Title: On quantum linear algebra for machine learning

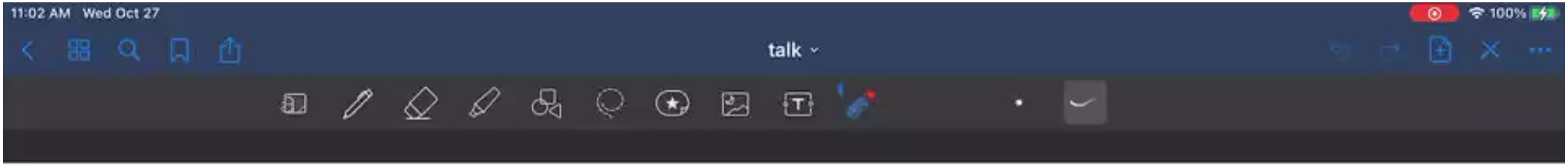
Speakers: Ewin Tang

Series: Colloquium

Date: October 27, 2021 - 2:00 PM

URL: https://pirsa.org/21100052

Abstract: We will discuss quantum singular value transformation (QSVT), a simple unifying framework for quantum linear algebra algorithms developed by Gilyén, Low, Su, and Wiebe. QSVT is often applied to try to achieve quantum speedups for machine learning problems. We will see the typical structure of such an application, the barriers to achieving super-polynomial quantum speedup, and the state of the literature that's attempting to bypass these barriers. Along the way, we'll also see an interesting connection between quantum linear algebra and classical sampling and sketching algorithms(explored in the form of "quantum-inspired" classical algorithms).



On quantum linear algebra for machine learning

Ewin Tang
University of Washington

October 27, 2021

What kinds of speedups can be gained from quantum computers?

Quantum Algorithm Zoo

This is a comprehensive catalog of quantum algorithms. If you notice any errors or omissions, please email me at stephen.jordan@microsoft.com. (Alternatively, you may submit a pull request to the repository on github.) Your help is appreciated and will be acknowledged.

Algebraic and Number Theoretic Algorithms

Algorithm: Factoring
Speedup: Superpolynomial
Description: Given an n-bit integer, find the prime factorization. The quantum algorithm of Peter Shor solves this in $\widetilde{O}(n^3)$ time [82,125]. The fastest known classical algorithm for integer factorization is the general number field sieve, which is believed to run in time $2^{\widetilde{O}(n^{1/3})}$. The best rigorously proven

Navigation

Translations

Source: quantumalgorithmzoo.org

Applications of quantum computing

1981 [Feynman] Quantum simulation

1994 [Shor] Crypto-analysis (factoring, discrete log)

1996 [Grover] Unstructured search

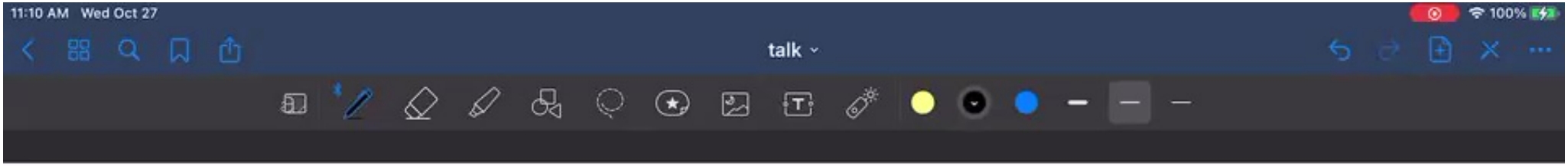
Applications of quantum computing

1981 [Feynman] Quantum simulation

1994 [Shor] Crypto-analysis (factoring, discrete log)

1996 [Grover] Unstructured search

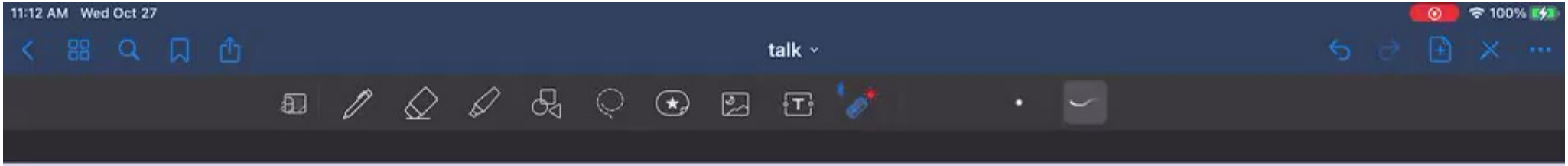
2009 [Harrow, Hassidim, Lloyd] ~~**Machine learning?**~~
Linear Algebra



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

This question:

- seems natural
- reflects a hope for exponential speedup that drives some of the hype behind quantum computing

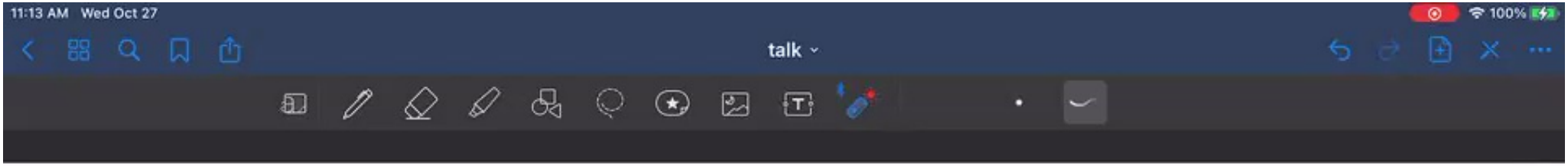


Talk overview

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

1. A framework that unifies many quantum algorithms by viewing them from a linear algebraic perspective
2. A series of quantum algorithms that apply this framework to solve linear algebraic tasks
3. A series of classical algorithms for linear algebra that can perform as well as the quantum ones, under certain circumstances

