Optimal Stochastic Trace Estimation

Christopher Musco

New York University, Tandon School of Engineering

COLLABORATORS

Raphael Meyer
(NYU)

Cameron Musco
(UMass. Amherst)

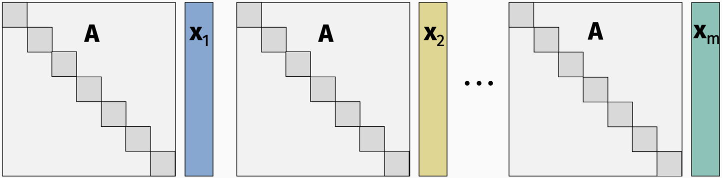
David Woodruff
(CMU)

Paper available at: https://arxiv.org/pdf/2010.09649.pdf.

Recently accepted to the Symposium on Simplicity in Algorithms (SOSA 2021).

Basic problem in linear algebra:

- Given access to a $n \times n$ matrix $\mathbf{A}$ through a matrix-vector multiplication oracle.
- Goal is to (approximately) compute $\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} \mathbf{A}_{ii}$.



Main question: How many matrix-vector multiplication "queries" $\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_m$ are required to compute $\text{tr}(\mathbf{A})$?[1]

[1] $\mathbf{x}_i$ can be chosen adaptively, based on result of $\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_{i-1}$.

Algorithms in this model are called matrix-free, or implicit matrix methods.

Typically useful when $\mathbf{A}$ is not stored explicitly, but we have an efficient algorithm for multiplying $\mathbf{A}$ by a vector.

Example: Hessian matrix-vector products.

Suppose we have some function $f(\mathbf{y})$ and we can efficiently compute gradients $\nabla f(\mathbf{y})$ for any $\mathbf{y}$. Let $\mathbf{A} = \nabla^2 f(\mathbf{y})$. Then:

$$\mathbf{A}\mathbf{x} \approx \frac{\nabla f(\mathbf{y} + \eta \mathbf{x}) - \nabla f(\mathbf{y})}{\eta} \quad \text{for sufficiently small } \eta.$$

Also important when $\mathbf{A}$ is a function of another matrix $\mathbf{B}$:

$$\mathbf{A} = f(\mathbf{B})$$

Common examples:

$$\mathbf{A} = \mathbf{B}^T\mathbf{B} \qquad \mathbf{A} = \mathbf{B}^3 \qquad \mathbf{A} = 2\mathbf{B}^3 - 3\mathbf{B}^2 - \mathbf{I}$$

Cost to compute $\mathbf{A}$ and $\text{tr}(\mathbf{A})$ explicitly:

$$O(n^3) \qquad O(n^3) \qquad O(n^3)$$

Cost to compute matrix-vector multiplication $\mathbf{Ax}$:

$$O(n^2) \qquad O(n^2) \qquad O(n^2)$$

All cheaper by a factor of n! Even more savings if A is sparse or structured.

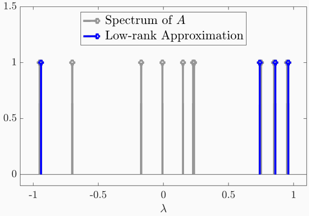
For more complex matrix functions, we can often compute $\mathbf{A}\mathbf{x} = f(\mathbf{B})\mathbf{x}$ efficiently using iterative methods:

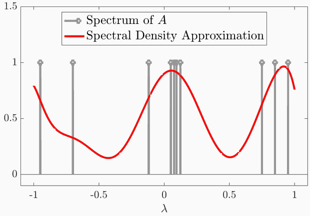
- Conjugate gradient, or any other system solver: $\mathbf{A} = \mathbf{B}^{-1}$.
- Lanczos method: $\mathbf{A} = \exp(\mathbf{B})$, $\mathbf{A} = \sqrt{\mathbf{B}}$, $\mathbf{A} = \log(\mathbf{B})$, etc.

All run in $O(n^2 \cdot C)$ time, where C depends on properties of $\mathbf{B}$. For example, for $\mathbf{A} = \mathbf{B}^{-1}$, $C = \sqrt{\kappa} \cdot \log(1/\epsilon)$.

