On quantum linear algebra for machine learning

Ewin Tang
University of Washington

January 25, 2022

Guiding question: when can quantum computing speed up classical linear algebra tasks?

This question:

- is natural, in light of QSVT;
- reflects a hope for exponential speedup that drives some of the hype behind quantum computing.

Talk overview

We will discuss three lines of work in quantum computing in broad terms:

1. QSVT, which unifies many quantum algorithms by viewing them from a linear algebraic perspective
2. Quantum linear algebra algorithms, which use QSVT as a framework to solve linear algebraic tasks
3. “Dequantized” classical algorithms, which can perform “similarly” to quantum linear algebra algorithms, under certain conditions

Throughout, we will assume a fault-tolerant circuit-based quantum computer (and later on, even more hardware).

Quantum singular value transformation

[Gilyén, Su, Low, Wiebe – Quantum singular value transformation and beyond][1]

[1] See also Yulong Dong's talk!

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

"Quantum algorithms offer significant speedups over their classical counterparts for a variety of problems. The strongest arguments for this advantage are borne by algorithms for quantum search, quantum phase estimation, and Hamiltonian simulation, which appear as subroutines for large families of composite quantum algorithms. A number of these quantum algorithms were recently tied together by a novel technique known as the quantum singular value transformation (QSVT) [....] This overview illustrates how **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.**"

How to apply a matrix to a quantum state

Consider a matrix $A \in \mathbb{C}^{N \times N}$ and a vector $v \in \mathbb{C}^N$. Consider encoding a (nonzero) vector $v \in \mathbb{C}^N$ into the amplitudes of a quantum state:

$$|v\rangle := \frac{1}{\|v\|} \sum_{i=1}^{N} v_i \, |i\rangle$$

and we **want the state** $|Av\rangle$.

If A is unitary, then we could try to find a circuit implementing it and simply apply it to $|v\rangle$.

$$|v\rangle \; \boxed{A} \; |Av\rangle$$

What about if A isn't unitary?

How to apply a matrix to a quantum state

Consider a matrix $A \in \mathbb{C}^{N \times N}$ and a vector $v \in \mathbb{C}^N$. Consider encoding a (nonzero) vector $v \in \mathbb{C}^N$ into the amplitudes of a quantum state:

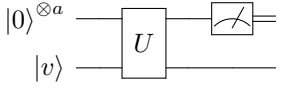
$$|v\rangle := \frac{1}{\|v\|} \sum_{i=1}^{N} v_i \, |i\rangle$$

and we **want the state** $|Av\rangle$.

If A isn't unitary, we can still try to find a circuit U such that (rescaling so that $\|A\| \leq 1$),

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \iff A = (\langle 0|^{\otimes a} \otimes I) U (|0\rangle^{\otimes a} \otimes I).$$

Then, we can do



If the measurement reads $|0\rangle^{\otimes a}$, the bottom state is $|Av\rangle$. This happens with probability $\frac{\|Av\|^2}{\|v\|^2} \leq \|A\|^2$.

Block-encodings formalizes these “linear algebraic” unitaries

Definition

We say we have a *block-encoding* of a matrix A with $\|A\| \leq 1$ if we can efficiently apply U *and* U^{-1}, where U is a unitary matrix satisfying

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix} \iff A = (\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I).$$

Technical note: assuming U is easy to implement, the “cost” parameter of the block-encoding is $1/\|A\|$.

The fundamental theorem of block-encodings (aka “quantum singular value transformation”)

Theorem

Given a block-encoding of (Hermitian) A, we can get a block-encoding of $\frac{1}{2}p(A)$,[2] where p is a degree-d polynomial satisfying

$$\left| \max_{x \in [-1,1]} p(x) \right| \leq 1.$$

The size of the quantum circuit implementing $\frac{1}{2}p(A)$ blows up by only a factor of d.

[2] If p is even or odd, the factor of two can be dropped, and the result can be generalized to non-Hermitian A.

Example: applying QSVT to matrix inversion [HHL09]

Suppose we have a block-encoding of $A \in \mathbb{C}^{n \times n}$ such that $\|A\| \leq 1$ and $\|A^{-1}\| \leq \kappa$, and copies of the quantum state $|b\rangle$.