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]

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 algortihm, 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 magnitudes 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 magnitudes of a quantum state:

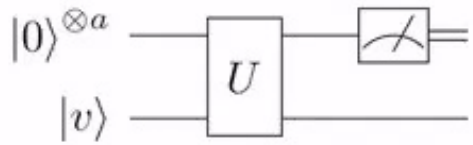
$$|v\rangle := \frac{1}{\|v\|} \sum_{i=1}^{N} v(i) \, |i\rangle$$

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 $\frac{1}{2}p(A)$,[1] 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.

[1] 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

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

   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

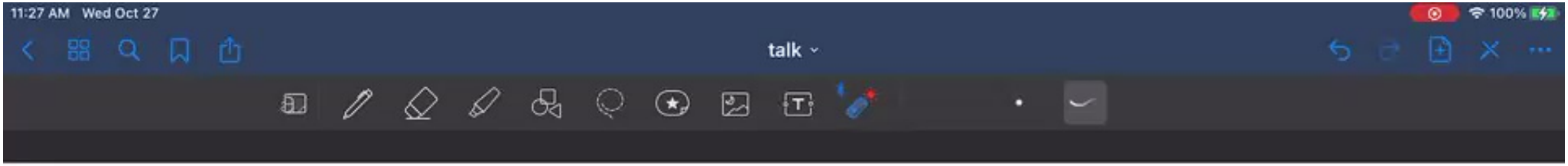
Jackson's inequality

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

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

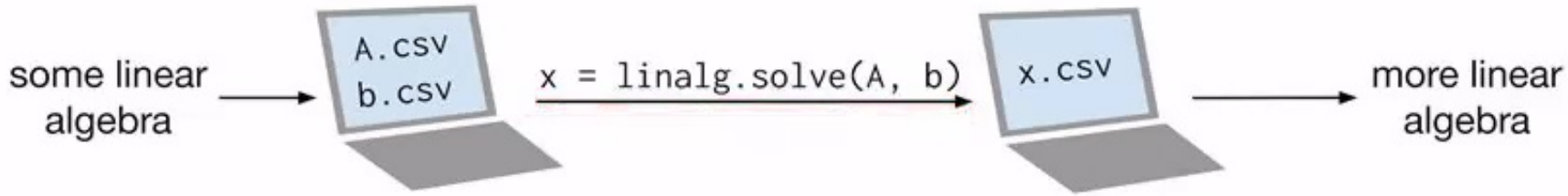
So, QSVT can be used to perform a wide range of operations in $O(\log N)$ time, *if* we have the right block-encodings.



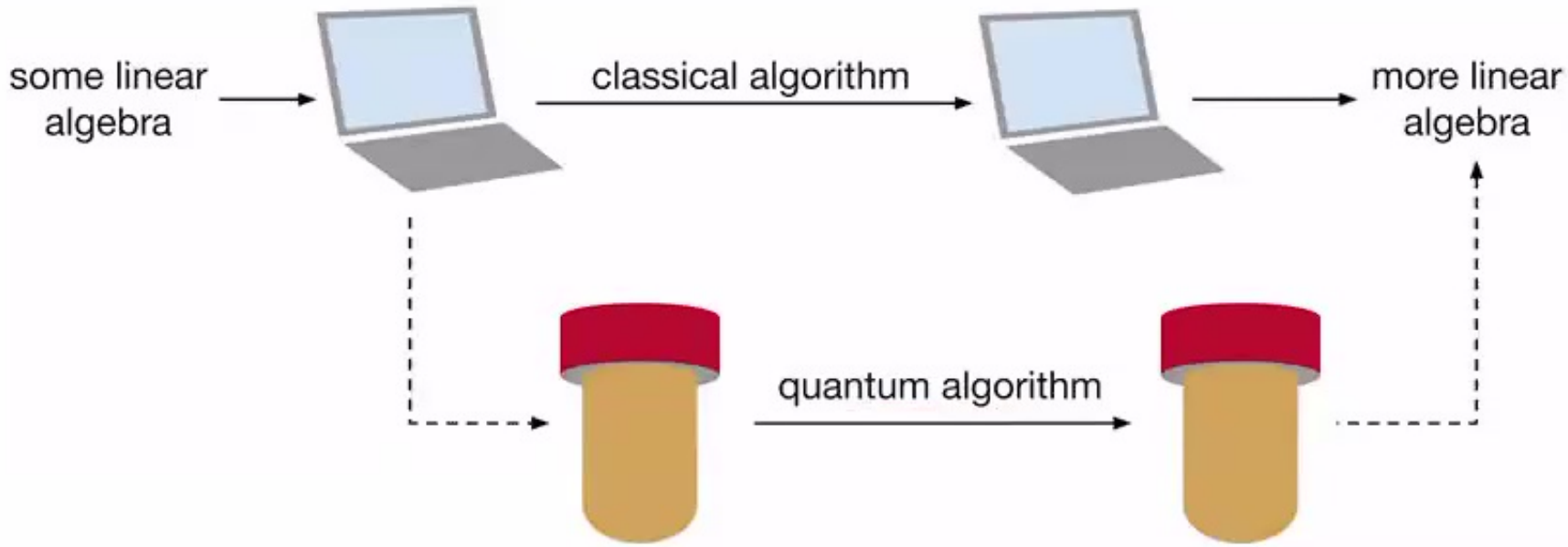
There are major barriers when trying to apply QSVT to machine learning tasks.

