Trace Estimation

Raphael Meyer, Caltech

UMich EECS 598-004 Randomized Numerical Linear Algebra in Machine Learning

Caltech

Motivation 1: Road Network Connectivity

NYC is adding a new bus line. Where should it go?

Idea: Maximize *connectivity* of transit network using existing bus stops

Given: Adjacency matrix $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ in memory

Algorithm: For each possible new bus route (edge), build new $\boldsymbol{B}' \in \mathbb{R}^{n \times n}$, and compute the connectivity of $\boldsymbol{B}'$.

Return: Edge that maximized connectivity

Bottleneck: Computing connectivity

[Oversimplification of "Public Transport Planning: When Transit Network Connectivity Meets Commuting Demand" by Wang et al.]

Computational Bottleneck

- We have matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ in memory
- We want to compute *connectivity* of $\boldsymbol{B}$:

  Estrada Index: $tr\left(e^{\boldsymbol{B}}\right)$

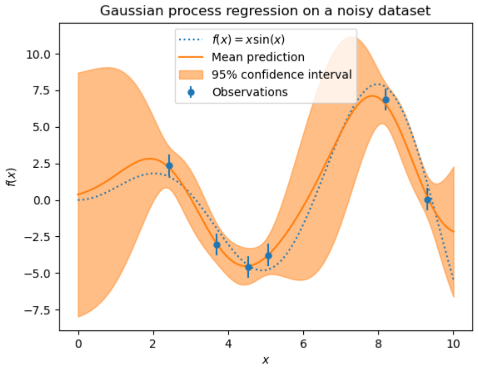
  Num of Triangles: $tr\left(\frac{1}{6}\boldsymbol{B}^3\right)$
- The trace is the sum of the diagonal of a matrix.
- Computing $\boldsymbol{B}^3$ takes $O\left(n^3\right)$ time. *slow*
- Computing $\boldsymbol{B}^3 x = \boldsymbol{B}(\boldsymbol{B}(\boldsymbol{B}x))$ takes $O\left(n^2\right)$ time. *fast*

Can we approximate $tr(\boldsymbol{B}^3)$ by computing few $\boldsymbol{B}^3 x_1, \dots, \boldsymbol{B}^3 x_m$?

- **Yes, we can!**

Caltech

Motivation 2: Gaussian Process Optimization



We have time series data, and want to interpolate it with kernel k_θ

Idea: Maximize log-likelihood kernel matrix K_θ over hyperparameter θ

Given: Data points $(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_\mathrm{n}, \mathrm{y}_\mathrm{n})$, kernel function $k_\theta(x, x')$

Algorithm: Compute Gradient Descent of Log-Likelihood to optimize θ

Return: Optimal θ

Bottleneck: Computing gradients

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{2} y^T \boldsymbol{K}^{-1} \frac{d\boldsymbol{K}}{d\theta} \boldsymbol{K}^{-1} y - \frac{1}{2} tr\left(\boldsymbol{K}^{-1} \frac{d\boldsymbol{K}}{d\theta}\right)$$

Caltech

[Image Source: scikit learn Chapter 1.7: Gaussian Processes]

Computing $K^{-1} \frac{dK}{d\theta} x$ takes $O(n^2)$ time, much faster than $O(n^3)$!

General Picture: Trace Estimation

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

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

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

Num. Triangles	Estrada Index	Log-Determinant
$tr\left(\frac{1}{6}\boldsymbol{B}^3\right)$	$tr\left(e^{\boldsymbol{B}}\right)$	$tr(\ln(\boldsymbol{B}))$

- If $\mathbf{A} = f(\boldsymbol{B})$, then we can often compute $\boldsymbol{A}x$ quickly
  - Compute $\boldsymbol{B}^q x$ in $\tilde{O}(n^2\sqrt{q})$ time
  - Lanczos-FA: compute $\boldsymbol{A}x$ in $\tilde{O}(n^2\sqrt{\kappa(B)})$ time for many f

- Goal: Estimate $tr(\boldsymbol{A})$ by computing $\boldsymbol{A}x_1, \ldots, \boldsymbol{A}x_m$

Caltech

Matrix-Vector Oracle Model

Think of the Matrix-Vector Product as a Computational Primitive

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

$$x \xRightarrow{input} ORACLE \xRightarrow{output} \boldsymbol{A}x$$

- e.g. Randomized SVD, Johnson-Lindenstrauss, Power Method, Lanczos-FA
- Some existing lower bounds; very active area of research!