We *want* $|A^{-1}b\rangle = |\phi(A)b\rangle$ where $\phi(x) = \frac{1}{\kappa x}$.

1. Use QSVT to create a block-encoding of $p(A)$, where

$$\Big|p(x) - \phi(x)\Big| \leq \varepsilon \text{ for all } x \in \Big[-1, -\frac{1}{\kappa}\Big] \cup \Big[\frac{1}{\kappa}, 1\Big]$$

   and has degree $O(\kappa \log(\frac{1}{\varepsilon}))$;
2. Apply $p(A)$ to $|b\rangle$ and post-select to get $|p(A)b\rangle \approx_\varepsilon |\phi(A)b\rangle = |A^{-1}b\rangle$

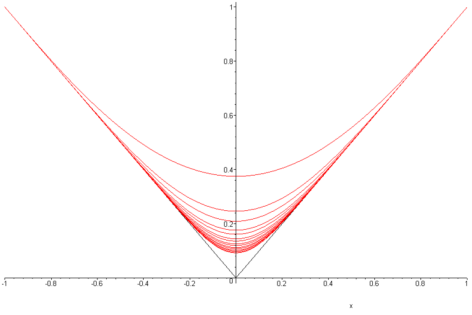
Any Lipschitz function can be approximated by a low-degree polynomial

Jackson's inequality

Let $f : [-1, 1] \to [-1, 1]$ be L-Lipschitz, i.e. $|f(x) - f(y)| \leq L|x - y|$ for all $x, y \in [-1, 1]$.

Then there exists a polynomial p of degree $\leq \frac{L}{\varepsilon}$ such that

$$\sup_{x \in [-1,1]} |f(x) - p(x)| \leq C\varepsilon$$



Approximation of $|x|$ by polynomials.[3]

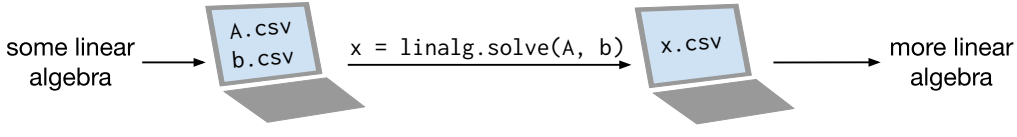
So, QSVT can be used to perform a wide range of operations in $O(\log N)$ time, *if* we have the right block-encodings.

[3] Source: Theon, Wikipedia Commons

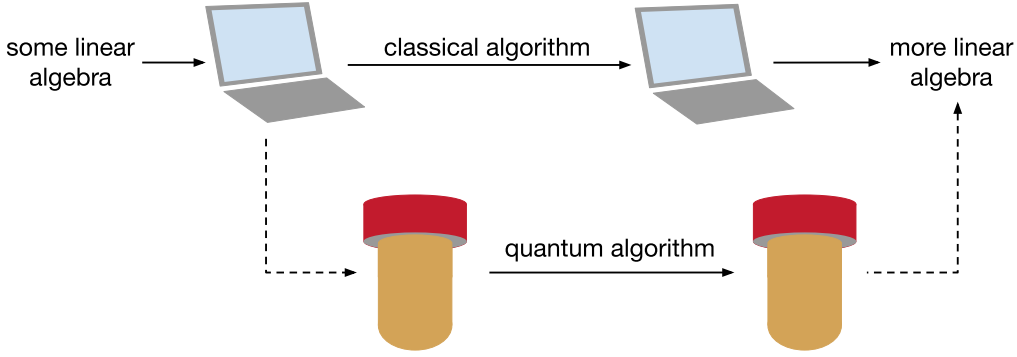
There are major barriers when trying to apply QSVT to machine learning tasks.

