - With high probability, $\text{tr}(\boldsymbol{A}_0) \leq (1-2\varepsilon)\,\text{tr}(\boldsymbol{A}_1)$
2. No algorithm can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$ with $\Omega(\frac{1}{\varepsilon})$ queries
   - Nature samples $i \sim \{0,1\}$, and $\boldsymbol{A} \sim \mathcal{P}_i$
   - User access $\boldsymbol{A}$ through the oracle

Statistical Hypothesis Testing

<u>Non-Adaptive Proof Framework</u>

Design distributions $\mathcal{P}_0$ and $\mathcal{P}_1$, for large enough n:

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}) \times d}$ Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}+1) \times d}$ Gaussian

1. A trace estimator can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$
   - If $\boldsymbol{A}_0 \sim \mathcal{P}_0$ and $\boldsymbol{A}_1 \sim \mathcal{P}_1$
   - With high probability, $\text{tr}(\boldsymbol{A}_0) \leq (1-2\varepsilon)\,\text{tr}(\boldsymbol{A}_1)$
2. No algorithm can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$ with $\Omega(\frac{1}{\varepsilon})$ queries
   - Nature samples $i \sim \{0, 1\}$, and $\boldsymbol{A} \sim \mathcal{P}_i$
   - User access $\boldsymbol{A}$ through the oracle
   - WLOG User picks standard basis vectors

Statistical Hypothesis Testing

Non-Adaptive Proof Framework

Design distributions $\mathcal{P}_0$ and $\mathcal{P}_1$, for large enough n:

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}) \times d}$ Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{(\frac{1}{\varepsilon}+1) \times d}$ Gaussian

1. A trace estimator can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$
    - If $\boldsymbol{A}_0 \sim \mathcal{P}_0$ and $\boldsymbol{A}_1 \sim \mathcal{P}_1$
    - With high probability, $\text{tr}(\boldsymbol{A}_0) \leq (1 - 2\varepsilon)\,\text{tr}(\boldsymbol{A}_1)$
2. No algorithm can distinguish $\mathcal{P}_0$ from $\mathcal{P}_1$ with $\Omega(\frac{1}{\varepsilon})$ queries
    - Nature samples $i \sim \{0, 1\}$, and $\boldsymbol{A} \sim \mathcal{P}_i$
    - User access $\boldsymbol{A}$ through the oracle
    - WLOG User picks standard basis vectors
    - Bound Total Variation between first k columns of $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$

Trace Estimation Summary

1. Introduced Hutchinson's Estimator for PSD $\boldsymbol{A}$
2. Improved it: Hutch++ uses $O(\frac{1}{\varepsilon})$
3. Two lower bounds: Adaptive & Non-Adaptive require $\Omega(\frac{1}{\varepsilon})$
4. Trace Estimation requires $\Theta(\frac{1}{\varepsilon})$ queries

Open Questions

- When is adaptivity helpful?
- What about inexact oracles? We often approximate $f(\boldsymbol{A})\mathbf{x}$ with iterative methods. How accurate do these computations need to be?
- Extend to include row/column sampling? This would encapsulate e.g. SGD/SCD.
- Memory-limited lower bounds? This is a realistic model for iterative methods.

THANK
YOU

Haim Avron and Sivan Toledo.
Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix.
Journal of the ACM, 58(2), 2011.

Tamas Sarlos.
Improved approximation algorithms for large matrices via random projections.
In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 143–152, 2006.

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

Karl Wimmer, Yi Wu, and Peng Zhang.
Optimal query complexity for estimating the trace of a matrix.
In Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP), pages 1051–1062, 2014.