An improved classical singular value transformation for quantum machine learning

Ainesh Bakshi
MIT

Ewin Tang
UC Berkeley

arXiv:2303.01492

Results

The problem: Computing matrix polynomials, $p(A)b$

Input

Hermitian matrix $A \in \mathbb{C}^{N \times N}$,
vector $b \in \mathbb{C}^N$,
degree-d polynomial[1] $p(x)$.

Normalization

$\|A\| \leq 1$;
$\|b\| \leq 1$;
$\|p(x)\|_{[-1,1]} \leq 1$.

Output

A vector $v \in \mathbb{C}^N$ such that
$\|v - p(A)b\| \leq \varepsilon$.

[1] We assume for this talk that the polynomial is either even or odd.

Main result

Input

Hermitian matrix $A \in \mathbb{C}^{N \times N}$,
vector $b \in \mathbb{C}^N$,
degree-d polynomial $p(x)$.

Normalization

$\|A\| \leq 1$;
$\|b\| \leq 1$;
$\|p(x)\|_{[-1,1]} \leq 1$.

Output

A vector $v \in \mathbb{C}^N$ such that
$\|v - p(A)b\| \leq \varepsilon$.

Main result

Input

Hermitian matrix $A \in \mathbb{C}^{N \times N}$,
vector $b \in \mathbb{C}^N$,
degree-d polynomial $p(x)$.

Normalization

$\|A\| \leq 1$;
$\|b\| \leq 1$;
$\|p(x)\|_{[-1,1]} \leq 1$.

Output

A vector $v \in \mathbb{C}^N$ such that
$\|v - p(A)b\| \leq \varepsilon$.

Naive classical algorithm

We can output $p(A)b$ in $O(dN^2)$ time.

Main result

Input

Hermitian matrix $A \in \mathbb{C}^{N \times N}$,
vector $b \in \mathbb{C}^N$,
degree-d polynomial $p(x)$.

Normalization

$\|A\| \leq 1$;
$\|b\| \leq 1$;
$\|p(x)\|_{[-1,1]} \leq 1$.

Output

A vector $v \in \mathbb{C}^N$ such that
$\|v - p(A)b\| \leq \varepsilon$.

Naive classical algorithm

We can output $p(A)b$ in $O(dN^2)$ time.

Prior classical algorithm [Chia, Gilyén, Li, Lin, T, Wang, '19]

After linear-time pre-processing, we can output v in $\widetilde{O}(d^{22}k^3/\varepsilon^6)$ time, where $k := \frac{\|A\|_F^2}{\|A\|^2}$ denotes stable rank.

Main result

Input

Hermitian matrix $A \in \mathbb{C}^{N \times N}$,
vector $b \in \mathbb{C}^N$,
degree-d polynomial $p(x)$.

Normalization

$\|A\| \leq 1$;
$\|b\| \leq 1$;
$\|p(x)\|_{[-1,1]} \leq 1$.

Output

A vector $v \in \mathbb{C}^N$ such that $\|v - p(A)b\| \leq \varepsilon$.

Naive classical algorithm

We can output $p(A)b$ in $O(dN^2)$ time.

Prior classical algorithm [Chia, Gilyén, Li, Lin, T, Wang, '19]

After linear-time pre-processing, we can output v in $\widetilde{O}(d^{22}k^3/\varepsilon^6)$ time, where $k := \frac{\|A\|_F^2}{\|A\|^2}$ denotes stable rank.

Our result

After linear-time pre-processing, we can output v in $\widetilde{O}(d^{11}k^2/\varepsilon^2)$ time.

Main result

Input

Hermitian matrix $A \in \mathbb{C}^{N \times N}$,
vector $b \in \mathbb{C}^N$,
degree-d polynomial $p(x)$.

Normalization

$\|A\| \leq 1$;
$\|b\| \leq 1$;
$\|p(x)\|_{[-1,1]} \leq 1$.

Output

A vector $v \in \mathbb{C}^N$ such that
$\|v - p(A)b\| \leq \varepsilon$.

Prior classical algorithm [Chia, Gilyén, Li, Lin, T, Wang, '19]

After linear-time pre-processing, we can output v in $\widetilde{O}(d^{22}k^3/\varepsilon^6)$ time, where $k := \frac{\|A\|_F^2}{\|A\|^2}$ denotes stable rank.

