RandNLA: Randomization in Numerical Linear Algebra

Petros Drineas

Department of Computer Science
Purdue University

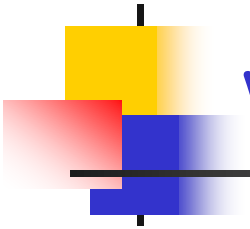
Google drineas

RandNLA:
Randomization in Numerical Linear Algebra:
estimating the Von Neumann entropy, the log-determinant, etc.

Petros Drineas

Department of Computer Science
Purdue University

Google drineas



Why RandNLA?

Randomization and sampling allow us to design provably accurate algorithms for problems that are:

- Massive

(matrices so large that can not be stored at all, or can only be stored in slow memory devices)

- Computationally expensive or NP-hard

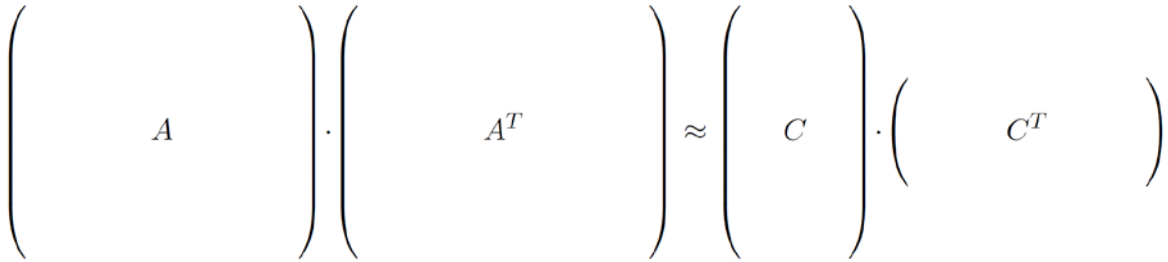
(combinatorial optimization problems, such as the Column Subset Selection Problem, sparse PCA, sparse approximations, k-means, etc.)

Basic RandNLA principles

Randomized algorithms

• By (carefully) sampling rows/columns/elements of a matrix, we can construct new, smaller matrices that are close to the original matrix (w.r.t. matrix norms) with high probability.

Example: Randomized Matrix Multiplication



• By preprocessing the matrix using "random projection" matrices, we can sample rows/columns much less carefully (uniformly at random) and still get nice bounds with high probability.

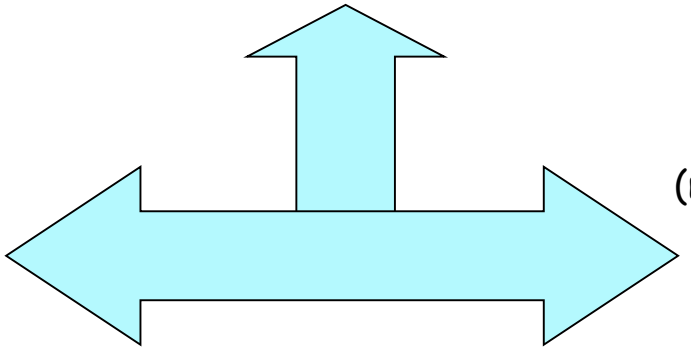
Matrix perturbation theory

• The resulting smaller matrices behave similarly (e.g., in terms of singular values and singular vectors) to the original matrices thanks to the norm bounds.

Interplay

Applications in BIG DATA

(Data Mining, Information Retrieval, Machine Learning, Bioinformatics, etc.)



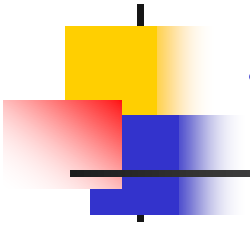
Theoretical Computer Science

Randomized and approximation algorithms

Applied Math

1. Numerical Linear Algebra
(matrix computations, perturbation theory)

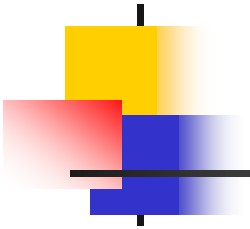
2. Probability theory
(esp. measure concentration for sums of random matrices)



IBM and RandNLA

IBM researchers have made numerous contributions to RandNLA. Here is a sample:

- *K. Clarkson & D. Woodruff:* matrix sketching and applications
- *H. Avron:* matrix sketching and preconditioning, trace estimators, etc.
- *C. Boutsidis:* randomized algorithms for matrix decompositions
- *D. Malioutov:* log-determinant estimators via Chebyshev polynomials
- *A. Zouzias & P. Kambadur:* log-determinant estimators via Taylor series
- *S. Ubaru & J. Chen:* approximating the trace of matrix functions
- *V. Kalantzis:* HPC implementations of subspace iteration & matrix decompositions



My talk today

- Estimating the Von Neumann Entropy (VNE) of a matrix

  (Kontopoulou, Dexter, Szpankowski, Grama & Drineas Transactions on Information Theory 2020 & ISIT 2018)

  - Express the entropy as the *trace* of a matrix function.
  - Use Taylor series/Chebyshev polynomials to approximate matrix functions.
  - Use a nice primitive to approximate the matrix trace.
  - Use matrix sketching for a special case of the VNE.
  - Works very well in practice (synthetic data); (the first two approaches) can be generalized to complex matrices.

- Estimating the log-determinant of a matrix

  (Boutsidis, Drineas, Kambadur, Kontopoulou & Zouzias LAA 2017)

Disclaimer

12:00 - 12:45 Rebooting Mathematics

Doron Zeilberger

What is Mathematics and What Should it Be?

Doron ZEILBERGER

world. All they care about is playing their artificial game, called [rigorous] *proving*, and observing their strict dogmas.

Beware of Greeks Carrying Mathematical Gifts

Once upon a time, a long time ago, mathematics was indeed a true science, and its practitioners devised methods for solving practical mathematical problems. The ancient Chinese, Indian, and Babylonian mathematicians were dedicated good scientists. Then came a major set-back, the Greeks!

Disclaimer

12:00 - 12:45 Rebooting Mathematics

Doron Zeilberger

What is Mathematics and What Should it Be?

Doron ZEILBERGER

world. All they care about is playing their artificial game, called [rigorous] *proving*, and observing their strict dogmas.