[Aaronson – Read the fine print]

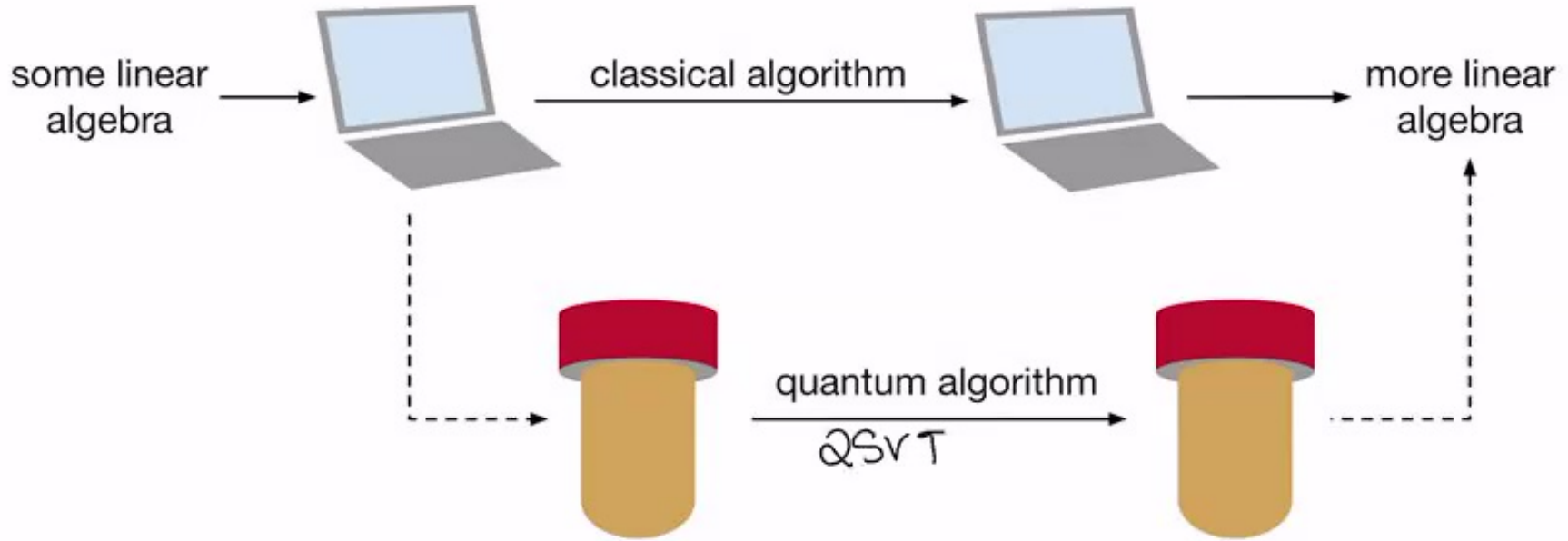
How we evaluate quantum speedups



How we evaluate quantum speedups



A typical application of quantum linear algebra to machine learning



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*
- $A/\operatorname{Tr}(A)$ if it can be prepared as a *purified density matrix*
- $A/\|A\|_{\mathrm{F}}$ if it is in *quantum random access memory*[2]