In practice, we typically have $O(n^2 \cdot C) \ll O(n^3)$.

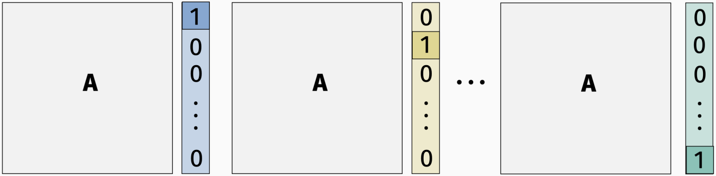
- Log-likelihood computation in Bayesian optimization, experimental design. $\mathrm{tr}(\log(\mathbf{B})) = \mathrm{logdet}(\mathbf{B})$.
- Estrada index, network connectivity. $\mathrm{tr}(\exp(\mathbf{B}))$.
- Triangle counting in graphs. $\mathrm{tr}(\exp(\mathbf{B}^3))$.
- Counting number of eigenvalues in an interval.
- Spectral density estimation.
- Matrix norms.





Naive approach:

- Set $\mathbf{x}_i = \mathbf{e}_i$ for $i = 1, \ldots, n$.
- Return $\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} \mathbf{x}_i^T \mathbf{A} \mathbf{x}_i$



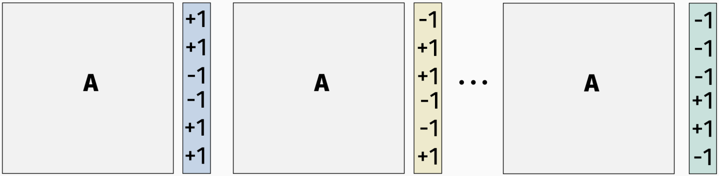
Returns exact solution, but requires n matrix-vector multiplies. We want $\ll n$ multiplies, and will do so by allowing for approximation.

HUTCHINSON'S STOCHASTIC TRACE ESTIMATOR

Simple, powerful, and widely used method for trace estimation.

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



Let $\tilde{T}$ be the trace estimate returned by Hutchinson's method.

Claim (Avron, Toledo 2011, Roosta, Ascher 2015)

If $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, *then with probability* $(1-\delta)$,

$$\left|\tilde{T} - \text{tr}(\mathbf{A})\right| \leq \epsilon \|\mathbf{A}\|_F.$$

If $\mathbf{A}$ is symmetric positive semidefinite (PSD) with eigenvalues $\lambda_1, \ldots, \lambda_n$, then

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

Corollary: For PSD A: $(1-\epsilon)\,\text{tr}(\mathbf{A}) \leq \tilde{T} \leq (1+\epsilon)\,\text{tr}(\mathbf{A})$.

Hutchinson's Estimator:

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

Expected value analysis:

For a single random ± 1 vector $\mathbf{x}$,

$$\mathbb{E}[\tilde{T}] = \mathbb{E}[\mathbf{x}^T\mathbf{A}\mathbf{x}] = \mathbb{E}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \mathbf{A}_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}[x_i x_j \mathbf{A}_{ij}] = \sum_{i=1}^{n} \mathbf{A}_{ii}$$

So the estimator is correct in expectation:

$$\mathbb{E}[\tilde{T}] = \text{tr}(\mathbf{A}).$$

Hutchinson's Estimator:

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

Variance analysis:

$$\text{Var}[\tilde{T}] = \frac{1}{m}\text{Var}[\mathbf{x}^T\mathbf{A}\mathbf{x}] = \frac{1}{m}\text{Var}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \mathbf{A}_{ij}\right]$$

$$= \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{n}\text{Var}[x_i x_j \mathbf{A}_{ij}] = \frac{1}{m}\sum_{i\neq j}^{n}\mathbf{A}_{ij}^2 \leq \frac{1}{m}\|\mathbf{A}\|_F^2$$

(We used that $x_i x_j$ and $x_j x_k$ are pairwise independent.)