Beware of Greeks Carrying Mathematical Gifts

Once upon a time, a long time ago, mathematics was indeed a true science, and its practitioners devised methods for solving practical mathematical problems. The ancient Chinese, Indian, and Babylonian mathematicians were dedicated good scientists. Then came a major set-back, the Greeks!

Λυδία is Greek as well…

Von Neumann Entropy (VNE)

- Typically defined for quantum systems.
- Applications in Information Theory, Quantum Mechanics, etc.
- Extension of Gibbs/Shannon entropy concept in quantum mechanics; special case of the Rényi entropy (as $\alpha \to 1$).
- **History:** Described by Von-Neumann in *Mathematische Grundlagen der Quantenmechanik (1932).*
- **Fundamental notion:** Density Matrix.

Von Neumann Entropy: definition

- **Density matrix:** $\mathbf{R} \in \mathbb{R}^{n \times n}$

- It represents the so-called statistical mixture of the pure states of the system and has the form:

$$\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$$

- The n-dimensional vectors y_i represent the pure states of a system and are **pairwise orthogonal and normal**.
- The p_i correspond to the probabilities of each state and satisfy:

$$p_i > 0 \text{ and } \sum_{i=1}^{n} p_i = 1$$

Von Neumann Entropy: definition

- **Density matrix:** $\mathbf{R} \in \mathbb{R}^{n \times n}$

Singular Value Decomposition (SVD) of R

- It represents the so-called statistical mixture of the pure states of the system and has the form:

$$\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \boldsymbol{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$$

- The columns of the matrix Y are the (pairwise orthogonal and normal) vectors y_i.
- The density matrix R is symmetric positive definite, by definition!
- Its eigenvalues are the probabilities p_i (strictly positive, thus guaranteeing that R has full rank).

Von Neumann Entropy: definition

- **Density matrix:** $\mathbf{R} \in \mathbb{R}^{n \times n}$

- It represents the so-called statistical mixture of the pure states of the system and has the form:

$$\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$$

- Formal definition of Von Neumann Entropy:

$$\mathcal{H}(\mathbf{R}) = -\sum_{i=1}^{n} p_i \log p_i$$

- **Computational complexity:** $O(n^3)$, prohibitive for large n.

Expressing the VNE as a trace

(of a matrix function)

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$
- We can express the Von Neumann Entropy as a trace of a matrix function!
- Consider the function: $h(x) = x \log x \in \mathbb{R}$
- Applying a function on symmetric (more generally, Hermitian) matrices is simple: one applies the function on the eigenvalues of the matrix.

Expressing the VNE as a trace

(of a matrix function)

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$
- We can express the Von Neumann Entropy as a trace of a matrix function!

- Consider the function: $h(x) = x \log x \in \mathbb{R}$
- Applying a function on symmetric (more generally, Hermitian) matrices is simple: one applies the function on the eigenvalues of the matrix.
- Formally:

$$h(\mathbf{R}) \quad = \quad \mathbf{Y} h(\mathbf{\Sigma}_p) \mathbf{Y}^\top$$

- We apply the function h entry-wise to the diagonal matrix of the eigenvalues.

Expressing the VNE as a trace

(of a matrix function)

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$
- We can express the Von Neumann Entropy as a trace of a matrix function!
- Recall:

$$h(x) = x \log x \in \mathbb{R}$$

$$h(\mathbf{R}) = \mathbf{Y} h(\mathbf{\Sigma}_p) \mathbf{Y}^\top$$

- After 3-4 lines of elementary algebra, we get

$$\mathcal{H}(\mathbf{R}) = -\mathbf{Tr}\left(h(\mathbf{R})\right)$$

Von Neumann Entropy

Matrix Trace:
sum of its diagonal entries

Approximating the VNE

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y}\mathbf{\Sigma}_p\mathbf{Y}^\top \in \mathbb{R}^{n\times n}$

Recall: $h(x) = x\log x \in \mathbb{R}$

$h(\mathbf{R}) = \mathbf{Y}h(\mathbf{\Sigma}_p)\mathbf{Y}^\top$

$\mathcal{H}(\mathbf{R}) = -\mathbf{Tr}(h(\mathbf{R}))$

Von Neumann Entropy

Approach:

Step 1: Approximate the function h(x) using Taylor series or Chebyshev polynomials. Thus, we are approximating the VNE by the trace of a (matrix) polynomial function.

Step 2: Recall that the trace is a linear function (e.g., Tr(A+B) = Tr(A)+Tr(B)). Approximate the trace of each term in the resulting polynomial.

Approximating the VNE

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y}\mathbf{\Sigma}_p\mathbf{Y}^\top \in \mathbb{R}^{n\times n}$

Recall: $h(x) = x \log x \in \mathbb{R}$

$h(\mathbf{R}) = \mathbf{Y}h(\mathbf{\Sigma}_p)\mathbf{Y}^\top$

$\mathcal{H}(\mathbf{R}) = -\mathbf{Tr}(h(\mathbf{R}))$

Von Neumann Entropy

Advantages:

The approach is straight-forward.

It uses a very elegant (randomized) primitive for trace estimation.

Also uses "standard" properties of Taylor series and Chebyshev polynomials.

Approximating the VNE: Taylor series

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be a symmetric matrix whose eigenvalues all lie in the interval* $(-1, 1)$. *Then,*

$$\log(\mathbf{I}_n - \mathbf{A}) = -\sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k}.$$

Approximating the VNE: Taylor series

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be a symmetric matrix whose eigenvalues all lie in the interval* $(-1, 1)$. *Then,*

$$\log(\mathbf{I}_n - \mathbf{A}) = -\sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k}.$$

Let $\mathbf{R} \in \mathbb{R}^{n \times n}$ *be a symmetric positive definite matrix with unit trace and whose eigenvalues lie in the interval* $[\ell, u]$, *for some* $0 < \ell \leq u \leq 1$. *Then,*

$$\mathcal{H}(\mathbf{R}) = \log u^{-1} + \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{R}(\mathbf{I} - u^{-1}\mathbf{R})^k\right)}{k}.$$

Approximating the VNE: Taylor series

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be a symmetric matrix whose eigenvalues all lie in the interval* $(-1, 1)$. *Then,*

$$\log(\mathbf{I}_n - \mathbf{A}) = -\sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k}.$$

Let $\mathbf{R} \in \mathbb{R}^{n \times n}$ *be a symmetric positive definite matrix with unit trace and whose eigenvalues lie in the interval* $[\ell, u]$, *for some* $0 < \ell \leq u \leq 1$. *Then,*

$$\mathcal{H}(\mathbf{R}) = \log u^{-1} + \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{R}(\mathbf{I} - u^{-1}\mathbf{R})^k\right)}{k}.$$

0 → 1

For simplicity, we can just use u=1 (still need ℓ though).