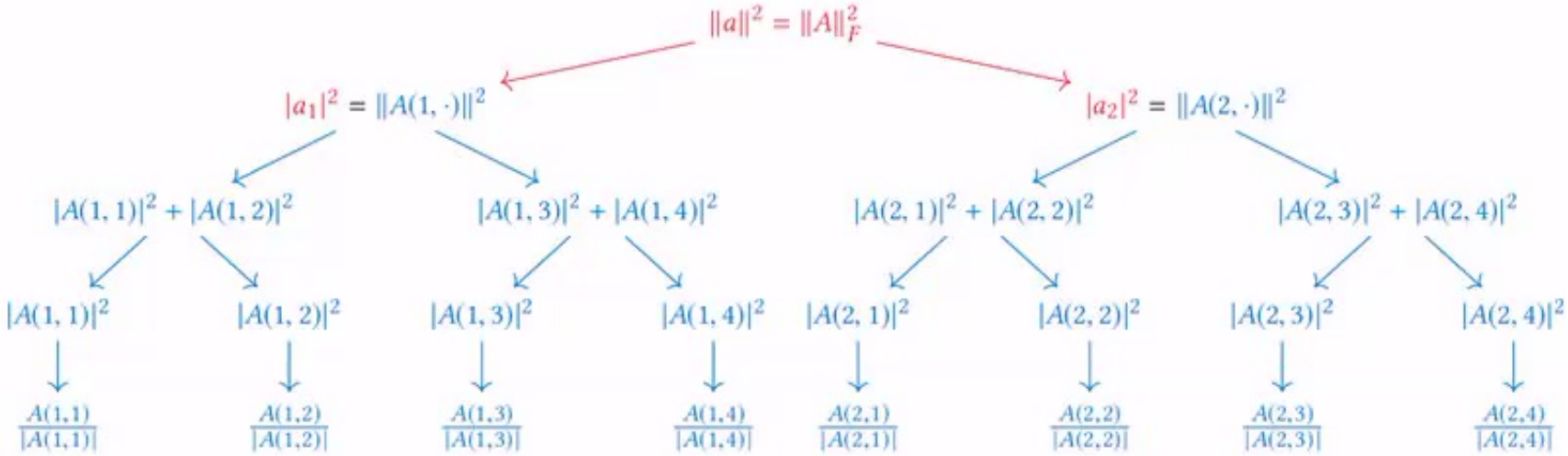
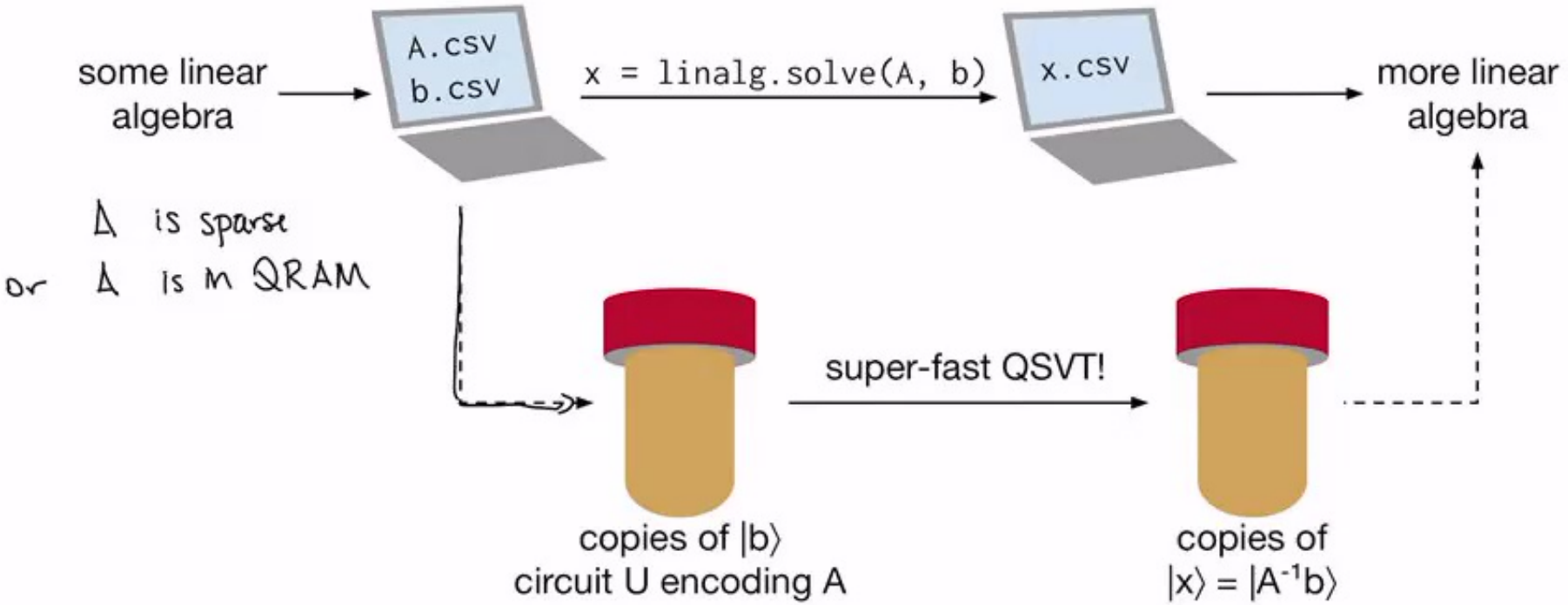