Trace Estimation: Estimate $tr(\boldsymbol{A})$ with as few Matrix-Vector products $\boldsymbol{A}x_1, \dots, \boldsymbol{A}x_k$ as possible.

$$|\mathrm{tr}(\boldsymbol{A}) - \widetilde{tr}(\boldsymbol{A})| \leq \varepsilon\, tr(\boldsymbol{A})$$

Caltech

Q: We will assume A is PSD. Why does that help us?

Outline

- Part 1: The Girard-Hutchinson Trace Estimator
  - 2 Lines of MATLAB Code
- Part 2: The Hutch++ Trace Estimation
  - 5 Lines of MATLAB Code
- Part 3: Extensions of Hutch++
  - 3 Follow-up papers, at a very high level
- Part 4: Lower Bounds
  - No algorithm can get better big-Oh than Hutch++!

Caltech

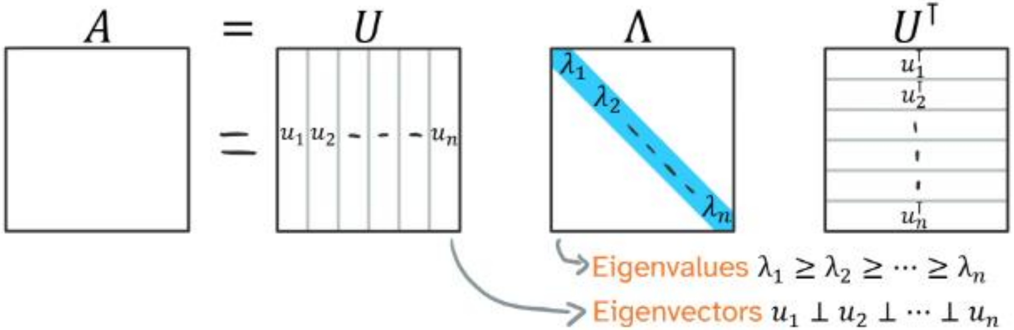
Part 1: Girard-Hutchinson

Part 2: Hutch++

Caltech

Linear Algebra Notation

- Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be symmetric. Then, $\boldsymbol{A} = \boldsymbol{U\Lambda U}^T$



- $\boldsymbol{A}$ is Positive Semi-Definite (PSD) if all $\lambda_i \geq 0$
- Trace $tr(\boldsymbol{A}) = \sum_{i=1}^{n} \boldsymbol{A}_{ii} = \sum_{i=1}^{n} \lambda_i$ 
- Frobenius Norm $||\boldsymbol{A}||_F^2 = \sum_{i=1}^{n} \boldsymbol{A}_{ii}^2 = \sum_{i=1}^{n} \lambda_i^2$ 
- For PSD matrices, $||\boldsymbol{A}||_F \leq tr(\boldsymbol{A})$

Caltech

Girard-Hutchinson Trace Estimator

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

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

- Girard-Hutchinson Estimator: $H_\ell(\boldsymbol{A}) := \frac{1}{\ell}\sum_{i=1}^{\ell} x_i^T \boldsymbol{A} x_i$

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

Lemma: $H_\ell(\boldsymbol{A})$ needs $\ell = O\left(\frac{1}{\varepsilon^2}\right)$ for PSD $\boldsymbol{A}$

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

$$\begin{aligned} |H_\ell(\boldsymbol{A}) - tr(\boldsymbol{A})| &\leq O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F && \text{(Chebyshev)} \\ &\leq O(\tfrac{1}{\sqrt{\ell}}) tr(\boldsymbol{A}) && (\|\boldsymbol{A}\|_F \leq tr(\boldsymbol{A})) \\ &= \varepsilon\, tr(\boldsymbol{A}) && (\ell = O(\tfrac{1}{\varepsilon^2})) \end{aligned}$$

Caltech

Girard-Hutchinson Trace Estimator

- When is this analysis tight?