Hutchinson's Estimator:

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

Final analysis: Chebyshev's inequality implies that, with probability 9/10,

$$\left|\tilde{T} - \text{tr}(\mathbf{A})\right| \leq \frac{1}{\sqrt{m/10}} \|\mathbf{A}\|_F.$$

Setting $m = O\left(1/\epsilon^2\right)$ gives $\left|\tilde{T} - \text{tr}(\mathbf{A})\right| \leq \epsilon \|\mathbf{A}\|_F$.

Getting correct $\log(1/\delta)$ dependence requires a bit more work (Hanson-Wright inequality).

Result: $O\left(1/\epsilon^2\right)$ matrix-vector multiplies suffice to return, with prob. 9/10, a trace estimate for a PSD matrix with relative error:

$$(1-\epsilon)\operatorname{tr}(\mathsf{A}) \leq \tilde{T} \leq (1\pm\epsilon)\operatorname{tr}(\mathsf{A}).$$

Research Question: Is this tight?

Broader line of work: Tight upper bounds and lower bounds on complexity of basic linear algebra problems in "matrix-vector query" model.

- **Top eigenvector:** Simchowitz, Alaoui, Recht, 2018.
- **Least squares regression:** Braverman, Hazan, Simchowitz, Woodworth, 2020.
- **Rank, symmetry test, and more:** Sun, Woodruff, Yang, and Zhang, 2019.

The matrix-vector query model generalizes the most common models of computations in linear algebra.

Krylov subpace model:

- Compute $\mathbf{A}\mathbf{x}, \mathbf{A}^2\mathbf{x}, \ldots, \mathbf{A}^m\mathbf{x}$ for chosen vector $\mathbf{x}$.
- Lower bounds typically via approximation theoretic arguments (understanding the limits of polynomials).

Matrix sketching model:

- Compute $\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_m$ where $\mathbf{x}_1 \ldots, \mathbf{x}_m$ are chosen non-adaptivity (usually chosen to be random vectors).
- Lower bounds typically via one-round communication complexity.

Merits of this model:

- Captures most algorithms that are used in practice, where matrix-vector multiplies often dominate computation cost.
- Allowing arbitrary adaptivity makes the model quite a bit richer. Proving lower bounds seems harder but doable.
- Appears to be a "sweet spot" for understanding problem complexity in linear algebra.

Limitation:

- Does not capture methods like stochastic gradient or coordinate descent.

Upper bound: $O(1/\epsilon)$ matrix-vector multiplies suffice to return, with prob. 9/10, a trace estimate for a PSD matrix with relative error:

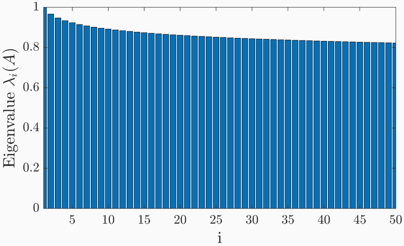
$$(1-\epsilon)\,\mathrm{tr}(\mathbf{A}) \leq \tilde{T} \leq (1+\epsilon)\,\mathrm{tr}(\mathbf{A}).$$

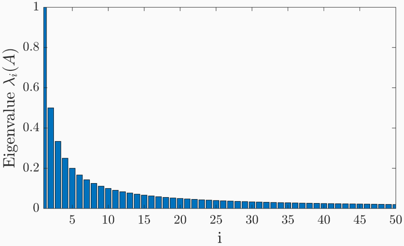
- Quadratic improvement over Hutchinson's $O(1/\epsilon^2)$.
- Algorithm achieving bound is nearly as simple.
- Performs much better experimentally.

Lower bound: $\Omega(1/\epsilon)$ matrix-vector multiplies are necessary to obtain a relative error approximation with probability $> 2/3$.

- Two different approaches: reduction from multi-round communication complexity, and from hypothesis testing for negatively spiked covariance matrices.

Observation: Hutchinson's method performs much better when $\mathbf{A}$ has a "flatter" spectrum.