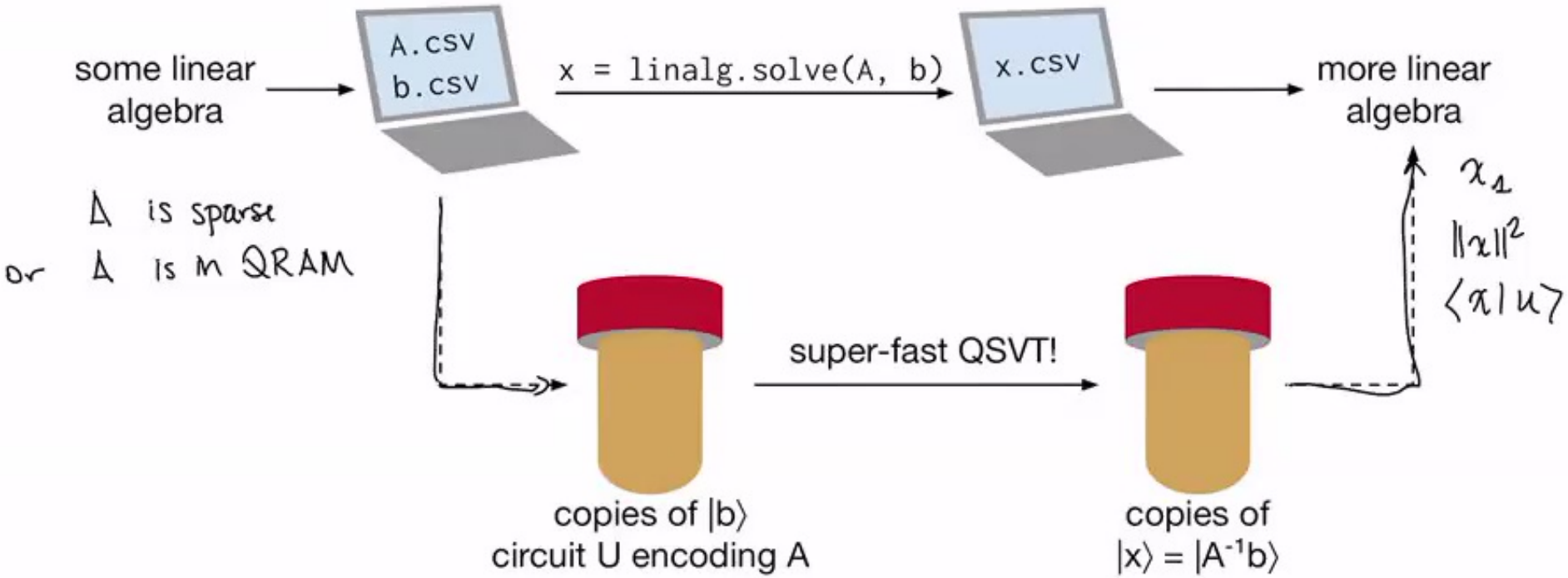
Figure 1: Dynamic data structure for $A \in \mathbb{C}^{2\times 4}$. We compose the data structure for a with the data structure for A's rows.

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

The input problem and the output problem



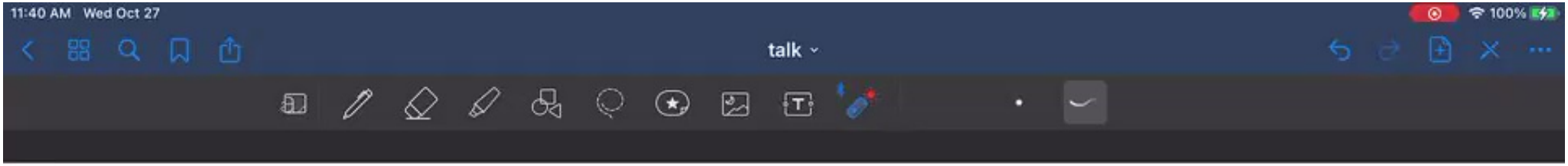
The input problem and the output problem



Take-aways

"*Quantum computers are slow in reading data... Quantum computers will excel for big-compute problems on small data*" –slide from Matthias Troyer's talk "Disentangling hype from reality"

- Though quantum computers implicitly perform extremely fast linear algebra, it's not clear if this could give broad speedups to linear algebra calculations.
- Current hopes in this discussion rely on (1) speculative quantum hardware and (2) finding the right application where it suffices to have very weak access to an output vector



Landscape: exponential speedups in quantum machine learning

sparse linear programming [HHL09]

supervised clustering [LMR13]

data fitting [WBL12]

principal component analysis [LMR14]

support-vector machines [RML14]

electromagnetic scattering [CJS13]

semidefinite programming [BKLLSW17]

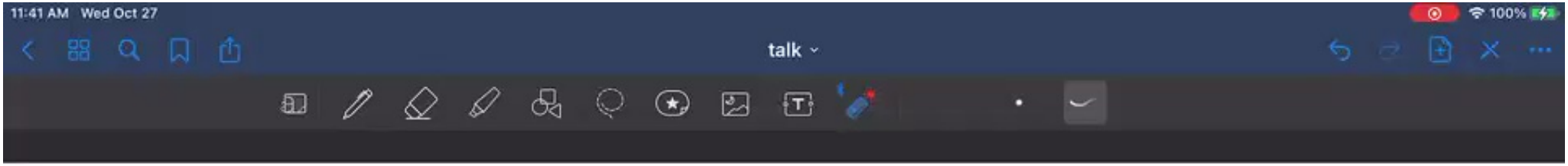
low-rank matrix decomposition [RSML18]

topological data analysis [LGZ16]

recommendation systems [KP17]

Gaussian process regression [ZFF19]

→ better candidate for exponential quantum speedup →



Landscape: exponential speedups in quantum machine learning

sparse linear programming [HHL09]

supervised clustering [LMR13]

data fitting [WBL12]

principal component analysis [LMR14]

support-vector machines [RML14]

electromagnetic scattering [CJS13]

semidefinite programming [BKLLSW17]

low-rank matrix decomposition [RSML18]

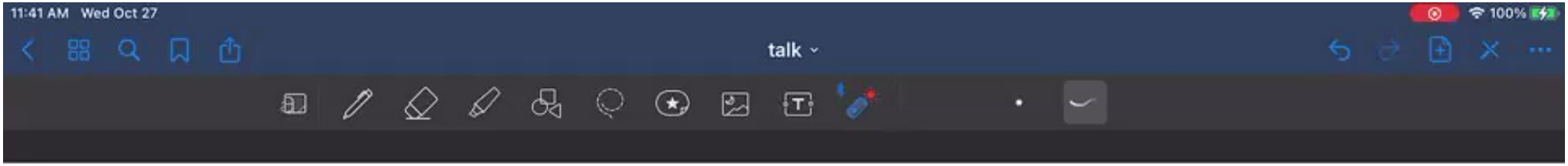
topological data analysis [LGZ16]

recommendation systems [KP17]

