

Lecture 5:

Quantum algorithm for solving linear systems

COMP3366

Quantum algorithms & computing architecture

Instructor: Yuxiang Yang

Department of Computer Science, HKU

Objectives:

- **[O1] Concepts:** HHL algorithm and its applications.
- **[O2] Problem solving:** Understanding the working principle of HHL algorithm, understanding the role of QPE in quantum algorithms.
- **[O3] Algorithm design:** Rejection sampling & Uncomputing.

Part I: Solving linear systems

Linear systems

- Solving a linear system:

Given a matrix A and a vector \vec{b} , find a vector \vec{x} such that $A \vec{x} = \vec{b}$.

- Example:

$$\begin{array}{rcl} 3x - y & = & 7 \\ + 2x + y & = & 8 \\ \hline 5x + 0 & = & 15 \end{array} \Rightarrow \begin{pmatrix} 3 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \end{pmatrix}$$

- For very large linear systems, it may be impossible to solve \vec{x} exactly.
- Instead, we may output an **approximate** solution \vec{x}' such that $\vec{x}' \approx \vec{x}$ up to a small error ϵ .

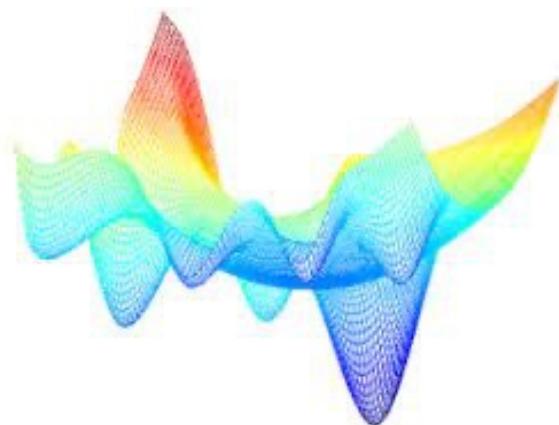
$\exists A \vec{x} = b \nexists$
 \vec{x} 近似 \vec{x}' , s.t. $|\vec{x}' - \vec{x}| \leq \epsilon$

Importance

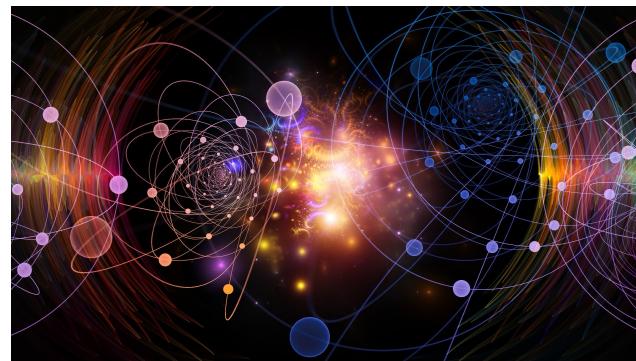
- Why bother solving linear systems?

Solving linear systems are important not for passing the linear algebra course.

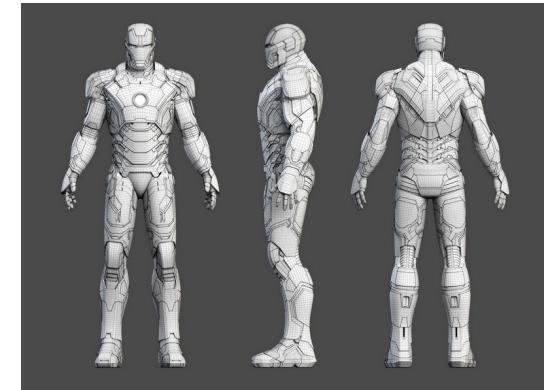
It is the bread-and-butter problem for science!



*Optimization
& Finance*



Quantum physics



3D graphics

Problem formulation

- Solving an N -dimensional linear system:

Given a matrix A and a vector \vec{b} , find a vector \vec{x} such that $A \vec{x} = \vec{b}$.

- The complexity often depends on two parameters:

- κ : the condition number \rightarrow 此时用 AHL 运行时间.
= the ratio between largest absolute and smallest absolute of eigenvalues.
- We assume A has non-zero eigenvalues whose absolute values range from κ^{-1} to 1, otherwise we can rescale A . That is, any eigenvalue $|\lambda| \in [\kappa^{-1}, 1]$.
- s : (non)-sparsity = the number of non-zero eigenvalues.

- Example:

$$A = \begin{pmatrix} 0.5 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.125 \\ 0.25 & 0.5 & 0.5 \end{pmatrix}$$

It has eigenvalues $\lambda_1 = 1, \lambda_2 = 0.25, \lambda_3 = 0 \Rightarrow s = 2, \kappa = 4$.