Approximating the VNE: Taylor series

Let $\mathbf{R} \in \mathbb{R}^{n \times n}$ *be a symmetric positive definite matrix with unit trace and whose eigenvalues lie in the interval* $[\ell, 1]$*, for some* $0 < \ell \leq 1$*. Then,*

$$\mathcal{H}(\mathbf{R}) = \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{R}(\mathbf{I} - \mathbf{R})^k\right)}{k}.$$

- We will truncate the Taylor series by keeping the first m terms only.
- Computing the trace of $R(I-R)^k$ needs $O(n^3)$ time.
- Therefore, we will estimate the traces instead of computing them exactly.

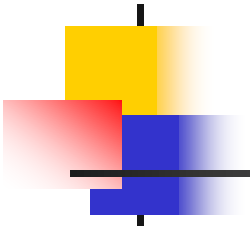
Trace Estimators

Given an n-by-n symmetric matrix A, we can estimate its trace as follows:

$$\widehat{\mathbf{Tr}(A)} = \frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^{\top}\, \mathbf{A}\, \mathbf{g}_i$$

The g_i are random n-dimensional vectors and could be:

- Random sign vectors (+1 or -1 with equal probability, often called Rademacher vectors), or
- Random Gaussian vectors (independent standard normal variables), etc.



Trace Estimators

Given an n-by-n symmetric matrix A, we can estimate its trace as follows:

$$\widehat{\mathbf{Tr}}(A) = \frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^{\top}\, \mathbf{A}\, \mathbf{g}_i$$

The g_i are random n-dimensional vectors and could be:

- Random sign vectors (+1 or -1 with equal probability, often called Rademacher vectors), or
- Random Gaussian vectors (independent standard normal variables), etc.

Notice that we only access A via matrix-vector products, which take $O(nnz(A))$ time to compute.

Larger values of s decrease the variance of the estimator.

Trace Estimators

(Avron & Toledo JACM 2011, Roosta & Ascher FOCM 2014)

Given an n-by-n symmetric matrix A, we can estimate its trace as follows:

$$\widehat{\mathbf{Tr}(A)} = \frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^{\top}\, \mathbf{A}\, \mathbf{g}_i$$

The estimator was pioneered by Hutchinson in 1990.

The original approach used random sign vectors; Hutchinson proved that the estimator works in expectation and has bounded variance.

Trace Estimators

(Avron & Toledo JACM 2011, Roosta & Ascher FOCM 2014)

Given an n-by-n symmetric matrix A, we can estimate its trace as follows:

$$\widehat{\mathbf{Tr}(A)} = \frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^{\top}\, \mathbf{A}\, \mathbf{g}_i$$

The estimator was pioneered by Hutchinson in 1990.

The original approach used random sign vectors; Hutchinson proved that the estimator works in expectation and has bounded variance.

Avron & Toledo JACM 2011: proposed the use of Gaussian random vectors g_i and provided relative error bounds for SPD matrices A.

Also proved that random sign vectors result in slightly weaker bounds; was later improved by Roosta & Ascher FOCM 2014.

Gaussian Trace Estimator

(Avron & Toledo JACM 2011, Roosta & Ascher FOCM 2014)

Let $\mathbf{A}$ *be an SPD matrix in* $\mathbb{R}^{n\times n}$, *let* $0 < \epsilon < 1$ *be an accuracy parameter, and let* $0 < \delta < 1$ *be a failure probability. Then for* $s = \lceil 20 \log(2/\delta)\epsilon^{-2} \rceil$, *with probability at least* $1 - \delta$,

$$\left| \mathbf{Tr}\left(\mathbf{A}\right) - \widehat{\mathbf{Tr}\left(A\right)} \right| \leq \epsilon \cdot \mathbf{Tr}\left(\mathbf{A}\right).$$

relative error approximation

Gaussian Trace Estimator

(Avron & Toledo JACM 2011, Roosta & Ascher FOCM 2014)

Let $\mathbf{A}$ *be an SPD matrix in* $\mathbb{R}^{n\times n}$, *let* $0 < \epsilon < 1$ *be an accuracy parameter, and let* $0 < \delta < 1$ *be a failure probability. Then for* $s = \lceil 20 \log(2/\delta)\epsilon^{-2} \rceil$, *with probability at least* $1 - \delta$,

$$\left| \mathbf{Tr}\left(\mathbf{A}\right) - \widehat{\mathbf{Tr}\left(A\right)} \right| \leq \epsilon \cdot \mathbf{Tr}\left(\mathbf{A}\right).$$

relative error approximation

A nice application of such estimators: We recently used such estimators to design and analyze randomized rounding algorithms for a Semi-Definite Programming relaxation of the Sparse Principal Components Analysis (SPCA) problem.

(Chowdhuri, Drineas, Woodruff & Zhu ArXiv 2020)

Approximating the VNE: algorithm

Input: $R \in \mathbb{R}^{n \times n}$, accuracy parameter $\epsilon > 0$, integer $m > 0$.

Output: $\widehat{\mathcal{H}(R)}$, the approximation to the $\mathcal{H}(R)$.

Create $s = \lceil 20 \log(2/\delta)/\epsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.

Compute $\widehat{\mathcal{H}(R)}$ as:

$$\widehat{\mathcal{H}(R)} = \frac{1}{s} \sum_{i=1}^{s} \sum_{k=1}^{m} \frac{g_i^\top R \left(\mathbf{I}_n - R\right)^k g_i}{k}.$$

Approximating the VNE: algorithm

Input: $R \in \mathbb{R}^{n \times n}$, accuracy parameter $\epsilon > 0$, integer $m > 0$.

Output: $\widehat{\mathcal{H}(R)}$, the approximation to the $\mathcal{H}(R)$.

Create $s = \lceil 20 \log(2/\delta)/\epsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.