Gaussian process regression [ZFF19]

factoring [Shor97]

→ better candidate for exponential quantum speedup →



Landscape: exponential speedups in quantum machine learning

sparse linear programming [HHL09]

supervised clustering [LMR13]

data fitting [WBL12]

principal component analysis [LMR14]

support-vector machines [RML14]

electromagnetic scattering [CJS13]

low-rank matrix decomposition [RSML18]

semidefinite programming [BKLLSW17]

topological data analysis [LGZ16]

recommendation systems [KP17] [Tang19]

Gaussian process regression [ZFF19]

→ better candidate for exponential quantum speedup →

Landscape: exponential speedups in quantum machine learning

sparse linear programming [HHL09]

supervised clustering [LMR13] [Tang18]

data fitting [WBL12]

principal component analysis [LMR14] [Tang18]

support-vector machines [RML14] [DBH19]

electromagnetic scattering [CJS13]

semidefinite programming [BKLLSW17] [CLLW19]

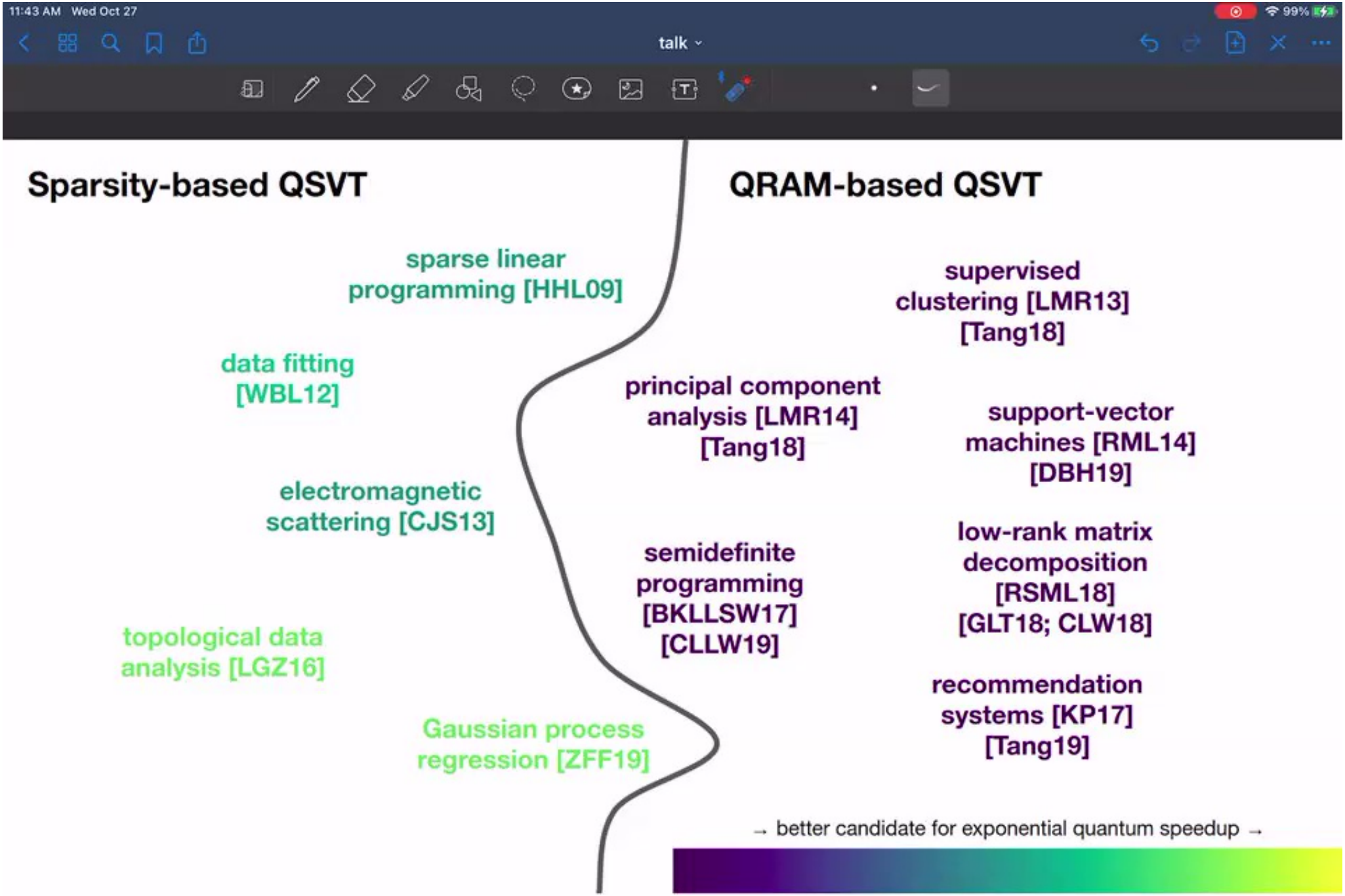
low-rank matrix decomposition [RSML18] [GLT18; CLW18]

topological data analysis [LGZ16]

recommendation systems [KP17] [Tang19]

Gaussian process regression [ZFF19]