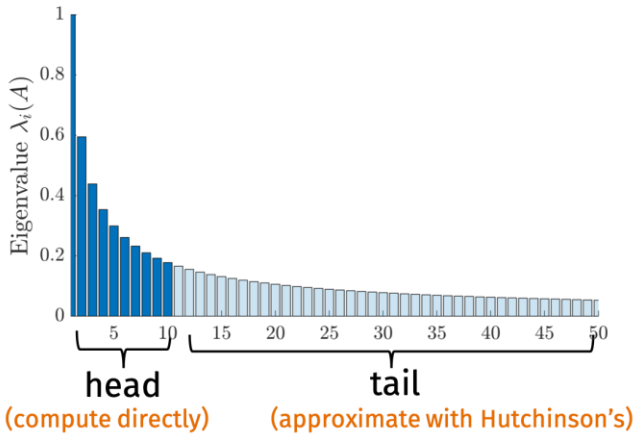
$$
\begin{aligned}
|H_\ell(\boldsymbol{A}) - tr(\boldsymbol{A})| &\approx O(\tfrac{1}{\sqrt{\ell}})\|\boldsymbol{A}\|_F && \text{(Chebyshev)} \\
&\leq O(\tfrac{1}{\sqrt{\ell}})tr(\boldsymbol{A}) && (\|\boldsymbol{A}\|_F \leq tr(\boldsymbol{A})) \\
&= \varepsilon\, tr(\boldsymbol{A}) && (\ell = O(\tfrac{1}{\varepsilon^2}))
\end{aligned}
$$

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

- The estimator only needs $O(\frac{1}{\varepsilon^2})$ when $\boldsymbol{A}$ has few large eigenvalues!

Caltech

Helping Hutchinson's Estimator



- Idea: Explicitly estimate top few eigenvals of $\boldsymbol{A}$. Use Hutchinson for rest.

1. Find a good rank-k approximation $\widetilde{\boldsymbol{A}_k}$
2. Notice that $tr(\boldsymbol{A}) = tr(\boldsymbol{A}_k) + tr(\boldsymbol{A} - \boldsymbol{A}_k)$
3. Compute $tr(A_k)$ exactly
4. Return Hutch++$(\boldsymbol{A}) \coloneqq tr(\widetilde{\boldsymbol{A}_k}) + H_\ell(\boldsymbol{A} - \widetilde{\boldsymbol{A}_k})$

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

Finding a good Low-Rank Approximation

- Recall Randomized SVD:
  1. Let $\boldsymbol{S} \in \mathbb{R}^{n\times(2k+1)}$ have iid $\mathcal{N}(0,1)$ entries
  2. Compute $\boldsymbol{Q} = orth(\boldsymbol{AS})$
  3. Then $\widetilde{\boldsymbol{A}_k} = \boldsymbol{AQQ}^{\mathrm{T}}$ is pretty good:

$$\mathbb{E}\left\|\boldsymbol{A} - \widetilde{\boldsymbol{A}_k}\right\|_F^2 \leq 2\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$$

- Notice $tr(\widetilde{\boldsymbol{A}_k})$ can be computed quickly:

$$tr\left(\widetilde{\boldsymbol{A}_k}\right) = tr\left(\boldsymbol{AQQ}^T\right) = tr\left(\boldsymbol{Q}^T\boldsymbol{AQ}\right)$$