Compute $\widehat{\mathcal{H}(R)}$ as:

approximate VNE

$$\widehat{\mathcal{H}(R)} = \frac{1}{s} \sum_{i=1}^{s} \sum_{k=1}^{m} \frac{g_i^\top R \left(\mathbf{I}_n - R\right)^k g_i}{k}.$$

exact VNE

$$\mathcal{H}(\mathbf{R}) = \sum_{k=1}^{\infty} \frac{\mathbf{Tr}\left(\mathbf{R}(\mathbf{I} - \mathbf{R})^k\right)}{k}$$

Approximating the VNE: algorithm

Input: $R \in \mathbb{R}^{n \times n}$, accuracy parameter $\epsilon > 0$, integer $m > 0$.
Output: $\widehat{\mathcal{H}(R)}$, the approximation to the $\mathcal{H}(R)$.

Create $s = \lceil 20 \log(2/\delta)/\epsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.
Compute $\widehat{\mathcal{H}(R)}$ as:

$$\widehat{\mathcal{H}(R)} = \frac{1}{s} \sum_{i=1}^{s} \sum_{k=1}^{m} \frac{g_i^\top R \left(\mathbf{I}_n - R\right)^k g_i}{k}.$$

Two sources of error:

(i) From the truncation of the Taylor series (only m terms are kept).

(ii) From the approximation of the trace using s random Gaussian vectors.

Approximating the VNE: algorithm

Input: $R \in \mathbb{R}^{n \times n}$, accuracy parameter $\epsilon > 0$, integer $m > 0$.
Output: $\widehat{\mathcal{H}(R)}$, the approximation to the $\mathcal{H}(R)$.

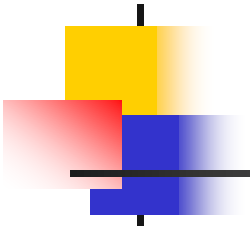
Create $s = \lceil 20 \log(2/\delta)/\epsilon^2 \rceil$ i.i.d random Gaussian vectors, $g_1, g_2, \ldots, g_s$.
Compute $\widehat{\mathcal{H}(R)}$ as:

$$\widehat{\mathcal{H}(R)} = \frac{1}{s} \sum_{i=1}^{s} \sum_{k=1}^{m} \frac{g_i^{\top} R \left(\mathbf{I}_n - R\right)^k g_i}{k}.$$

We analyze and bound them separately:

(i) The first one is mostly matrix algebra, and

(ii) the second one is (almost) immediate from Avron & Toledo (JACM 2011).



Final bound

Let $\mathbf{R}$ *be a density matrix such that all probabilities* p_i, $i = 1 \ldots n$ *satisfy* $0 < \ell \leq p_i$. *Let* u *be the output of the power method and let* $\widehat{\mathcal{H}(\mathbf{R})}$ *be the output of the Taylor-based algorithm on inputs* $\mathbf{R}$, m, *and* $\epsilon < 1$; *Then, with probability at least* $1 - 2\delta$,

$$\left| \widehat{\mathcal{H}(\mathbf{R})} - \mathcal{H}(\mathbf{R}) \right| \leq 2\epsilon \mathcal{H}(\mathbf{R}),$$

by setting $m = \left\lceil \frac{u}{\ell} \log(1/\epsilon) \right\rceil$.

Final bound

Let $\mathbf{R}$ *be a density matrix such that all probabilities* p_i, $i = 1 \ldots n$ *satisfy* $0 < \ell \leq p_i$. *Let* u *be the output of the power method and let* $\widehat{\mathcal{H}(\mathbf{R})}$ *be the output of the Taylor-based algorithm on inputs* $\mathbf{R}$, m, *and* $\epsilon < 1$; *Then, with probability at least* $1 - 2\delta$,

$$\left| \widehat{\mathcal{H}(\mathbf{R})} - \mathcal{H}(\mathbf{R}) \right| \leq 2\epsilon \mathcal{H}(\mathbf{R}),$$

by setting $m = \lceil \frac{u}{\ell} \log(1/\epsilon) \rceil$.

Computational Time

$$\mathcal{O}\left(\frac{u}{\ell} \cdot \frac{\log(1/\epsilon) \log(1/\delta)}{\epsilon^2} \cdot nnz(\mathbf{R}) + \log n \cdot \log(1/\delta) \cdot nnz(\mathbf{R}) \right)$$

Approximating the VNE: Chebyshev

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$

Recall: $h(x) = x \log x \in \mathbb{R}$

$h(\mathbf{R}) = \mathbf{Y} h(\mathbf{\Sigma}_p) \mathbf{Y}^\top$

$\mathcal{H}(\mathbf{R}) = -\mathbf{Tr}(h(\mathbf{R}))$

Von Neumann Entropy

Similar approach to the previous one:

(i) Approximate h(x) (and thus h(R)) using Chebyshev polynomials of the first kind.

(ii) Bound the error due to the approximation using "standard" theory.

(iii) Estimate the trace for each term of the polynomial.

Approximating the VNE: Chebyshev

- **Density matrix:** $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$

Recall:
$$h(x) = x \log x \in \mathbb{R}$$
$$h(\mathbf{R}) = \mathbf{Y} h(\mathbf{\Sigma}_p) \mathbf{Y}^\top$$

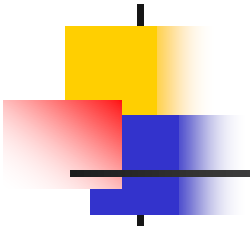
$$\mathcal{H}(\mathbf{R}) = -\mathbf{Tr}(h(\mathbf{R}))$$

Von Neumann Entropy

$$\widehat{\mathcal{H}(R)} = -\frac{1}{s} \sum_{i=1}^{s} g_i^\top f_m(R) g_i$$

$$f_m(x) = \sum_{t=0}^{m} \alpha_t \mathcal{T}_t(x)$$

- $T_t(x)$: Chebyshev polynomials of the first kind.
- Coefficients of the overall polynomial can be computed efficiently using Clenshaw's algorithm.



Final bound

Let $\mathbf{R}$ *be a density matrix such that all probabilities* p_i, $i = 1 \ldots n$ *satisfy* $0 < \ell \leq p_i$. *Let* u *be the output of the power method and let* $\widehat{\mathcal{H}(\mathbf{R})}$ *be the output of the Chebyshev-based algorithm on inputs* $\mathbf{R}$, m, *and* $\epsilon < 1$. *Then, with probability at least* $1 - 2\delta$,

$$\left| \widehat{\mathcal{H}(\mathbf{R})} - \mathcal{H}(\mathbf{R}) \right| \leq 3\epsilon \mathcal{H}(\mathbf{R}),$$

by setting $m = \sqrt{\frac{u}{2\epsilon\ell \ln(1/(1-\ell))}}$.