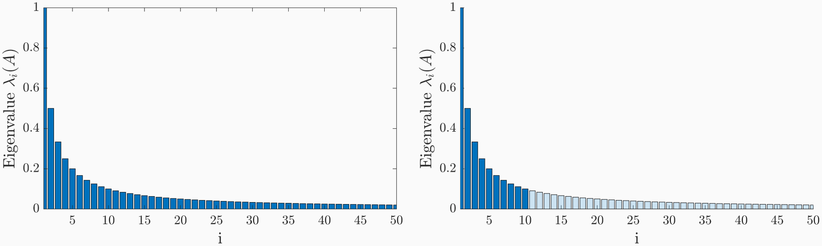


We proved that: $|\tilde{T} - \text{tr}(\mathbf{A})| \leq \epsilon\|\mathbf{A}\|_F \leq \epsilon\,\text{tr}(\mathbf{A})$, but when the spectrum is decaying $\|\mathbf{A}\|_F \ll \text{tr}(\mathbf{A})$.

In the extreme case when $\lambda_1 = \lambda_2 = \ldots = \lambda_n$, we have:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n} \lambda_i^2} = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \lambda_i = \frac{1}{\sqrt{n}}\,\text{tr}(\mathbf{A}).$$

On the other hand, when A's spectrum is decaying, we get a good approximation by simply computing its top eigenvectors.



$$\mathrm{tr}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i \approx \sum_{i=1}^{k} \lambda_k = \mathrm{tr}(\mathbf{A}\mathbf{Q}\mathbf{Q}^T)$$

where $\mathbf{Q} \in \mathbb{R}^{n \times m}$ is an orthonormal span A's top k eigenvalues.

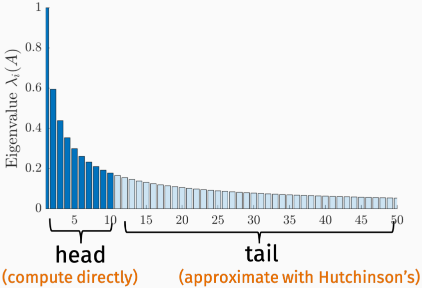
.

- Q itself can be computed with $\sim O(k)$ matrix-vector multiplication queries using block power method or a Krlyov method (Saibaba, Alexanderian, Ipsen, 2018).
- Then $\text{tr}(\mathbf{AQQ}^T) = \text{tr}(\mathbf{Q}^T(\mathbf{AQ}))$ can be computed with k additional matrix-vector multiplies.

Main observation: Every spectrum is either "flat enough" or "decaying enough" to prove a better bound than $O(1/\epsilon^2)$.

OUR METHOD: HUTCH++

1. Find approximate span for top k eigenvectors $\mathbf{Q}$.
2. Observe that $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{AQQ}^T) + \text{tr}(\mathbf{A}(\mathbf{I} - \mathbf{QQ}^T))$
3. Approximate $\tilde{P} = \text{tr}(\mathbf{A}(\mathbf{I} - \mathbf{QQ}^T))$ using Hutchinson's with ℓ vectors.
4. Return $\tilde{T} = \text{tr}(\mathbf{AQQ}^T) + \tilde{P}$.



The only error is from the estimator for $\text{tr}(\mathbf{A}(\mathbf{I} - \mathbf{QQ}^T))$, which will have much lower variance if $\|\mathbf{A}(\mathbf{I} - \mathbf{QQ}^T)\|_F \ll \|\mathbf{A}\|_F$.

Standard result in Randomized Numerical Linear Algebra:

Lemma (Sarlos 2006, Woodruff 2014)

If $\mathbf{S} \in \mathbb{R}^{n \times m}$ *is chosen with i.i.d.* ± 1 *entries, then* $\mathbf{Q} = \text{orth}(\mathbf{AS})$ *satisfies with probability* $(1-\delta)$*,*

$$\|\mathbf{A} - \mathbf{AQQ}^T\|_F \leq 2\|\mathbf{A} - \mathbf{A}_k\|_F,$$

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

Here $\mathbf{A}_k$ is the best k-rank approximation to $\mathbf{A}$, obtained by projecting onto $\mathbf{A}$'s top k eigenvectors.

Note that $\mathbf{Q}$ can be view as the result of running a single step of power method on $\mathbf{A}$.