Our result

After linear-time pre-processing, we can output v in $\widetilde{O}(d^{11}k^2/\varepsilon^2)$ time.

Quantum algorithm [Gilyén, Su, Low, Wiebe, '18]

After linear-time pre-processing with a quantum-accessible RAM, we can output $|p(A/2)b\rangle$ in $O(d\sqrt{k})$ time.

Motivation and implications

The success of the quantum singular value transformation

arXiv.org > quant-ph > arXiv:2105.02859

Quantum Physics

[Submitted on 6 May 2021 (v1), last revised 20 Aug 2021 (this version, v3)]

A Grand Unification of Quantum Algorithms

John M. Martyn, Zane M. Rossi, Andrew K. Tan, Isaac L. Chuang

QSVT is a single framework comprising the three major quantum algorithms [Shor's algorithm, Grover's algorithm, and Hamiltonian simulation], thus suggesting a grand unification of quantum algorithms.

QSVT is the dominant technique for classical linalg speedups

Many proposals for quantum speedup in machine learning use QSVT+QRAM:

- Principal component analysis [Lloyd, Mohseni, Rebentrost ’14]
- Support vector machines [Rebentrost, Lloyd, Mohseni ’14]
- Discriminant analysis [Cong, Duan ’16]
- Recommendation systems [Kerenidis, Prakash ’17]
- k-means [Kerenidis, Landman, Luongo, Prakash ’18]
- Low-rank semidefinite program solving [Brandão et al. ’19]

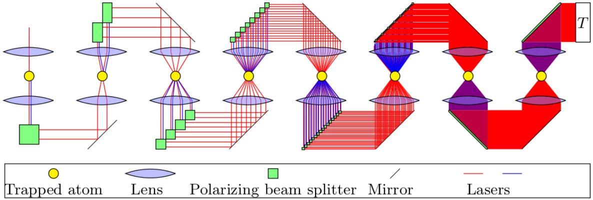

FIG. 9: Schematic of quantum optical fanout QRAM, almost exactly as shown in [GLM08a].

What is the classical version of QSVT?

What kind of speedups can QSVT achieve for linear algebraic tasks?

“No exponential speedup” results still leave hope

	Time to compute $p(A)b$
Quantum	$d\sqrt{k}$
Prior classical	$d^{22}k^3/\varepsilon^6$
Our result	$d^{11}k^2/\varepsilon^2$

“No exponential speedup” results still leave hope

	Time to compute $p(A)b$
Quantum	$d\sqrt{k}$
Prior classical	$d^{22}k^3/\varepsilon^6$
Our result	$d^{11}k^2/\varepsilon^2$

The prior result leaves open the possibility of large polynomial quantum speedups for low-rank QSVT.

“A practical quantum advantage for low-rank linear algebra, based on a theoretical high-degree polynomial speedup, remains a very viable possibility.” [KP22; KLLP19]

“No exponential speedup” results still leave hope

	Time to compute $p(A)b$
Quantum	$d\sqrt{k}$
Prior classical	$d^{22}k^3/\varepsilon^6$
Our result	$d^{11}k^2/\varepsilon^2$

The prior result leaves open the possibility of large polynomial quantum speedups for low-rank QSVT.

“A practical quantum advantage for low-rank linear algebra, based on a theoretical high-degree polynomial speedup, remains a very viable possibility.” [KP22; KLLP19]

Our work challenges this claim.

Conclusions

	Time to compute $p(A)b$
Quantum	$d\sqrt{k}$
Prior classical	$d^{22}k^3/\varepsilon^6$
Our result	$d^{11}k^2/\varepsilon^2$

Error dependence

There is no classical barrier at $1/\varepsilon^4$.

(Stable) rank dependence

The quartic gap, $\sqrt{k}$ vs k^2, may be "real".[2]

[2] Hastings, *Classical and Quantum Algorithms for Tensor Principal Component Analysis*

The algorithm

Algorithm

Preprocessing: Sketch A to $SAT \in \mathbb{C}^{s\times s}$ and b to $\hat{b} \in \mathbb{C}^s$ with

$$s = \tilde{\Theta}\Big(\frac{d^6 k}{\varepsilon^2}\Big) \text{ rows and columns;} \qquad \text{(linear time)}$$

Algorithm

Preprocessing: Sketch A to $SAT \in \mathbb{C}^{s\times s}$ and b to $\hat{b} \in \mathbb{C}^s$ with

$$s = \tilde{\Theta}\Big(\frac{d^6 k}{\varepsilon^2}\Big) \text{ rows and columns;} \qquad \text{(linear time)}$$