Final bound

Let $\mathbf{R}$ *be a density matrix such that all probabilities* p_i, $i = 1 \ldots n$ *satisfy* $0 < \ell \leq p_i$. *Let* u *be the output of the power method and let* $\widehat{\mathcal{H}(\mathbf{R})}$ *be the output of the Chebyshev-based algorithm on inputs* $\mathbf{R}$, m, *and* $\epsilon < 1$. *Then, with probability at least* $1 - 2\delta$,

$$\left| \widehat{\mathcal{H}(\mathbf{R})} - \mathcal{H}(\mathbf{R}) \right| \leq 3\epsilon \mathcal{H}(\mathbf{R}),$$

by setting $m = \sqrt{\frac{u}{2\epsilon\ell \ln(1/(1-\ell))}}$.

Computational Time

$$\mathcal{O}\left(\sqrt{\frac{u}{\ell \ln(1/(1-\ell))}} \cdot \frac{\ln(1/\delta)}{\epsilon^{2.5}} \cdot nnz(\mathbf{R}) + \ln(n) \cdot \ln(1/\delta) \cdot nnz(\mathbf{R}) \right)$$

Using random projections (outline)

Density matrix: $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \boldsymbol{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$

Both previous approaches necessitated that all $p_i > 0$. What if only $k << n$ of the p_i are strictly positive?

Using random projections (outline)

Density matrix: $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$

Both previous approaches necessitated that all $p_i > 0$. What if only $k << n$ of the p_i are strictly positive?

- We can use random projections (e.g., the Count-Min sketch by Clarkson & Woodruff STOC 2013) to approximate all the non-zero singular values of R up to relative error in time proportional to nnz(R).
- The above fact seems to be well-known in the RandNLA community and can be proven in various ways.
- This implies relative error approximations for each p_i for i=1…k.
- The final bound for the VNE approximation has an additive error term: **it is not a relative error bound.**

Using random projections (outline)

Density matrix: $\mathbf{R} = \sum_{i=1}^{n} p_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{Y} \mathbf{\Sigma}_p \mathbf{Y}^\top \in \mathbb{R}^{n \times n}$

Both previous approaches necessitated that all $p_i > 0$. What if only $k << n$ of the p_i are strictly positive?

Still open: what if we have $k < n$ strictly positive p_i with $k = \Omega(n)$?

Related work

Ubaru, Chen & Saad (SIMAX 2017): stochastic Lanczos quadrature approach to estimate the trace of **any** matrix function.

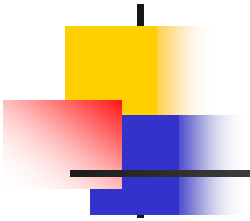
Works well *in theory* (comparable to the Chebyshev bounds, up to logarithmic factors) and *in practice*.

Musco, Netrappali, Sidford, Ubaru, & Woodruff (ITCS 2018): compute an approximate histogram of the matrix spectrum faster than matrix multiplication.

Relative error approximation to the VNE subject to a condition number assumption.

Running time does not depend on the condition number of the density matrix.

Mostly of theoretical interest (?).



Related work (cont'd)

Wihler, Bessire & Stefanov (Journal of Physics A 2014): approximates the entropy of a density matrix using Chebyshev polynomials.

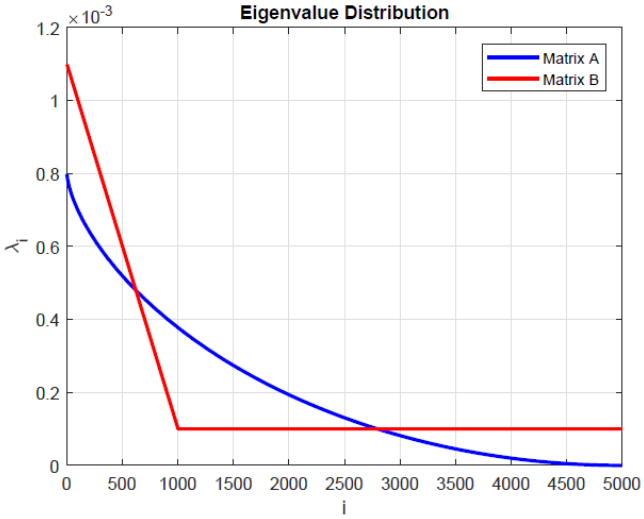
Weak bounds on the accuracy of the approach.

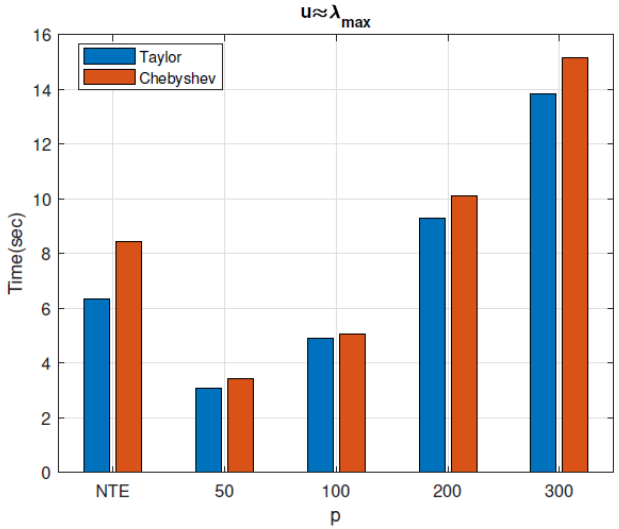
Choi, He, He & Shi (LAA 2020, CSIP 2018): approximating the VNE via low-rank approximations.

Experiments

Random density matrices of size $5,000 \times 5,000$

- ✓ **Matrix A:** exponentially decaying probabilities.
- ✓ **Matrix B:** $1,000$ linearly decaying probabilities.