[Aaronson – Read the fine print]

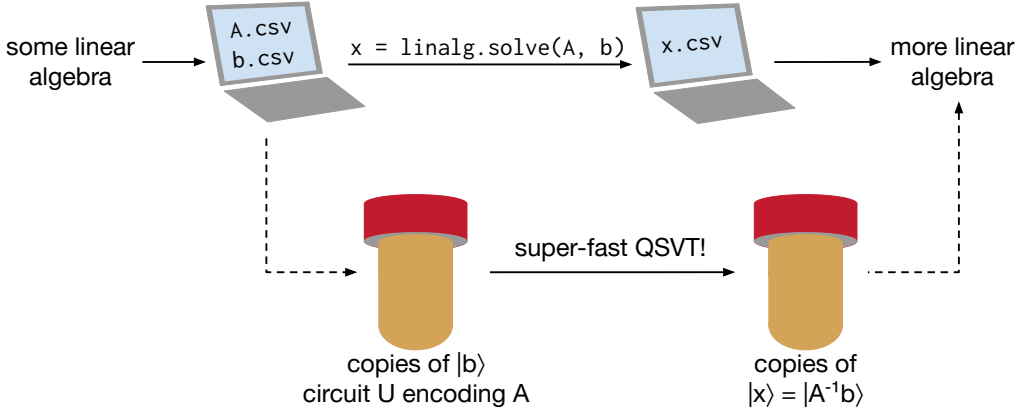
A typical application of quantum linear algebra to machine learning



A typical application of quantum linear algebra to machine learning



Example: applying QSVT to matrix inversion [HHL09]



1. Encode input matrices as block-encodings; encode input vectors as quantum states;
2. Use QSVT to compute an algebraic expression of the input;
3. Extract information from the output (say, an estimator of a desired value).

Hope: this gives exponential speedups.

Creating block-encodings

Consider some $A \in \mathbb{C}^{N \times N}$. We can get a block-encoding to

- A/s if it is s-*sparse with efficiently computable, bounded entries*
- $A/\|A\|_{\mathrm{F}}$ if it is in *quantum random access memory*[4]

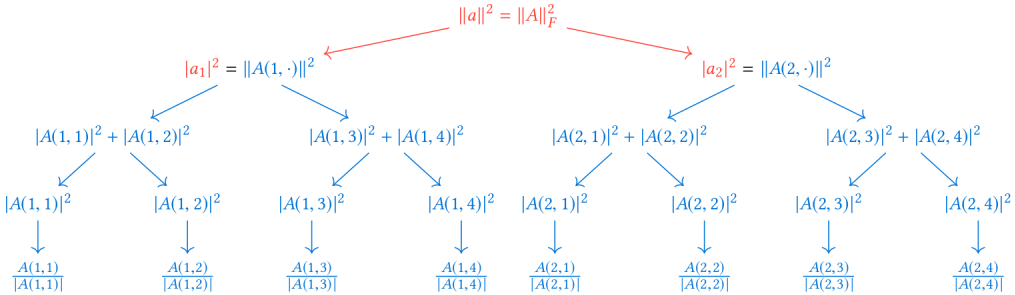
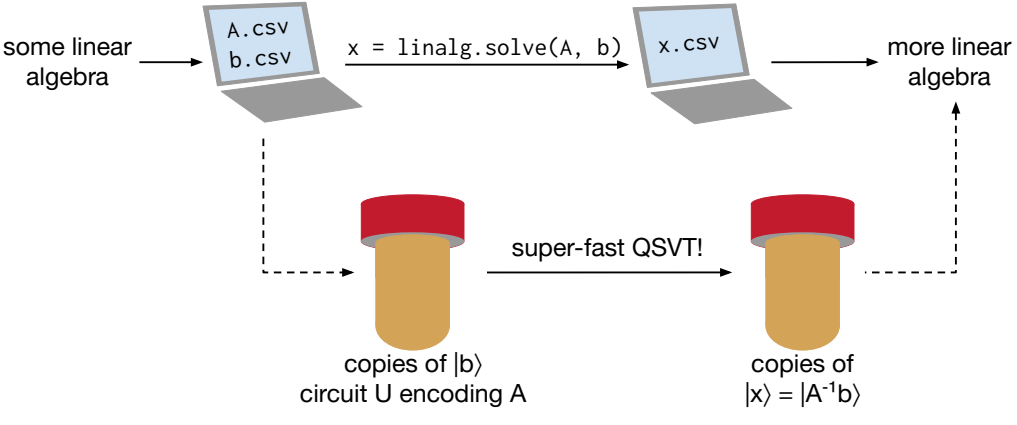


Figure 1: Dynamic data structure for $A \in \mathbb{C}^{2\times 4}$. We compose the data structure for a with the data structure for A's rows.