→ better candidate for exponential quantum speedup →


11:43 AM Wed Oct 27
99%
talk
Sparsity-based QSVT
QRAM-based QSVT
sparse linear programming [HHL09]
data fitting [WBL12]
electromagnetic scattering [CJS13]
topological data analysis [LGZ16]
Gaussian process regression [ZFF19]
supervised clustering [LMR13] [Tang18]
principal component analysis [LMR14] [Tang18]
support-vector machines [RML14] [DBH19]
semidefinite programming [BKLLSW17] [CLLW19]
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 $\mathrm{SQ}_{\varphi_1}(A_1), \ldots, \mathrm{SQ}_{\varphi_k}(A_k)$.
- Get $\mathrm{SQ}_{\varphi_A}(A)$ where A is some smooth function of the input.
- Apply to $\mathrm{SQ}_{\varphi_b}(b)$ to get $\mathrm{SQ}_{\phi}(Ab)$.

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

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 $|Ab\rangle$.

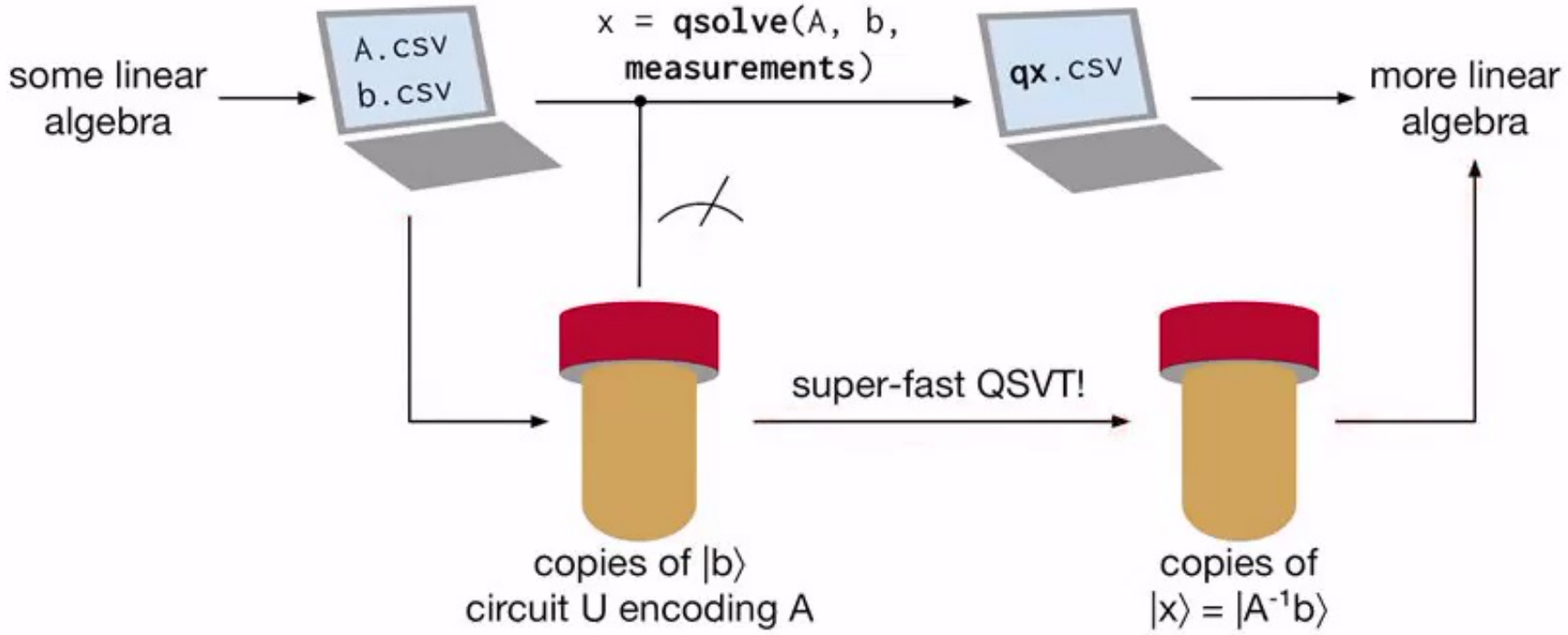
QI-SVT

- Get $\mathrm{SQ}_{\varphi_1}(A_1), \ldots, \mathrm{SQ}_{\varphi_k}(A_k)$.
- Get $\mathrm{SQ}_{\varphi_A}(A)$ where A is some smooth function of the input.
- Apply to $\mathrm{SQ}_{\varphi_b}(b)$ to get $\mathrm{SQ}_{\phi}(Ab)$.