Parameters

- ✓ Polynomial terms: $m = [5 : 5 : 30]$
- ✓ Gaussian vectors: $s = \{50, 100, 200, 300\}$
- ✓ Largest probability: $u \approx \lambda_{max}$

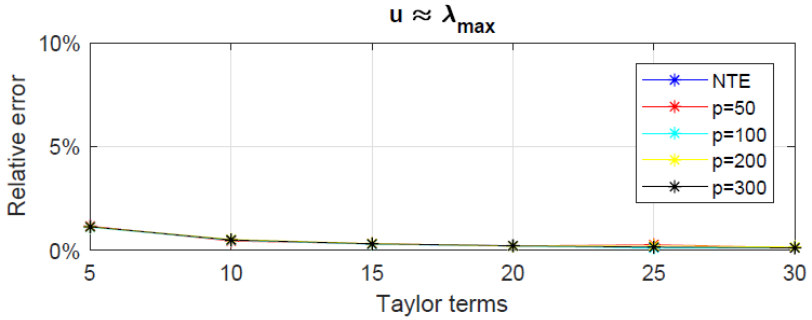
Notes

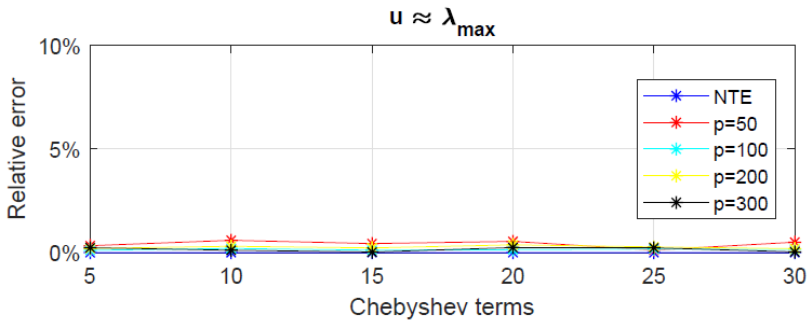
- Exact computation: 1.5 minutes.
- Approximation of λ_{max}: $<$ 1 second.

Experiments (cont'd)

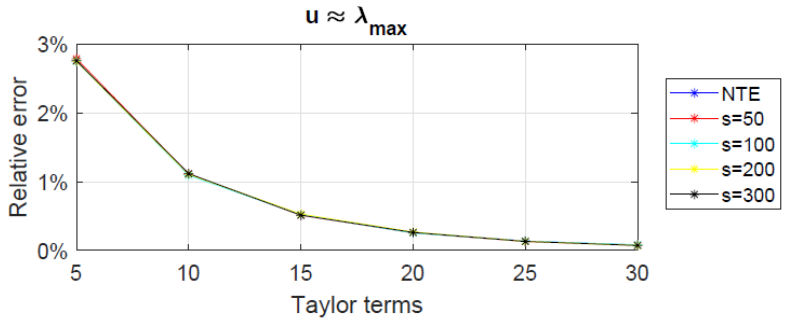
Parameters

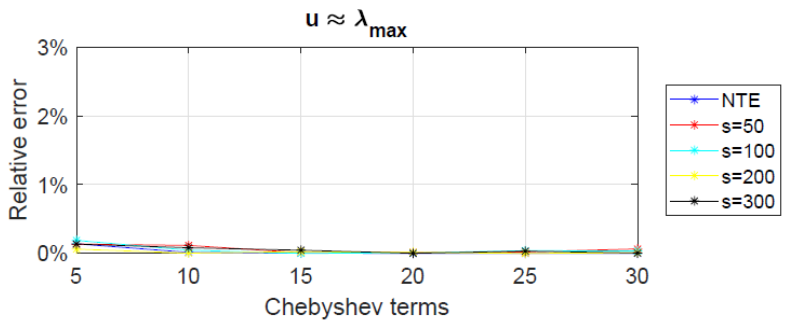
- ✓ Polynomial terms: $m = [5 : 5 : 30]$
- ✓ Gaussian vectors: $s = \{50, 100, 200, 300\}$
- ✓ Largest probability: $u \approx \lambda_{max}$





Matrix A



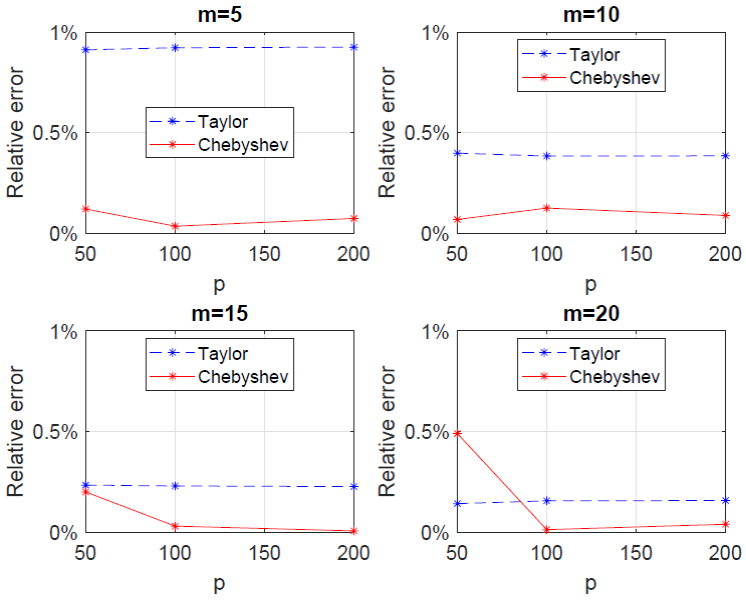


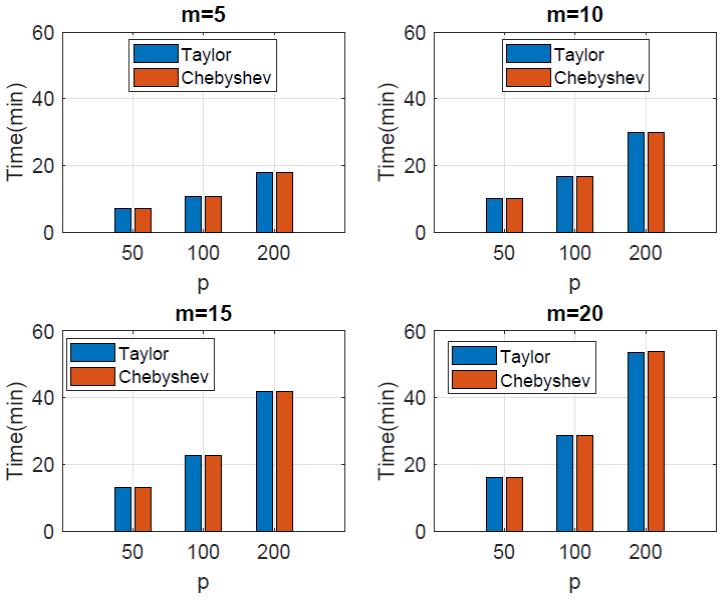
Matrix B

Experiments (cont'd)