Caltech

Hutch++

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

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

```
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
```

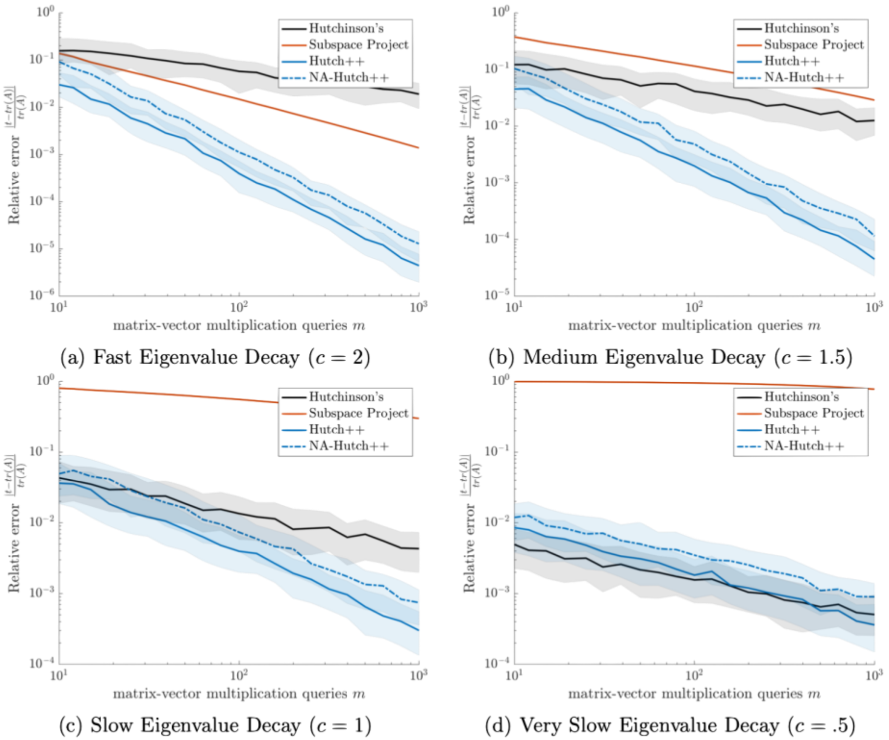
$H_\ell(A) = \frac{1}{\ell} tr\left(G^T A G\right)$ is another way to write the Girard-Hutchinson estimator.

Experiments – Hutch++ vs Girard-Hutchinson

Hutch++ is fastest when $\boldsymbol{A}$ has fast eigenvalue decay. Here, $\lambda_i = i^{-c}$

Error ε

Error ε



Matrix-Vector Products m Matrix-Vector Products m

Caltech

Checkpoint – Hutch++

Can we approximate $tr(\boldsymbol{A})$ by computing few $\boldsymbol{A}x_1, \dots, \boldsymbol{A}x_m$?

- **Yes, we can!**
- Girard-Hutchinson $H_\ell(\boldsymbol{A})$ uses $O\left(\frac{1}{\varepsilon^2}\right)$ matrix-vector products
- We can estimate $tr(\boldsymbol{A})$ with $O(\frac{1}{\varepsilon})$ matrix-vector products
- Four step algorithm:
  1. Find Low-Rank Approximation $\widetilde{\boldsymbol{A}_k}$
  2. Compute $tr\left(\widetilde{\boldsymbol{A}_k}\right)$
  3. Compute $H_\ell\left(\boldsymbol{A} - \widetilde{\boldsymbol{A}_k}\right)$
  4. Return $tr\left(\widetilde{\boldsymbol{A}_k}\right) + H_\ell\left(\boldsymbol{A} - \widetilde{\boldsymbol{A}_k}\right)$

Caltech

Part 3: Extensions of Hutch++

Caltech

Practical Considerations

- What if my matrix is not PSD?