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

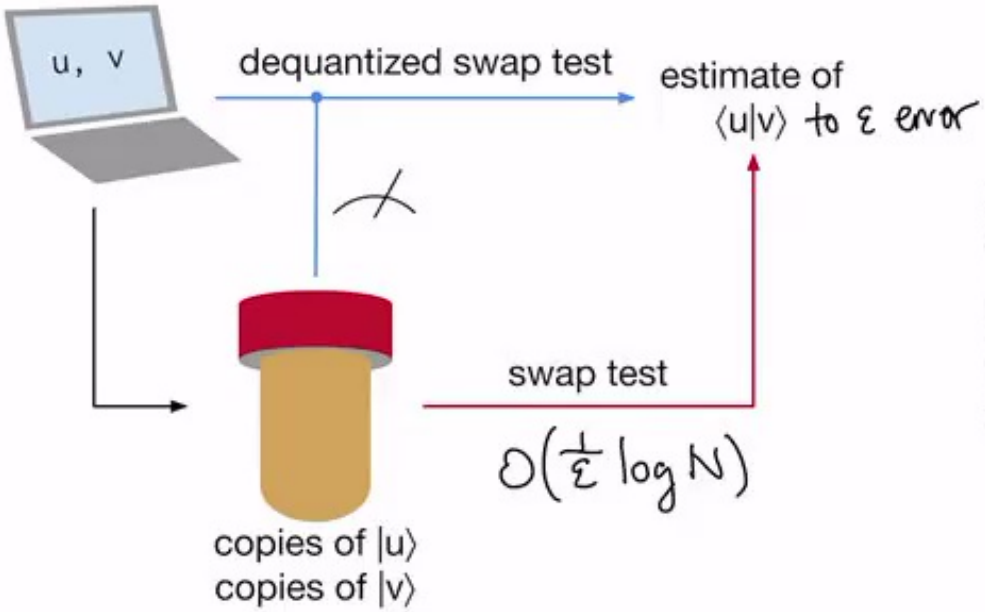
How to think of dequantized algorithms

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)?



Toy example: dequantizing the *swap test*

$u, v \in \mathbb{C}^N \quad \|u\| = \|v\| = 1$



Input: u, v, measurements of $|u\rangle$

Algorithm:

- pull a sample i w.p. $|u_i|^2$
- read u_i, v_i
- let $z = \frac{\bar{u}_i v_i}{|u_i|^2}$

$\mathbb{E}[z] = \sum \frac{\bar{u}_i v_i}{|u_i|^2} |u_i|^2 = \langle u|v \rangle$

$\mathrm{Var}[z] \le \sum (\quad)^2 |u_i|^2 \le 1$

→ do $\frac{1}{\varepsilon^2}$ times and output the average of the z's

Quantum-inspired algorithms exploit sampling strategies familiar to the classical algorithms literature

Well-known observation

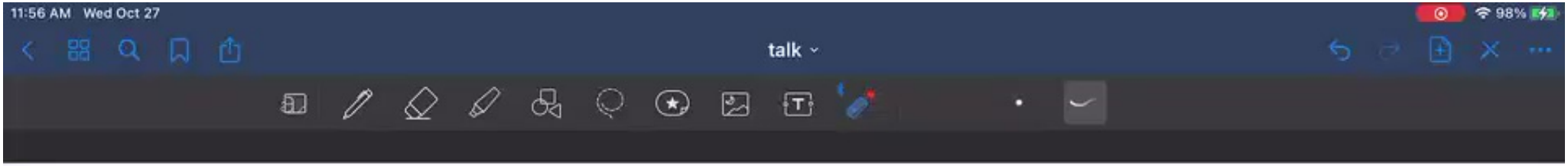
Born rule-type measurements speed up machine learning and randomized linear algebra [SWZ16; HKS11; KV17; DMM08; FKV98]

Take-away

The best hope for exponential quantum speedups from QSVT is when the input matrices are *sparse*.

Note the limitations of this result:

- Algorithms are relatively impractical compared to existing classical algorithms
- Leaves open the possibility of large polynomial speedups



Some interesting directions forward

Take-away

The best hope for exponential quantum speedups from QSVT is when the input matrices are *sparse*.

Note the limitations of this result:

- Algorithms are relatively impractical compared to existing classical algorithms
- Leaves open the possibility of large polynomial speedups

How big is the polynomial quantum speedup for QSVT really?

- Getting better quantum-inspired algorithms for regression via optimization

  [Gilyén, Song, Tang – An improved quantum-inspired algorithm for linear regression]
  [Shao, Montanaro – Faster quantum-inspired algorithms for solving linear systems]

- Making the correspondence between QSVT and QI-SVT cleaner

What possibilities for super-polynomial quantum speedup for QML are left?

- Topological data analysis

  [Lloyd, Garnerone, Zanardi – Quantum algorithms for topological and geometric analysis of data]
  [Gunn, Kornerup – Review of a quantum algorithm for Betti numbers]
  [Gyurik, Cade, Dunjko – Towards quantum advantage via topological data analysis]

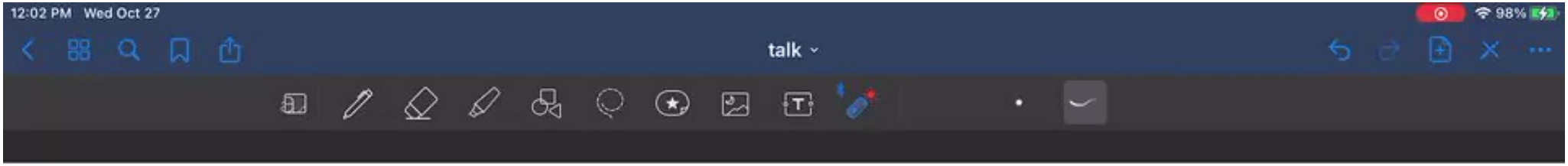
- Gaussian process regression

  [Zhao, Fitzsimons, Fitzsimons – Quantum-assisted Gaussian process regression]

- Sampling optimized random features

  [Yamasaki, Subramanian, Sonoda, Koashi - Learning with optimized random features]

Could QSVT be avoided entirely? Is there another method for block-encoding that avoids these issues?



Thank you!