[4] $\|A\|_{\mathrm{F}} := (\sum_{i,j=1}^{N} |A_{ij}|^2)^{\frac{1}{2}}$

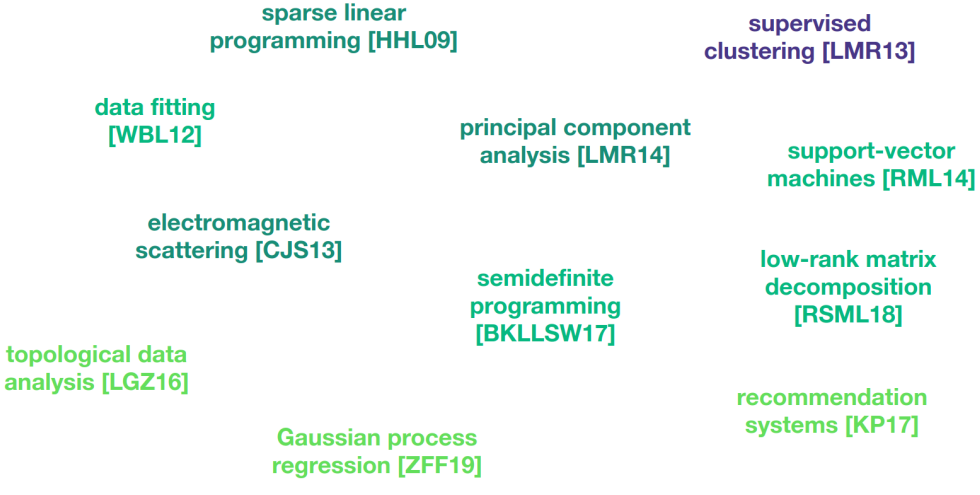
The input problem and the output problem



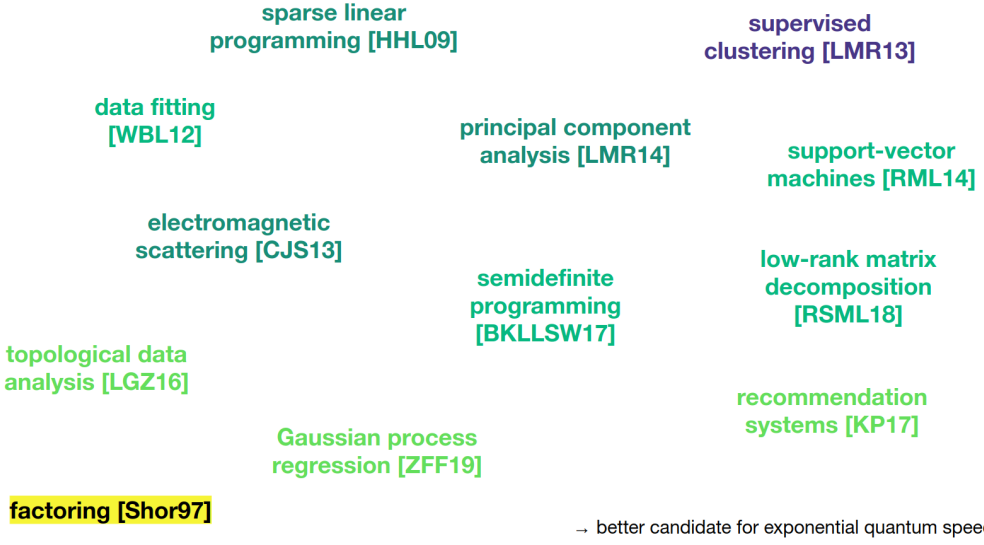
The “dequantized algorithms” problem

[Chia, Gilyén, Li, Lin, T, Wang – Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning]