Complexity of classical solvers

- Solving an N -dimensional linear system:

Given a matrix A and a vector \vec{b} , find a vector \vec{x} such that $A \vec{x} = \vec{b}$.

- Classical solvers have complexity $\Omega(N)$.

- The basic Gaussian elimination algorithms has complexity $O(N^3)$

- More advanced approaches, e.g., conjugate gradient descent method has complexity $O\left(Ns\kappa \log\left(\frac{1}{\epsilon}\right)\right)$.

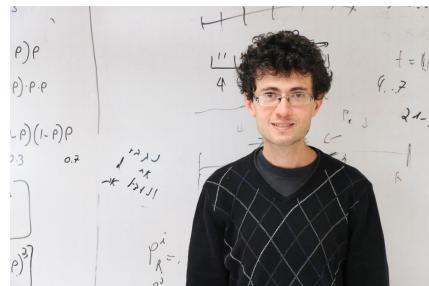
Part II: The HHL algorithm

Overview of the HHL algorithm

- The HHL (Harrow-Hassidim-Lloyd) algorithm:
 - is a quantum algorithm that solves a N -dimensional linear system $A \vec{x} = \vec{b}$ up to an error ϵ with circuit complexity $\tilde{O}((\log N) \cdot s^2 \cdot \kappa^3 / \epsilon)$.
- Here s is the sparsity and κ is the condition number introduced before.
- \vec{b} is given as a quantum state $|b\rangle$ (see bonus material for more discussions).
- Main message:
 Exponential speedup: $N \rightarrow \sim \log N$ for sparse matrices with $s = O(\log N)$.
Big-O, up to some unimportant terms
- Potential exponential quantum advantage but not confirmed (like Shor).



Aram *Harrow*



Avinatan *Hassidim*



Seth *Lloyd*

Setting

重要假设

Exercise: Verify the “w.l.o.g.” claim in the Hermitian assumption.

- Hermitian assumption:

We can assume w.l.o.g. $A = A^\dagger$, because otherwise we can just solve

$\tilde{A}\vec{y} = \vec{b}'$ with $\tilde{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$ and $\vec{b}' = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$. The solution will be $\vec{y} = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}$.

- Linear algebra 101: For a Hermitian A , we can always diagonalize it as

$$A = \sum_i \lambda_i |u_i\rangle\langle u_i|, \quad \lambda_i \in \mathbb{R}.$$

- Here $\{|u_i\rangle\}$ is a set of orthonormal vectors, and we can write $|b\rangle = \sum_j b_j |u_j\rangle$. Then the solution is simply:

$$|x\rangle = \sum_i \frac{b_i}{\lambda_i} |u_i\rangle$$

Prove that $|b\rangle$ should live in the eigenspace of non-zero eigenvalues. Otherwise, there is no solution to the linear system.

- The problem is that the diagonalization takes too much effort! HHL algorithm exploits the above reasoning and computes $|x\rangle$ without explicitly doing the diagonalization.

X

HHL algorithm

- **Input:**
 $N \times N$ Hermitian matrix A , and a unit vector b .
- **Output:** A quantum state $|x\rangle \approx A^{-1}|b\rangle$

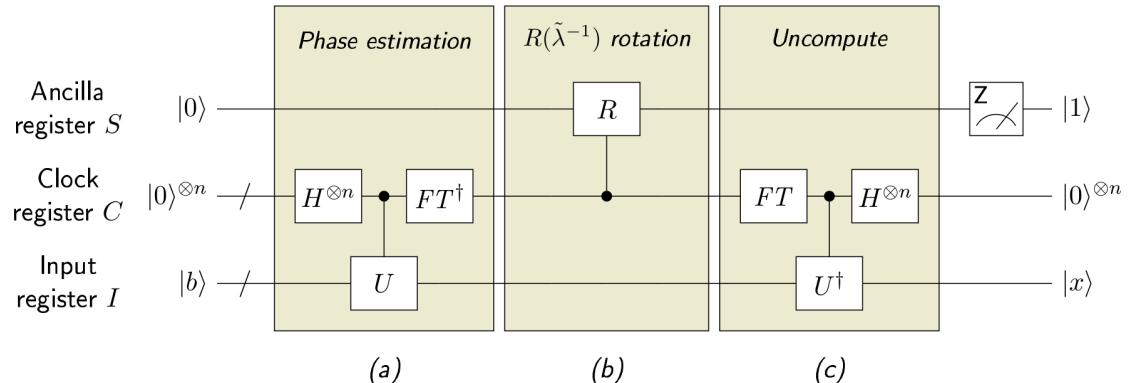
$\beta_j|b\rangle = \sum_j \beta_j |u_j\rangle$
QPE $\sum_j \beta_j |\tilde{\lambda}_j\rangle \otimes |u_j\rangle$ 对 A 特征值 λ_j
 的近似