  This is fine. Same proof shows $|\text{Hutch++}(\boldsymbol{A}) - \operatorname{tr}(\mathbf{A})| \leq \varepsilon \|\boldsymbol{A}\|_1$

- Can we estimate the variance of Hutch++?
- Can we maximize parallelism without wasting matrix-vector products?
- Can we just be more efficient?
- Can we take advantage of the fact that $\boldsymbol{A} = f(\boldsymbol{B})$?

Caltech

A Posteriori Variance Estimation

- What is the confidence interval / actual variance of my Hutch++ sample?
- Intuitive answer: Sample variance of the Girard-Hutchinson Samples.
- Better answer: use Bootstrapping to estimate the true variance below

Lemma: $Var[\text{Hutch++}(\boldsymbol{A})] = \mathbb{E}[\, Var[\, H_\ell(A - \widetilde{A_k}) \mid Q] \,]$

Proof. By law of total variance,

$$Var[\text{Hutch++}(\boldsymbol{A})] = \mathbb{E}[Var[\text{Hutch++}(\boldsymbol{A})|\boldsymbol{Q}]\,] + Var[\mathbb{E}[\text{Hutch++}(\boldsymbol{A})|\boldsymbol{Q}]]$$

But observe that

$$\mathbb{E}[\text{Hutch++}(\boldsymbol{A})|\boldsymbol{Q}] = tr(\widetilde{\boldsymbol{A}_k}) + tr(\boldsymbol{A} - \widetilde{\boldsymbol{A}_k}) = tr(\boldsymbol{A})$$

So

$$Var[\mathbb{E}[\text{Hutch++}(\boldsymbol{A})|\boldsymbol{Q}]] = 0$$

Caltech

Maximizing Parallelism

- To maximize parallelism, we want none of our matrix-vector products to depend on previous matrix-vector products.

- In Hutch++, we must compute $\boldsymbol{AS}$ to compute $\boldsymbol{Q}$ to compute $\widetilde{\boldsymbol{A}_k} = \boldsymbol{AQQ}^T$

- Formally, this is an adaptive algorithm:

$$
\begin{tikzcd}
\mathbf{x}_{k+1} \arrow[r, Rightarrow] & \text{ORACLE} \arrow[r, Rightarrow] & \boldsymbol{A}\mathbf{x}_k \arrow[d, no head] \\
\arrow[u] & \text{ALGORITHM} \arrow[l, no head] & \arrow[l]
\end{tikzcd}
$$

- Non-obvious question: is trace estimation possible non-adaptively?

$$
\begin{array}{ccccc}
\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} & \Longrightarrow & \text{ORACLE} & \Longrightarrow & \{\boldsymbol{A}\mathbf{x}_1, \ldots, \boldsymbol{A}\mathbf{x}_m\} \\
\uparrow & & & & \downarrow \\
\text{ALGORITHM} & & & & \text{ALGORITHM}
\end{array}
$$

Nyström++

- Idea: Use the Nyström approximation to PSD $\boldsymbol{A}$:
  - $\widetilde{\boldsymbol{A}}_k = \boldsymbol{A}\langle\boldsymbol{S}\rangle \coloneqq \boldsymbol{AS}\left(\boldsymbol{S}^T\boldsymbol{AS}\right)^+\boldsymbol{S}^T\boldsymbol{A}$
  - Unlike before, we construct $\widetilde{\boldsymbol{A}}_k$ non-adaptively!

- Updated Algorithm:
  1. Use $\frac{m}{2}$ matrix-vector products to build $\widetilde{\boldsymbol{A}}_k$
  2. Use $\frac{m}{2}$ matrix-vector products to compute $x_i^T\boldsymbol{A}x_i$
  3. Return $tr\left(\widetilde{\boldsymbol{A}}_k\right) + \frac{1}{m/2}\sum_{i=1}^{m/2}(x_i^T\boldsymbol{A}x_i - x_i^T\widetilde{\boldsymbol{A}}_k x_i)$

- Non-adaptive and basically always as fast/faster than Hutch++!