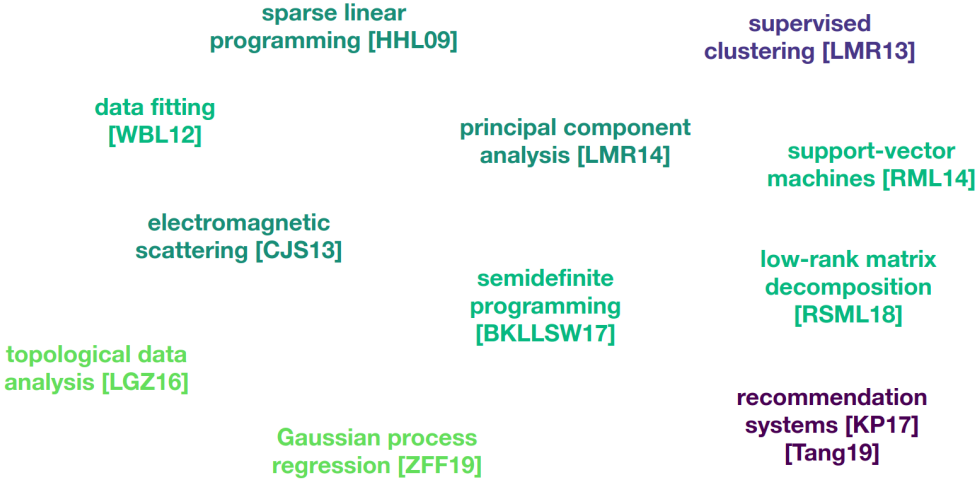
Landscape: exponential speedups in quantum machine learning



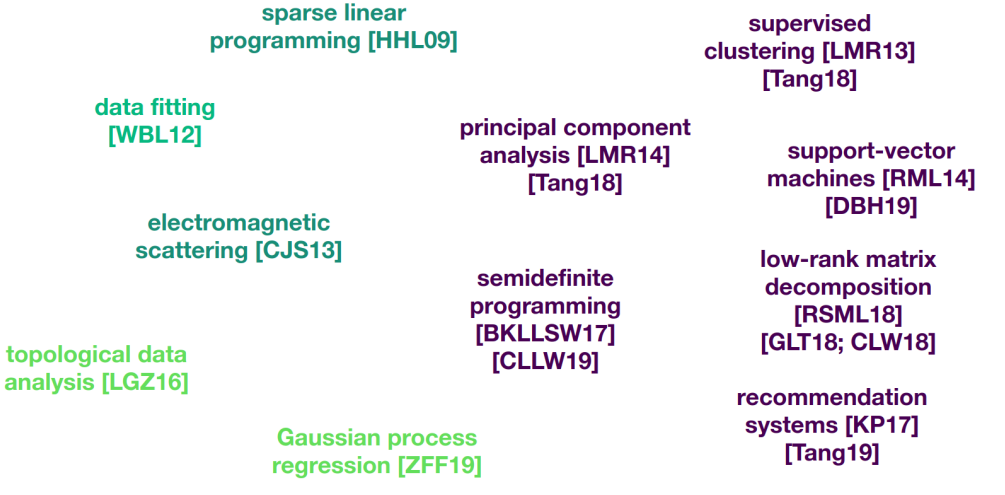
Landscape: exponential speedups in quantum machine learning