For any PSD matrix $\mathbf{A}$:

$$\|\mathbf{A}-\mathbf{A}_k\|_F^2 = \sum_{i=k+1}^{n} \lambda_i^2 \leq \lambda_{k+1} \sum_{i=k+1}^{n} \lambda_i \leq \frac{1}{k}\operatorname{tr}(\mathbf{A}) \cdot \operatorname{tr}(\mathbf{A}).$$

So if $\left\|\mathbf{A}(\mathbf{I}-\mathbf{Q}\mathbf{Q}^T)\right\|_F \leq 2\left\|\mathbf{A}-\mathbf{A}_k\right\|_F$, then with high probability,

$$\left|\operatorname{tr}(\mathbf{A}(\mathbf{I}-\mathbf{Q}\mathbf{Q}^T)) - \tilde{P}\right| \leq \frac{1}{\sqrt{\ell}}\left\|\mathbf{A}(\mathbf{I}-\mathbf{Q}\mathbf{Q}^T)\right\|_F \leq \frac{1}{\sqrt{\ell}} \cdot \frac{2}{\sqrt{k}}\operatorname{tr}(\mathbf{A}).$$

Setting $\ell = k = O(1/\epsilon)$ gives error $\epsilon\operatorname{tr}(\mathbf{A})$ and thus:

$$\left|\operatorname{tr}(\mathbf{A}) - \tilde{T}\right| = \left|\operatorname{tr}(\mathbf{A}(\mathbf{I}-\mathbf{Q}\mathbf{Q}^T)) - \tilde{P}\right| \leq \epsilon\operatorname{tr}(\mathbf{A}).$$

Theorem (Final Result)

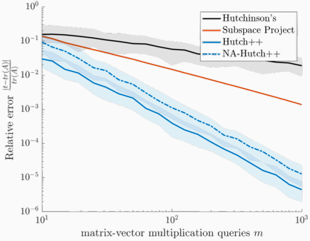
If $m = O\left(\frac{\log(1/\delta)}{\epsilon}\right)$ and $\mathbf{A}$ is PSD then with probability $(1-\delta)$, Hutch++ returns $\tilde{T}$ satisfying:

$$(1-\epsilon)\,\text{tr}(\mathbf{A}) \leq \tilde{T} \leq (1+\epsilon)\,\text{tr}(\mathbf{A})$$

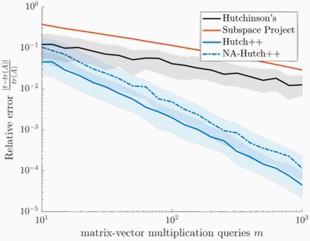
```
function T = hutchplusplus(A, m)
    S = 2*randi(2,size(A,1),m/3);
    G = 2*randi(2,size(A,1),m/3);
    [Q,~] = qr(A*S,0);
    G = G - Q*(Q'*G);
    T = trace(Q'*A*Q) + 1/size(G,2)*trace(G'*A*G);
end
```

This algorithm is adaptive, meaning that the choice of $\mathbf{x}_i$ depends on $\mathbf{A}\mathbf{x}_1 \ldots, \mathbf{A}\mathbf{x}_{i-1}$. We also have a non-adaptive method, NA-Hutch++ that achieves the same bound.

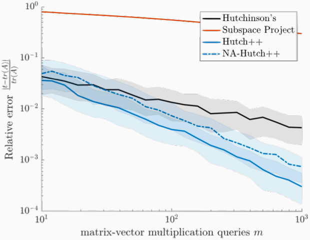
Results on synthetic matrix **A** with spectrum $\lambda_i = i^{-c}$ for different values of c.



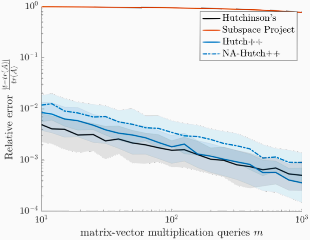
(a) Fast Eigenvalue Decay ($c = 2$)



(b) Medium Eigenvalue Decay ($c = 1.5$)