- Analysis is more involved

Caltech

[Persson Cortinovis Kressner “Improved variants of the Hutch++ algorithm for trace estimation”] -- More cool ideas are in here!

XTrace & Exchangeability

- Motivation: Exchangeability Principle in Statistics
  - *Informally*: any algorithm that uses iid samples, is unbiased, and has optimally small variance must be exchangeable.
  - If we reorder our samples, the algorithm's output doesn't change.

- The Girard-Hutchinson Estimator is exchangeable
  - $H_\ell(\boldsymbol{A}) = \sum_{i=1}^{\ell} x_i^T \boldsymbol{A} x_i$

- Hutch++ and Nyström++ are NOT exchangeable!
  - Why?

- XTrace: leave-one-out version of Hutch++ that is exchangeable
  XNysTrace: Nyström version of Xtrace

- Even faster than Nyström++ in most cases!

Caltech

[Epperly Tropp Webber "XTrace: Making the Most of Every Sample in Stochastic Trace Estimation"] -- More cool ideas are in here

Krylov-Aware Trace Estimation

- Remember we often have $\boldsymbol{A} = f(\boldsymbol{B})$ where $\boldsymbol{B}$ is in memory
- Use Lanczos-FA Algorithm to approximately compute $x \mapsto f(\boldsymbol{B})x$

- Observation: Hutch++ style algorithms throw away pre-computed information if matrix-vector products with $\boldsymbol{A}$ are computed via Lanczos-FA

- "Krylov-Aware" Trace Estimator

  Takes $\boldsymbol{B}, f, m$ as inputs

  Avoids throwing away information

- Fastest algorithm I know for estimating $f(\boldsymbol{B})$

- Maybe incompatible with XTrace style algorithms?

Caltech

[Chen Hallman "Krylov-Aware Stochastic Trace Estimation"] -- More cool ideas are in here too!

Checkpoint – Hutch++ Extentions

Can we approximate $tr(\boldsymbol{A})$ *by computing few* $\boldsymbol{A}x_1, \dots, \boldsymbol{A}x_m$*?*

- **Yes, we can! Non-adaptively and super efficiently!**

- Practitioner Advice:
  - XNysTrace for PSD Matrices
  - XTrace for non-PSD Matrices
  - Krylov-Aware Trace Estimation for $\boldsymbol{A} = f(\boldsymbol{B})$

- Natural theoretical question: Is better than $O(\frac{1}{\varepsilon})$ possible?
  - **No!** $\Omega(\frac{1}{\varepsilon})$ is optimal!

Caltech

Part 4: Lower Bounds

Lower Bound Super Rough Intuition

$$x \overset{input}{\Longrightarrow} ORACLE \overset{output}{\Longrightarrow} \boldsymbol{A}x$$

- View the Matrix-Vector Oracle as a limit on information about $\boldsymbol{A}$:
  1. Suppose $\boldsymbol{A} \sim \mathcal{D}$ is a random matrix
  2. Then $tr(\boldsymbol{A})$ is a random variable with variance
  3. Any algorithm that computes few oracle queries has little information about $tr(\boldsymbol{A})$
  4. Then, due to variance, the algorithm cannot predict $tr(\boldsymbol{A})$ well
- Step 3 is hard to prove rigorously.

Caltech

Removing the Algorithm's Agency

- **Want to show:** Any algorithm that computes few oracle queries has little information about $tr(\boldsymbol{A})$
- **Problem:** Algorithm can pick many different query vectors x
- If instead algorithm had no freedom, we could use classical statistics to make lower bounds.