Iteration: Compute $u \approx q(SAT)\hat{b}$ for a polynomial $q(x)$ (think: $p(x)/x$);
Every iteration, sparsify $SAT \approx M$ to a matrix with

$$r = \tilde{\Theta}\Big(\frac{d^{10} k^2}{\varepsilon^2}\Big) \text{ entries;} \qquad (O(r) \text{ time} \times d \text{ iterations})$$

Algorithm

Preprocessing: Sketch A to $SAT \in \mathbb{C}^{s\times s}$ and b to $\hat{b} \in \mathbb{C}^s$ with

$$s = \tilde{\Theta}\Big(\frac{d^6 k}{\varepsilon^2}\Big) \text{ rows and columns;} \qquad \text{(linear time)}$$

Iteration: Compute $u \approx q(SAT)\hat{b}$ for a polynomial $q(x)$ (think: $p(x)/x$);
Every iteration, sparsify $SAT \approx M$ to a matrix with

$$r = \tilde{\Theta}\Big(\frac{d^{10} k^2}{\varepsilon^2}\Big) \text{ entries;} \qquad (O(r) \text{ time} \times d \text{ iterations})$$

Output: $v = (AT)u \approx p(A)b$.

Roadmap of improvements

Prior work [CGLLTW19]	$d^{22}k^3/\varepsilon^6$
Step 1: Using the polynomial structure	$d^{\mathbf{15}}k^{\mathbf{2}}/\varepsilon^{\mathbf{4}}$
Step 2: Tightening the stability analysis	$d^{\mathbf{11}}k^2/\varepsilon^4$
Step 3: Sparsifying the matrices	$d^{11}k^2/\varepsilon^{\mathbf{2}}$

Step 1: Using the polynomial structure

Prior work computes $f(M)$ for M an $s \times s$ matrix with $s = O(\mathrm{poly}(d) k/\varepsilon^2)$, picking up a k^3/ε^6 dependence.

We use that p is a polynomial to compute $p(A)b$ via an iterative algorithm.

Evaluating polynomials numerically stably

Input: Polynomial p with $\|p\|_{[-1,1]} \leq 1$ and $x \in [-1,1]$, except multiplication by x is *approximate*:

$$x \odot y \in ((1-\delta)x \cdot y, (1+\delta)x \cdot y)$$

Output: $p(x)$ up to small error.

Evaluating polynomials numerically stably

Input: Polynomial p with $\|p\|_{[-1,1]} \leq 1$ and $x \in [-1, 1]$, except multiplication by x is *approximate*:

$$x \odot y \in ((1-\delta)x \cdot y, (1+\delta)x \cdot y)$$

Output: $p(x)$ up to small error.

computing $p(x)$	$\iff$	computing $p(A)b$
error in multiplying by x	$\iff$	per-iteration sketching error of A
scalar stability bounds	$\iff$	matrix error bounds

Proving tight stability bounds for this appears to be open.

Step 2: Tighter stability analysis of the Clenshaw iteration

Suppose we have a polynomial[3] $p(x) = \sum_{k=0}^{d} a_k T_k(x)$. Given x, we compute $p(x)$ with the *Clenshaw recurrence* [Clenshaw, '55]:

$$\begin{aligned} \widetilde{q}_{d+1}, \widetilde{q}_{d+2} &= 0; \\ \widetilde{q}_k &= 2x \odot \widetilde{q}_{k+1} - \widetilde{q}_{k+2} + a_k; \\ \text{then } p(x) \approx \widetilde{p}(x) &= \tfrac{1}{2}(a_0 + \widetilde{q}_0 - \widetilde{q}_2) \end{aligned}$$

[3] $T_k(x)$ is the degree-k Chebyshev polynomial of the first kind.

Step 2: Tighter stability analysis of the Clenshaw iteration

Suppose we have a polynomial[3] $p(x) = \sum_{k=0}^{d} a_k T_k(x)$. Given x, we compute $p(x)$ with the *Clenshaw recurrence* [Clenshaw, '55]:

$$\begin{aligned} \widetilde{q}_{d+1}, \widetilde{q}_{d+2} &= 0; \\ \widetilde{q}_k &= 2x \odot \widetilde{q}_{k+1} - \widetilde{q}_{k+2} + a_k; \\ \text{then } p(x) \approx \widetilde{p}(x) &= \tfrac{1}{2}(a_0 + \widetilde{q}_0 - \widetilde{q}_2) \end{aligned}$$

Prior analysis [Musco, Musco, Sidford '18]:

$$|p(x) - \widetilde{p}(x)| = O(\delta d^3).$$

[3] $T_k(x)$ is the degree-k Chebyshev polynomial of the first kind.