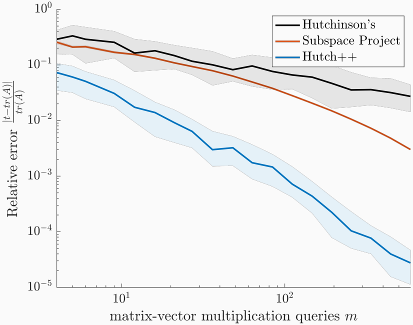


(c) Slow Eigenvalue Decay ($c = 1$)

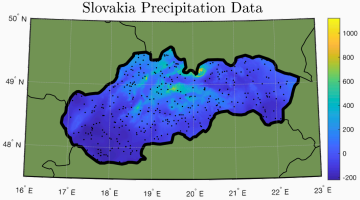


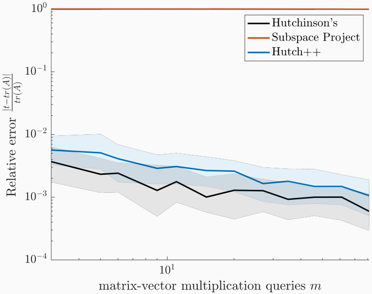
(d) Very Slow Eigenvalue Decay ($c = .5$)

If $\mathbf{B}$ is symmetric with eigendeposition $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, we let $f(\mathbf{B})$ denote $\mathbf{V}f(\mathbf{\Lambda})\mathbf{V}^T$, which means that f is applied entrywise to the diagonal matrix of eigenvalues, $\mathbf{\Lambda}$. Note that $\text{tr}(\mathbf{B}) = \sum_{i=1}^n f(\lambda_i)$.



$\mathbf{A} = \exp(\mathbf{B})$ for graph adjacency matrix $\mathbf{B}$ from linguistics application. $\text{tr}(\mathbf{A})$ is the well known Estrada Index or "natural connectivity".



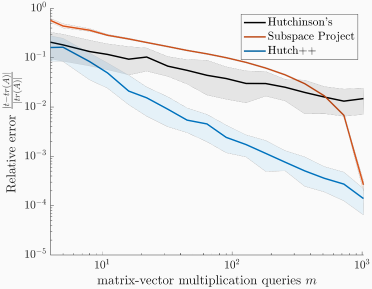


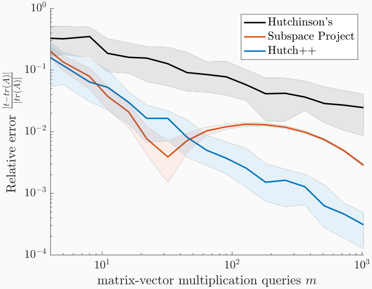
$A = \log(B + \lambda I)$ for kernel matrix B from Gaussian process regression. $\text{tr}(A) = \log\det(B)$, which is used in loglikelihood calculations.

Takeaway: For matrix functions that flatten B's spectrum, Hutchinson's estimator performs far better than the $O(1/\epsilon^2)$ bound predicts. Hutch++ will never perform much worse.

Hutch++ works well empirically for many non-PSD matrices.

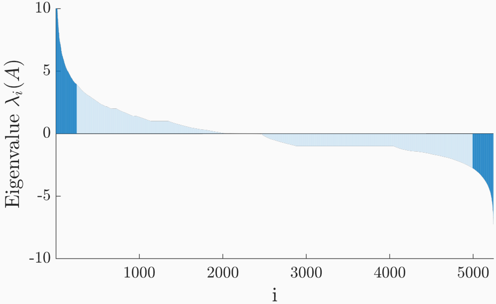
Let B is the (indefinite) adjacency matrix of an undirected graph G, $\text{tr}(B^3)$ is exactly equal to the number of triangles in G.





$A = B^3$ for arXiv.org citation network and Wikipedia voting network.

For non-PSD $\mathbf{A}$, the projection step, $\mathbf{A}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)$ approximately removes $\mathbf{A}$'s largest magnitude eigenvalues, which can still reduces variance substantially.



Spectrum of $\mathbf{A} = \mathbf{B}^3$ for arXiv.org citation network.

