Lessons from Trace Estimation

Testing, Communication, and Anti-Concentration

Raphael A. Meyer (New York University)

Collaborators

Christopher Musco
(NYU)

Cameron Musco
(UMass. Amherst)

David P. Woodruff
(CMU)

Hutch++: Optimal Stochastic Trace Estimation

Trace Estimation

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

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

Trace Estimation

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

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

- 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
$\operatorname{tr}(\frac{1}{6}\boldsymbol{B}^3)$	$\operatorname{tr}(e^{\boldsymbol{B}})$	$\operatorname{tr}(\ln(\boldsymbol{B}))$

Trace Estimation

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

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

- 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
$\text{tr}(\frac{1}{6}\boldsymbol{B}^3)$	$\text{tr}(e^{\boldsymbol{B}})$	$\text{tr}(\ln(\boldsymbol{B}))$

- If $\boldsymbol{A} = f(\boldsymbol{B})$, then we can often compute $\boldsymbol{A}\mathbf{x}$ quickly

Matrix-Vector Oracle Model

Idea: Matrix-Vector Product as a Computational Primitive

Matrix-Vector Oracle Model

Idea: Matrix-Vector Product as a Computational Primitive

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

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

- e.g. Krylov Methods, Sketching, Streaming, ...

Matrix-Vector Oracle Model

Idea: Matrix-Vector Product as a Computational Primitive

- Given access to a $n \times n$ 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, ...

Implicit Matrix 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.

$$(1-\varepsilon)\,\text{tr}(\boldsymbol{A}) \leq \tilde{\text{tr}}(\boldsymbol{A}) \leq (1+\varepsilon)\,\text{tr}(\boldsymbol{A})$$

- For constant failure probability, $k = \Theta(\frac{1}{\varepsilon})$ queries is optimal

Ideals for Lower Bounds

1. Generalization
   - Lower bounds beyond $\mathrm{tr}(\boldsymbol{A})$

Ideals for Lower Bounds

1. Generalization
   - Lower bounds beyond $\mathrm{tr}(\boldsymbol{A})$
2. Adaptivity
   - Can you use previous MatVec products to pick future ones?

Ideals for Lower Bounds

1. Generalization
   - Lower bounds beyond $\mathrm{tr}(\boldsymbol{A})$
2. Adaptivity
   - Can you use previous MatVec products to pick future ones?
3. Proof Complexity
   - Short proofs are nice.

Ideals for Lower Bounds

1. Generalization
   - Lower bounds beyond $\mathrm{tr}(\boldsymbol{A})$
2. Adaptivity
   - Can you use previous MatVec products to pick future ones?
3. Proof Complexity
   - Short proofs are nice.
4. Interpretable
   - What property of the hard distribution over inputs is important?
   - Trace estimation is hard for matrices that are nearly rank-$\frac{1}{\varepsilon}$

Communication Complexity

Given an instance of Gap-Hamming,

1. Define a matrix $\boldsymbol{A}$ in terms of $\mathbf{x}$ and $\mathbf{y}$ such that:
   - $(1 \pm \varepsilon)\,\mathrm{tr}(\boldsymbol{A})$ estimation solves Gap-Hamming
   - Alice and Bob can compute $\boldsymbol{A}\mathbf{x}$ with $\tilde{O}(\frac{1}{\varepsilon})$ bits
2. They can simulate any k-query algorithm with $\tilde{O}(\frac{k}{\varepsilon})$ bits
3. They must use $\Omega(\frac{1}{\varepsilon^2})$ bits, so $k = \tilde{\Omega}(\frac{1}{\varepsilon})$

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 user WLOG picks the first k standard basis 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 user WLOG picks the first k standard basis vectors

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

Non-Adaptive Proof Framework

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

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

Non-Adaptive Proof Framework

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

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{d \times (\frac{1}{\varepsilon})}$	Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{d \times (\frac{1}{\varepsilon}+1)}$	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, $\text{tr}(\boldsymbol{A}_0) \leq (1-2\varepsilon)\,\text{tr}(\boldsymbol{A}_1)$

Non-Adaptive Proof Framework

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

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

Non-Adaptive Proof Framework

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

$\mathcal{P}_0$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{d \times (\frac{1}{\varepsilon})}$	Gaussian
$\mathcal{P}_1$	$\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ for $\boldsymbol{G} \in \mathbb{R}^{d \times (\frac{1}{\varepsilon}+1)}$	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, $\mathrm{tr}(\boldsymbol{A}_0) \leq (1-2\varepsilon)\,\mathrm{tr}(\boldsymbol{A}_1)$
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 d:

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

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 $\boldsymbol{A} = \boldsymbol{G}^\intercal \boldsymbol{G}$.
- 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 $\boldsymbol{A} = \boldsymbol{G}^\intercal \boldsymbol{G}$.
- 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 $\boldsymbol{A} = \boldsymbol{G}^\intercal \boldsymbol{G}$.
- 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. $\text{tr}(\boldsymbol{A}) = \text{tr}(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^\intercal) = \text{tr}(\boldsymbol{\Delta}) + \text{tr}(\tilde{\boldsymbol{A}})$

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

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

Wigner/Wishart Anti-Concentration Method

Consider any adaptive algorithm after k steps:

1. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{VAV}^\top) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
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. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{VAV}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
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 $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$
    - $|t - \mathrm{tr}(\boldsymbol{A})| = |\tilde{t} - \mathrm{tr}(\tilde{\boldsymbol{A}})| \geq \Omega(d-k)$
    - $\mathrm{tr}(\boldsymbol{A}) \leq O(d^2)$
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. $\mathrm{tr}(\boldsymbol{A}) = \mathrm{tr}(\boldsymbol{VAV}^\intercal) = \mathrm{tr}(\boldsymbol{\Delta}) + \mathrm{tr}(\tilde{\boldsymbol{A}})$
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 $\mathrm{tr}(\tilde{\boldsymbol{A}}) \sim \chi^2_{(d-k)^2}$
    - $|t - \mathrm{tr}(\boldsymbol{A})| = |\tilde{t} - \mathrm{tr}(\tilde{\boldsymbol{A}})| \geq \Omega(d-k)$
    - $\mathrm{tr}(\boldsymbol{A}) \leq O(d^2)$
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: $k \geq \frac{1}{4C\varepsilon}$

Open Questions

- **In progress:** Lower bounds for e.g. $\text{tr}(\boldsymbol{A}^3)$, $\text{tr}(e^{\boldsymbol{A}})$, $\text{tr}(\boldsymbol{A}^{-1})$
- 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