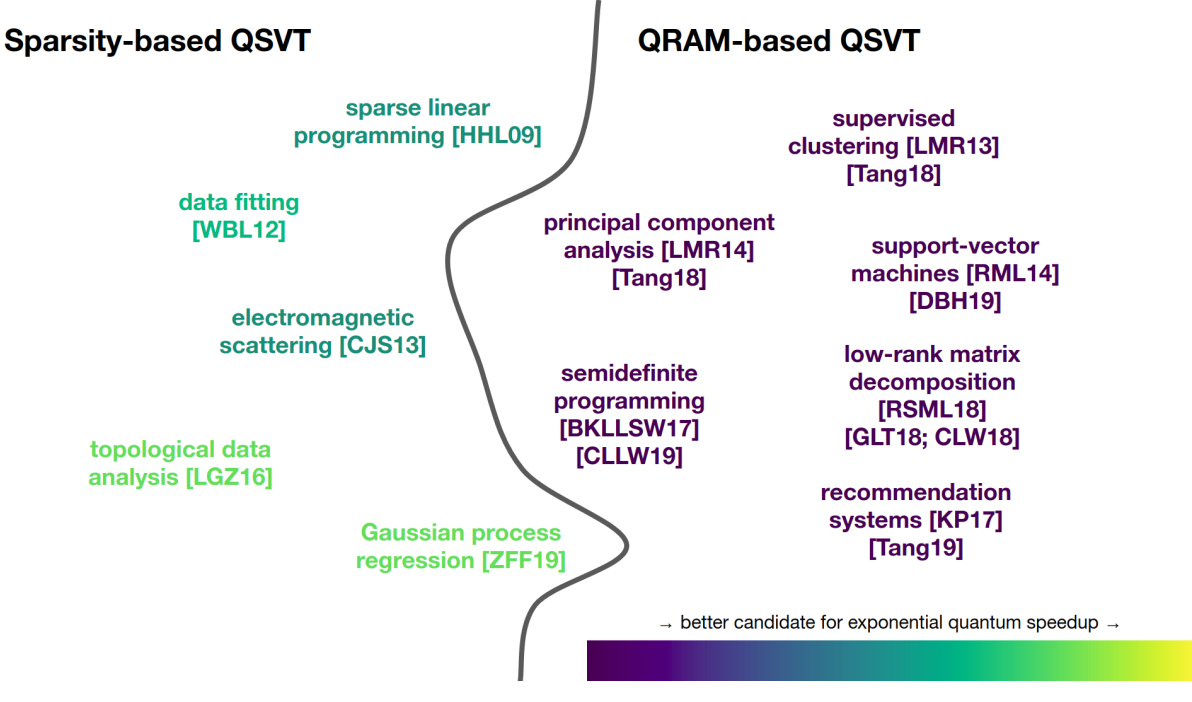


Landscape: exponential speedups in quantum machine learning



Landscape: exponential speedups in quantum machine learning




Sparsity-based QSVT
QRAM-based QSVT
sparse linear programming [HHL09]
data fitting [WBL12]
electromagnetic scattering [CJS13]
topological data analysis [LGZ16]
Gaussian process regression [ZFF19]
semidefinite programming [BKLLSW17] [CLLW19]
principal component analysis [LMR14] [Tang18]
supervised clustering [LMR13] [Tang18]
support-vector machines [RML14] [DBH19]
low-rank matrix decomposition [RSML18] [GLT18; CLW18]
recommendation systems [KP17] [Tang19]
→ better candidate for exponential quantum speedup →

A classical analogue to the QSVT framework

We define the notion of *sampling and query access*, which serves as the analogue to the block-encoding.

QSVT [GSLW19]

- Get block-encodings to $A_1, \ldots, A_k$.
- Get a block-encoding for A where A is some bounded low-degree polynomial of the input.
- Apply to $|b\rangle$ to get $|Ab\rangle$.

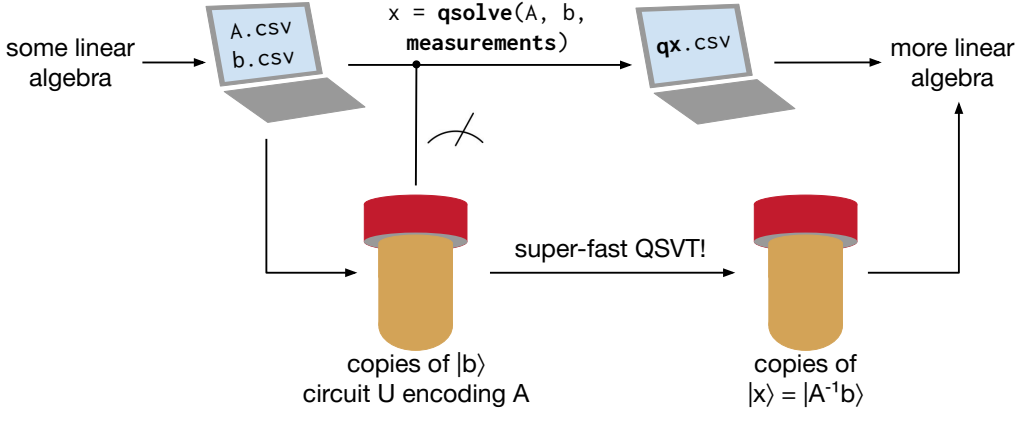
QI-SVT

- Get $\mathrm{SQ}(A_1), \ldots, \mathrm{SQ}(A_k)$.
- Get $\mathrm{SQ}(A)$ where A is some smooth function of the input.
- Apply to $\mathrm{SQ}(b)$ to get $\mathrm{SQ}(Ab)$.