- **Procedure (simplified):** $\xrightarrow{\text{QPE}} \sum_j \beta_j |\tilde{\lambda}_j\rangle \otimes |u_j\rangle$

a) Apply QPE for the unitary $e^{2\pi i At}$ on $|0^n\rangle \otimes |b\rangle$.

b) (**Rejection sampling**) Apply a controlled-unitary $|0\rangle \mapsto \sqrt{1 - \frac{c^2}{\lambda_k^2}}|0\rangle + \frac{c}{\lambda_k}|1\rangle$ on an ancillary qubit with the clock register as control.
 Measure the ancillary in the computational basis, and restart from a) if the outcome is 0.

c) (**Uncomputation**) Apply inverse of QPE to the clock register. Output the input register.



We learned a) in last lecture; today we focus on b) and c)

拿取 $e^{2\pi i A t}$

Step 1: Encode A into a unitary

- A is neither a quantum state nor a quantum gate.
- We need first to encode A into a gate (or a state, but HHL does the former).
- **Question:** How?

Recall that A can be assumed w.l.o.g. to be Hermitian.

- Answer: $A \rightarrow e^{2\pi i A t}$ for some time parameter $t \in \mathbb{R}$ (controlled by us)!
- Since $A = \sum_j \lambda_j |u_j\rangle\langle u_j|$ is **Hermitian**, $e^{2\pi i A t} = \sum_j e^{2\pi i \lambda_j t} |u_j\rangle\langle u_j|$ must be a **unitary**!

We assumed A has non-zero eigenvalues whose absolute values range from κ^{-1} to 1

- **Regularization:** We choose $t = (2 \max |\lambda_j|)^{-1} = \frac{1}{2}$ so that $\lambda_j t \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ for every λ_j . We can shift by a global phase so that, effectively, $\lambda_j t \in [0,1]$ and can be treated with QPE.

Step 1: Encode A into a unitary

- The $A \rightarrow e^{2\pi i A t}$ transformation can be done via quantum simulation, with an $\tilde{O}(\log(N) s^2 t)$ time complexity. (Technical details omitted.)
- Note: This doesn't require the knowledge of $\{|u_j\rangle\}$
- It is efficient if s is not large, i.e., if A is sparse. → 有很少的



*Richard P. Feynmann
on Quantum Simulation, 1982*

Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical ... (we need) not a Turing machine, but a machine of a different kind.

$$3K+b \geq m$$

$$3K = 104$$

$$k = 38$$

$$A = \sum_j b_j |u_j\rangle \langle u_j|$$

$$\text{对 } V = e^{2\pi i At} \text{ 应用 QPE} \rightarrow \sum_j b_j |\lambda_j t\rangle_C |u_j\rangle_I$$

Step 2: QPE

→ we want

$$\sum_j \frac{b_j}{\lambda_j} |u_j\rangle$$

- Question: How do we make use of $e^{2\pi i At}$?

- Answer: Quantum Phase Estimation!

- We can apply QPE with $e^{2\pi i At}$ and $|b\rangle = \sum_j b_j |u_j\rangle$ as the input.

- Assume for now that the QPE has no error.

This happens when every $\lambda_j t$ is a n -digit binary number.

Recall that we choose t so that $\lambda_j t \in (0,1]$ for every j .

We get (before the measurement in the computational basis) the state :

$$\sum_j b_j |\lambda_j t\rangle_C |u_j\rangle_I$$

- Our goal is $|x\rangle \propto \sum_i \frac{b_i}{\lambda_i} |u_i\rangle$.

How can we extract λ_i from $|\lambda_i t\rangle$ and apply $1/\lambda_i$ to the amplitude?

Loading a function onto a state



- Suppose you have a state $|\psi\rangle \propto \sum_j b_j |j\rangle$ in the computational basis, how to transform it into $|\psi'\rangle \propto \sum_j b_j f(j) |j\rangle$ for some known function f ? (e.g. $f(j) = j^2$)

- Hint: There might not exist a gate U such that $|\psi'\rangle = U|\psi\rangle$. But we can try a probabilistic approach.

- Rejection sampling** for loading $f(j)$:

- Initialize **an ancilla qubit S** in $|0\rangle$ and the overall state $|\psi_1\rangle = \sum_j b_j |0\rangle_S |j\rangle$.
- Perform $W = \sum_i (R_i)_S \otimes |i\rangle\langle i|$ and the state becomes

$$|\psi_2\rangle = \sum_j b_j \left(\sqrt{1 - A^2 |f(j)|^2} |0\rangle + A f(j) |1\rangle \right)_S \otimes |j\rangle.$$

Exercise: What's the probability of success in one single round?