Step 2: Tighter stability analysis of the Clenshaw iteration

Suppose we have a polynomial[3] $p(x) = \sum_{k=0}^{d} a_k T_k(x)$. Given x, we compute $p(x)$ with the *Clenshaw recurrence* [Clenshaw, '55]:

$$\begin{aligned} \widetilde{q}_{d+1}, \widetilde{q}_{d+2} &= 0; \\ \widetilde{q}_k &= 2x \odot \widetilde{q}_{k+1} - \widetilde{q}_{k+2} + a_k; \\ \text{then } p(x) \approx \widetilde{p}(x) &= \tfrac{1}{2}(a_0 + \widetilde{q}_0 - \widetilde{q}_2) \end{aligned}$$

Prior analysis [Musco, Musco, Sidford '18]:

$$|p(x) - \widetilde{p}(x)| = O(\delta d^3).$$

Our improvement:

$$|p(x) - \widetilde{p}(x)| = O(\delta d^2 \log(d))$$

[3] $T_k(x)$ is the degree-k Chebyshev polynomial of the first kind.

Thank you!

Future directions

- Is the quartic quantum speedup for spectral algorithms “real”?
- Is it possible to prove instance-specific stability for the Clenshaw iteration? Is the Clenshaw iteration optimally stable?
- Do these improvements extend to, e.g. low-rank SDP solving?

Step 3: Sparsifying the matrices

To compute $p(M)v$, we lift the Clenshaw recurrence to matrices and vectors:

$$\begin{aligned} q_{d+1}, q_{d+2} &= \vec{0}; \\ q_k &= 2Mq_{k+1} - q_{k+2} + a_k v; \\ p(M)v &= \tfrac{1}{2}(a_0 v + q_0 - q_2) \end{aligned}$$

We pay $O(1/\varepsilon^4)$ because we are working with an $s \times s$ matrix M with $s = O(1/\varepsilon^2)$.

We are allowed to incur $\varepsilon\|q_{k+1}\|$ error each iteration. Can we sparsify $M \approx \widetilde{M}$ to improve this dependence?

Importance sampling for entry-wise sparsification

Consider an $s \times s$ matrix M.

Sparsifying to operator norm error [Drineas, Zouzias, '10]

For $\|\widetilde{M} - M\| \leq \varepsilon$, a matrix with $\widetilde{O}(s\|M\|_F^2/\varepsilon^2) \approx 1/\varepsilon^4$ entries suffice.

Importance sampling for entry-wise sparsification

Consider an $s \times s$ matrix M.

Sparsifying to operator norm error [Drineas, Zouzias, '10]

For $\|\widetilde{M} - M\| \leq \varepsilon$, a matrix with $\widetilde{O}(s\|M\|_F^2/\varepsilon^2) \approx 1/\varepsilon^4$ entries suffice.

Our estimator

- Sample the index (i, j) with probability $M_{i,j}^2/\|M\|_F^2$ and take $\widetilde{M}$ to be the unbiased estimator.

This estimator concentrates poorly, but we show that it is ε-accurate in $O(1)$ directions, and 0.1-accurate in the rest.

Importance sampling for entry-wise sparsification

Consider an $s \times s$ matrix M.

Sparsifying to operator norm error [Drineas, Zouzias, '10]

For $\|\widetilde{M} - M\| \leq \varepsilon$, a matrix with $\widetilde{O}(s\|M\|_F^2/\varepsilon^2) \approx 1/\varepsilon^4$ entries suffice.

Our estimator

- Sample the index (i, j) with probability $M_{i,j}^2/\|M\|_F^2$ and take $\widetilde{M}$ to be the unbiased estimator.

This estimator concentrates poorly, but we show that it is ε-accurate in $O(1)$ directions, and 0.1-accurate in the rest.

In our context, for sparsifying matrices in "product expressions", $O(d^4\|M\|_F^2(s + 1/\varepsilon^2))$ samples suffice.