Random density matrix of size $30,000 \times 30,000$

- ✓ Polynomial terms: $m = [5 : 5 : 20]$
- ✓ Largest probability: $u \approx \lambda_{max}$





Gaussian vectors: $s = [50 : 50 : 200]$

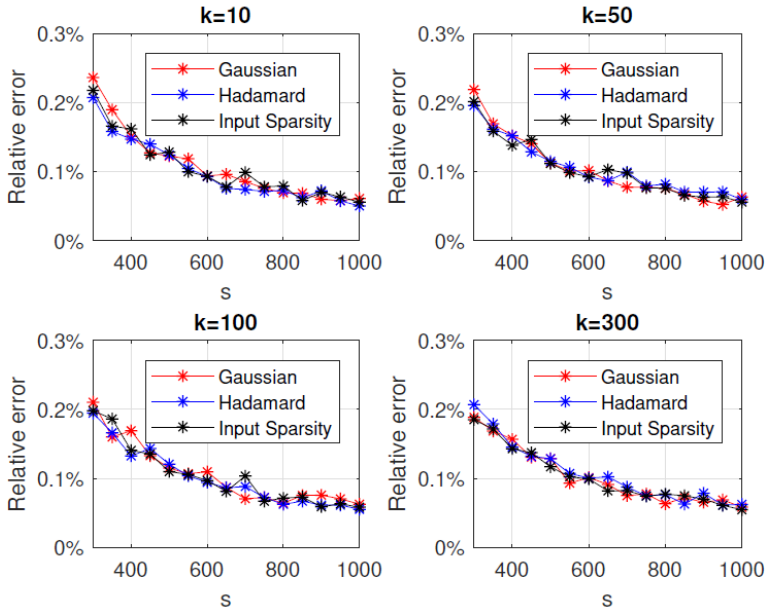
Gaussian vectors: $s = \{50, 100, 200\}$

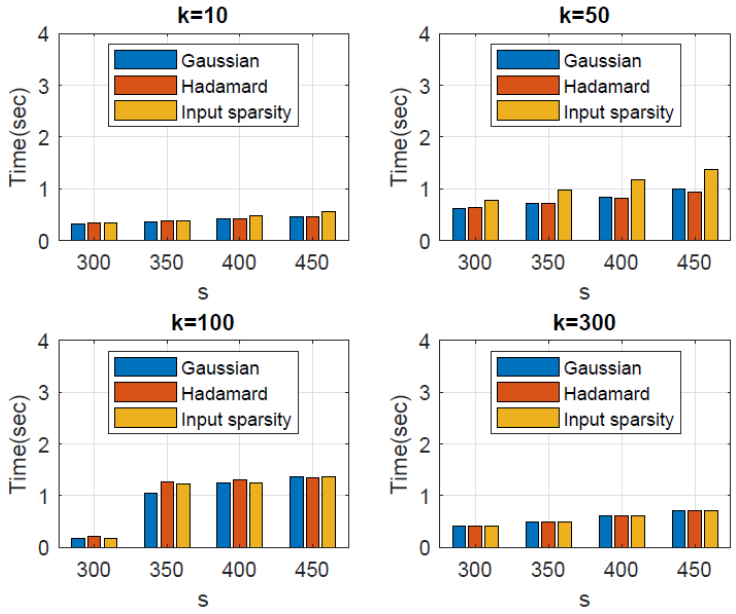
Notes

- Exact computation: 5.6 hours.
- Approximation of λ_{max}: 3.6 minutes.

Experiments (cont'd)

Matrix of size $4,096 \times 4,096$ and $k = \{10, 50, 100, 300\}$.





$s = \{400, 600, 800, 1000\}$

$s = [300 : 50 : 450]$

Notes

Exact computation for various k.

k	10	50	100	300
Time	1.5 sec	8 sec	15 sec	1 min

Log-determinant estimation

(Boutsidis, Drineas, Kambadur, Kontopoulou, Zouzias, LAA 2017)

Consider the following problem:

Given an n-by-n symmetric, positive definite matrix A compute its log determinant.

- All the eigenvalues of A are strictly positive and thus the determinant of A is strictly positive.
- The best exact algorithm for the above problem simply computes the determinant of A in cubic time and takes its logarithm.

Log-determinant estimation

Consider the following problem:

For the sake of simplicity, assume that the singular values of the n-by-n symmetric, positive definite matrix A lie in the interval $(\theta_1,1)$ with $0 < \theta_1 < 1$.

- The (upper bound) assumption can (and has) been removed, but the error bound worsens.

Let $C = I - A$; then,

$$\operatorname{logdet}(\mathbf{A}) = -\sum_{k=1}^{\infty} \frac{tr\left(\mathbf{C}^k\right)}{k}.$$

Proof

Let $C = I - A$, with A an SPD matrix with eigenvalues in the interval interval $(\theta_1,1)$ with $0 < \theta_1 < 1$. Then,

$$\operatorname{logdet}(\mathbf{A}) = -\sum_{k=1}^{\infty} \frac{\boldsymbol{tr}\left(\mathbf{C}^k\right)}{k}.$$

First, note that: $\operatorname{logdet}(\mathbf{A}) = \log\left(\prod_{i=1}^{n} \lambda_i(\mathbf{A})\right) = \sum_{i=1}^{n} \log(\lambda_i(\mathbf{A})) = \mathbf{tr}\left(\mathbf{log}\left[\mathbf{A}\right]\right)$

Proof

Let $C = I - A$, with A an SPD matrix with eigenvalues in the interval interval $(\theta_1,1)$ with $0 < \theta_1 < 1$. Then,

$$\operatorname{logdet}(\mathbf{A}) = -\sum_{k=1}^{\infty} \frac{tr\left(\mathbf{C}^k\right)}{k}.$$

First, note that: $\operatorname{logdet}(\mathbf{A}) = \log\left(\prod_{i=1}^{n} \lambda_i(\mathbf{A})\right) = \sum_{i=1}^{n} \log(\lambda_i(\mathbf{A})) = \mathbf{tr}\left(\mathbf{log}\left[\mathbf{A}\right]\right)$