This framework captures the capabilities of QRAM-based QSVT, giving strong evidence that quantum linear algebra based on QRAM admits no exponential speedups.

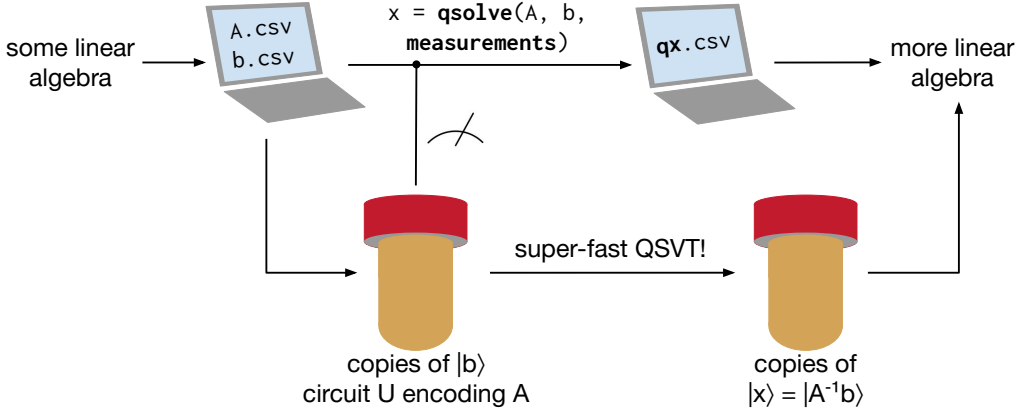
How to think of dequantized algorithms

Can we perform a classical version of the quantum algorithm, using *a little* bit of information from the quantum input assumptions (e.g. measurements of input quantum states)?



How to think of dequantized algorithms

Can we perform a classical version of the quantum algorithm, using *a little* bit of information from the quantum input assumptions (e.g. measurements of input quantum states)?



Well-known observation

Born rule-type measurements speed up machine learning and randomized linear algebra [SWZ16; HKS11; KV17; DMM08; FKV98]

Example: matrix inversion [SM21]

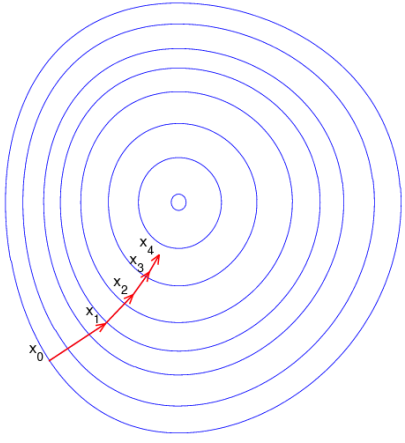
Consider solving a linear system with *gradient descent*: when $Ax^* = b$, $x^* = \min_x f(x) = \min_x \|Ax - b\|^2$.

$$x^{(0)} = \vec{0}$$
$$x^{(t)} = x^{(t-1)} - \eta_t \nabla f(x^{(t-1)})$$

Use samples to speed up evaluation of $\nabla f(x)$.

$$\begin{aligned}\nabla f(x) &= A^\dagger Ax - A^\dagger b \\ &= \sum_{i=1}^{m} A^\dagger_{i,*}(A_{i,*}x - b_i) \\ &= \sum_{i=1}^{m} A^\dagger_{i,*}\Big(\sum_{j=1}^{n} A_{i,j}x_j - b_i\Big)\end{aligned}$$

Use *measurements* to randomly sample i and j to get a $g_{i,j}(x)$ such that $\mathbb{E}_{i,j}[g_{i,j}(x)] = \nabla f(x)$. This is called *randomized Kaczmarz* [SV08].



Takeaways

QSVT with sparse input

- potential for exponential speedup
- mild hardware assumptions
- more brittle, less applicable

QSVT with QRAM input

- potential for large polynomial speedup
- advanced hardware assumptions
- less brittle, more applicable

My (biased) intuitions

- Though quantum computers implicitly perform extremely fast linear algebra, input and output problems mean that QSVT is perhaps not a fruitful perspective towards finding new quantum speedups to linear algebra calculations.
- Stepping outside this paradigm seems to be productive, i.e. by assuming quantum data.

Thank you!