Theorem

Any algorithm that accesses a PSD matrix $\mathbf{A}$ *via matrix-vector multiplication queries* $\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_m$, *where* $\mathbf{x}_1, \ldots, \mathbf{x}_m$ *are possibly adaptively chosen vectors with integer entries in* $\{-2^b, ..., 2^b\}$, *needs*

$$m = \Omega\left(\frac{1}{\epsilon \cdot [b + \log(1/\epsilon)]}\right) \text{ queries}$$

to approximate $\text{tr}(\mathbf{A})$ *to multiplicative error* $(1 \pm \epsilon)$.

Reduction to 2-party multi-round communication problem. "Hard" input distribution will involve $\mathbf{A}$ with integer entries, which is why we need the bit complexity bound b.

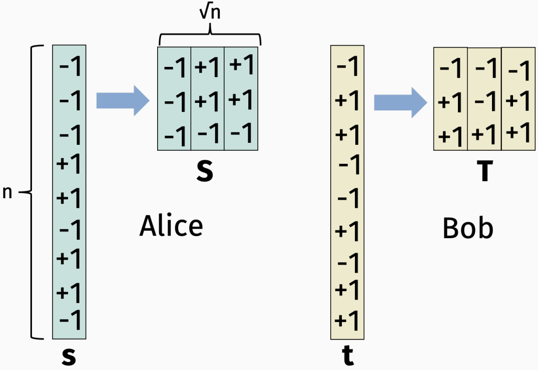
Problem (Gap Hamming)

Let Alice and Bob be communicating parties who hold vectors $\mathsf{s}, \mathsf{t} \in \{-1, 1\}^n$*, respectively. Must decide with few bits of communication if:*

$$\langle \mathsf{s}, \mathsf{t} \rangle \geq \sqrt{n} \qquad \textit{or} \qquad \langle \mathsf{s}, \mathsf{t} \rangle \leq -\sqrt{n}$$

Theorem (Chakrabarti, Regev 2012)

The randomized communication complexity for solving Problem 1 with probability $\geq 2/3$ *is* $\Omega(n)$ *bits.*



Let $\mathbf{Z} = \mathbf{S} + \mathbf{T}$ and $\mathbf{A} = \mathbf{Z}^T\mathbf{Z}$.

$$\text{tr}(\mathbf{A}) = \|\mathbf{Z}\|_F^2 = \|\mathbf{s} + \mathbf{t}\|_2^2 = 2n - 2\langle \mathbf{s}, \mathbf{t} \rangle.$$

So if Alice and Bob and estimate $\text{tr}(\mathbf{A})$ up to error $(1 \pm 1/\sqrt{n})$, then they will solve the Gap Hamming problem.

Claim: Alice and Bob can simulate any m query algorithm for estimating the trace of $\mathsf{A} = (\mathsf{S}+\mathsf{T})^T(\mathsf{S}+\mathsf{T})$ with $O\left(m\sqrt{n}(\log n + b)\right)$ bits of communication.

- Alice decides on x_1, sends to Bob with $\sqrt{n} \cdot \log(2^b)$ bits.
- Bob computes $\mathsf{T}\mathsf{x}_1$, sends to Alice with $\sqrt{n} \cdot \log(\sqrt{n}2^b)$ bits.
- Alice computes $(\mathsf{S}+\mathsf{T})\mathsf{x}_1$.
- Repeat to multiply $(\mathsf{S}+\mathsf{T})\mathsf{x}_1$ by $(\mathsf{S}+\mathsf{T})^T$
- Alice decides on x_2, process repeats m times.

So, by $\Omega(n)$ lower bound for Gap Hamming, we can't have m less than $\frac{\sqrt{n}}{\log n + b}$. Setting $\epsilon = 1/\sqrt{n}$ gives the result.

- **In progress:** Lower bounds for e.g. $\text{tr}(\mathbf{A}^3)$, $\text{tr}(\exp(\mathbf{A}))$, $\text{tr}(\mathbf{A}^{-1})$.
- What about (coarse) approximate matrix vector multiplications? We have some upcoming work on this related to spectral density estimation problems, but there's a lot to think about.
- Relates to model where we sample rows or columns of $\mathbf{A}$ (and implement things like SGD/SCD).
- Can we get conditional lower bounds for simple problems like triangle counting in a completely general computational model?

THANKS! QUESTIONS?