Then,

$$\mathbf{tr}\left(\mathbf{log}\left[\mathbf{A}\right]\right) = \mathbf{tr}\left(\mathbf{log}\left[\mathbf{I}_n - (\mathbf{I}_n - \mathbf{A})\right]\right) = \mathbf{tr}\left(-\sum_{k=1}^{\infty} \frac{(\mathbf{I}_n - \mathbf{A})^k}{k}\right) = -\sum_{k=1}^{\infty} \frac{\mathbf{tr}\left(\mathbf{C}^k\right)}{k}.$$

We used the Taylor expansion for log (I-C); true for any symmetric matrix C with eigenvalues in the interval (-1,1):

$$\log(\mathbf{I}_n - \mathbf{C}) = -\sum_{k=1}^{\infty} \frac{\mathbf{C}^k}{k}.$$

Algorithmic implications

Let $C = I - A$, with A an SPD matrix with eigenvalues in the interval interval $(\theta_1,1)$ with $0 < \theta_1 < 1$. Then,

$$\operatorname{logdet}(\mathbf{A}) = -\sum_{k=1}^{\infty} \frac{tr\left(\mathbf{C}^k\right)}{k}.$$

We can approximate the logdet by approximating the above sum. Two tools:

- Truncate the above summation to include only the first m terms.
- Estimate the trace of C^k using random projections

This results in a fast approximation algorithm for the logdet of symmetric positive definite matrices (with eigenvalues in the interval $(\theta_1,1)$ with $0 < \theta_1 < 1$).

The algorithm

Input: $\mathbf{A} \in \mathbb{R}^{n \times n}$, accuracy parameter $\epsilon > 0$, integer $m > 0$.

Output: $\widehat{\log\det}(\mathbf{A})$, the approximation to the $\log\det(\mathbf{A})$.

1. $\mathbf{C} = \mathbf{I}_n - \mathbf{A}$
2. Create $p = \lceil 20 \log(2/\delta)/\epsilon^2 \rceil$ i.i.d random Gaussian vectors, $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_p$.
3. Compute $\widehat{\log\det}(\mathbf{A})$ with a truncated Taylor Series type randomized trace estimator as:

$$\widehat{\log\det}(\mathbf{A}) = -\sum_{k=1}^{m} \left(\frac{1}{p} \sum_{i=1}^{p} \mathbf{g}_i^\top \mathbf{C}^k \mathbf{g}_i \right).$$

Taylor series (truncated)

Trace estimator

Analysis

Two sources of error:

- First, the Taylor series was truncated to include only the first m terms; we need to bound the contribution of the remaining terms.
- Trace estimation introduces (relative) error as well.

After some algebra, we get:

$$\left|\widehat{\log\det}\left(\mathbf{A}\right) - \log\det\left(\mathbf{A}\right)\right| \leq \left(\varepsilon + \left(1 - \lambda_n\left(\mathbf{A}\right)\right)^m\right) \cdot \left|\log\det\left(\mathbf{A}\right)\right|$$

Our lower bound on the smallest eigenvalue of A (at least θ_1) implies:

$$\left|\widehat{\log\det}\left(\mathbf{A}\right) - \log\det\left(\mathbf{A}\right)\right| \leq \left(\varepsilon + \left(1 - \theta_1\right)^m\right) \cdot \left|\log\det\left(\mathbf{A}\right)\right|$$

Setting $m = \left\lceil \frac{1}{\theta_1} \cdot \log\left(\frac{1}{\varepsilon}\right) \right\rceil$ guarantees a relative error approximation.

Running time

The running time of the algorithm is

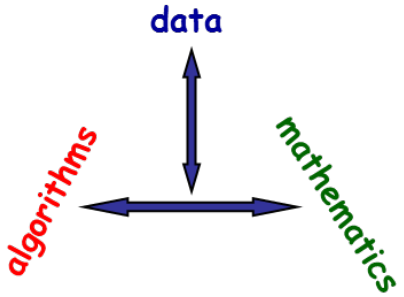
$$\mathcal{O}\left(\frac{\log(1/\varepsilon)\log(1/\delta)}{\varepsilon^2\theta_1}nnz(\mathbf{A})\right).$$

The running time depends on:

- the target accuracy (better accuracy implies higher running time),
- the failure probability (lower failure probability implies higher running time),
- the bound on the smallest eigenvalue (small eigenvalues imply higher running times), and
- the sparsity of the input matrix (denser matrices imply higher running times).

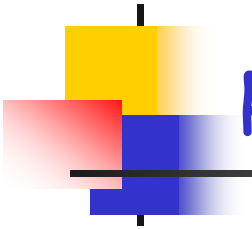
RandNLA events

"Randomization is arguably the most exciting and innovative idea to have hit linear algebra in a long time." (Avron et al. (2010) SISC)



- DIMACS Workshop on RandNLA, DIMACS, Sep 2019.
- RandNLA workshop, Simons Institute for the Theory of Computing, UC Berkeley, Foundations of Data Science, Sep 2018
- RandNLA course, PCMI Summer School on Mathematics of Data, Jul 2016
- Highlighted at the Workshops on Algorithms for Modern Massive Datasets (MMDS) 2006, 2008, 2010, 2012, 2014, and 2016.

  http://mmds-data.org/
- Gene Golub SIAM Summer School (G2S3), Δελφοί, Greece, June 2015
- Invited tutorial at SIAM ALA 2015
- RandNLA workshop in FOCS 2012



RandNLA review articles

P. G. Martinsson and J. A. Tropp, *Randomized Numerical Linear Algebra: Foundations & Algorithms*, Acta Numerica, 2020.

P. Drineas and M. W. Mahoney, *Lectures on Randomized Numerical Linear Algebra*, Amer. Math. Soc., 2018.

M. W. Mahoney and P. Drineas, *RandNLA: Randomized Numerical Linear Algebra*, Communications of the ACM, 2016.

D. Woodruff, *Sketching as a Tool for Numerical Linear Algebra*, Foundations and Trends in Theoretical Computer Science, 2014.

M. W. Mahoney, *Randomized Algorithms for Matrices and Data*, Foundations and Trends in Machine Learning, 2011.

N. Halko, P. G. Martinsson, J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*, SIAM Review, 2011.