Two Observations:

1. Without loss of generality, query vectors are orthonormal. (Why?)
2. Let $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ be iid $\mathcal{N}(0,1)$ matrix
   Let $\mathbf{Q} \in \mathbb{R}^{n \times k}$ have orthonormal columns
   Then $\boldsymbol{GQ}$ is a $\mathcal{N}(0,1)$ matrix, even conditioned on knowing $\boldsymbol{Q}$.
   (informal) If $\boldsymbol{A}$ is Gaussian, then $\{\boldsymbol{A}x_1, \dots, \boldsymbol{A}x_m\}$ is independent of $\{x_1, \dots, x_m\}$

Caltech

(informal) Without loss of generality, algorithm looks at first k cols of $\boldsymbol{A}$

Wishart Anti-Concentration Method

Hidden Wishart Theorem:

- Let $\boldsymbol{G} \in \mathbb{R}^{n\times n}$ be a $\mathcal{N}(0,1)$ matrix
- Let $\boldsymbol{A} = \boldsymbol{G}^T\boldsymbol{G}$ be a Wishart Matrix
- Suppose *Algorithm* computes $\boldsymbol{A}x_1, \dots, \boldsymbol{A}x_m$, possibly adaptively.
- Then, there exists orthogonal $V \in \mathbb{R}^{n\times n}$ such that

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

  where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{(n-m)\times(n-m)}$ is distributed as $\tilde{\boldsymbol{G}}^T\tilde{\boldsymbol{G}}$ where $\tilde{\boldsymbol{G}}$ is $\mathcal{N}(0,1)$, conditioned on all observations $\{x_1, \boldsymbol{A}x_1, \dots, x_m, \boldsymbol{A}x_m\}$

- $\boldsymbol{\Delta}$ is known deterministically by the algorithm

We can exactly separate the information we do/don't know about $\boldsymbol{A}$!

Caltech

[Braverman, Hazan, Simchowitz, Woodworth – "The Gradient Complexity of Linear Regression"]

Wishart Anti-Concentration Method

- Consider any (possibly adaptive) algorithm after m queries. Then,

1. $tr(\boldsymbol{A}) = tr\left(\boldsymbol{VAV}^T\right) = tr(\boldsymbol{\Delta}) + tr\left(\widetilde{\boldsymbol{A}}\right)$
2. Let t be *Algorithm*'s estimate of $tr(\boldsymbol{A})$. Define $\tilde{t} := t - tr(\boldsymbol{\Delta})$
3. Note $tr(\boldsymbol{A}) = \|\boldsymbol{G}\|_F^2 \sim \chi^2_{n^2}$ and $tr\left(\widetilde{\boldsymbol{A}}\right) \sim \chi^2_{(n-m)^2}$
   - $|t - tr(\boldsymbol{A})| = \left|\tilde{t} - tr\left(\tilde{A}\right)\right| \geq \Omega(n-m)$
   - $tr(\boldsymbol{A}) \leq O\left(n^2\right)$
4. Enforce $|t - tr(\boldsymbol{A})| \leq \varepsilon\, tr(\boldsymbol{A})$

   $$(n-m) \leq \varepsilon C n^2$$
5. Set $n = \frac{1}{2C\varepsilon}$ and simplify: $m \geq \frac{1}{4C\varepsilon}$

Caltech

[Chen Hallman "Krylov-Aware Stochastic Trace Estimation"] -- More cool ideas are in here too!

Checkpoint – Lower Bounds

Can we approximate $tr(\boldsymbol{A})$ by computing few $\boldsymbol{A}x_1, \dots, \boldsymbol{A}x_m$?

- **Yes, but only if we take $\mathbf{m} = \boldsymbol{\Omega}\left(\frac{\mathbf{1}}{\boldsymbol{\varepsilon}}\right)$ queries!**

- Proven via Hidden Wishart, but other methods exist:
    - Communication Complexity
    - Statistical Hypothesis Testing
    - Block Krylov Reduction

- Hidden second meaning: Wishart matrices are unstructured covariance matrices.
- If we want to estimate the trace of a covariance matrix (e.g. a Kernel matrix or Hessian), then do not expect that better than Hutch++ is possible! Otherwise, faster might exist!

Caltech

The End.

Thanks!

Questions?

Caltech