- Measure the ancilla:
If outcome = 1, output state = $|\psi'\rangle$; if outcome = 0, restart from Step 1.
- Remark: “amplitude kickback” (but only probabilistically).

Post-measurement state for partial measurements

- Recall (Lecture 4): Born's rule for partial measurement:

$$P(x) = \langle \psi | (|\phi_x\rangle\langle\phi_x| \otimes I) |\psi\rangle$$

Bipartite state
(e.g., the state of the clock register +
input register in QPE)

- Recall (Lecture 1): ~~3B1~~ ~~是~~ ~~从~~ ~~.14>~~ collapse to:
When measuring $|\psi\rangle$ in basis $\{|\phi_x\rangle\}$, if we get outcome x the state collapse to $|\phi_x\rangle$.

- Combining these:

The post-measurement state for partial measurement is:
$$\frac{(|\phi_x\rangle\langle\phi_x| \otimes I) |\psi\rangle}{\sqrt{P(x)}}.$$

- Example:

- $|\psi\rangle = |\psi_1\rangle|\psi_2\rangle$ is a product state \Rightarrow post-measurement state on getting outcome x is $|\phi_x\rangle|\psi_2\rangle$ (which is proportional to $\langle\phi_x|\psi_1\rangle|\phi_x\rangle|\psi_2\rangle$).
- Measuring the 1st qubit of Bell state $|B_0\rangle$ in the computational basis collapse the state to $|00\rangle$ or $|11\rangle$, measuring in the Hadamard basis collapse the state to $|+\rangle$ or $|-\rangle$.

scalar

Applying the rules of partial measurement ...

- In rejection sampling, the pre-measurement state is

$$|\psi_2\rangle = \sum_j b_j \left(\sqrt{1 - A^2 |f(j)|^2} |0\rangle + A f(j) |1\rangle \right)_S \otimes |j\rangle.$$

- By the rules of partial measurement: when getting the outcome 1, post measurement state is $\frac{(|1\rangle\langle 1|_S \otimes I)|\psi_2\rangle}{\sqrt{P(1)}}$, with $P(1) = \langle\psi_2|(|1\rangle\langle 1|_S \otimes I)|\psi_2\rangle$.
- When we get outcome 1, the post-measurement state becomes
$$\propto \sum_j b_j f(j) |j\rangle.$$
- With rejection sampling, we can probabilistically load an arbitrary function onto a state.

Step 3: Rejection sampling in HHL

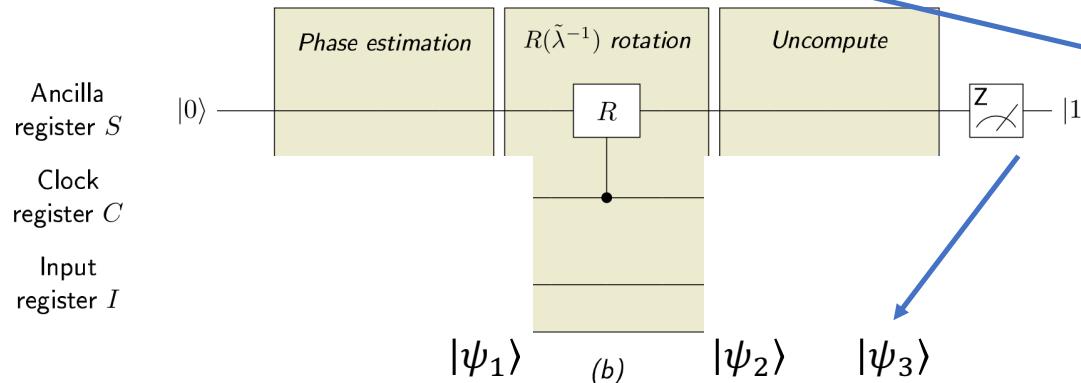
- We can now apply rejection sampling (with C as control and $f(\lambda) \propto 1/\lambda$):
 - We initialize an ancilla qubit in $|0\rangle$ and the (overall) state $|\psi_1\rangle = \sum_j b_j |0\rangle_S |\lambda_j t\rangle_C |u_j\rangle_I$.
 - We perform $W = \sum_i (R_i)_S \otimes |i\rangle\langle i|_C \otimes I_I$ and the state is

$$|\psi_2\rangle = \sum_j b_j \left(\sqrt{1 - A^2/(\lambda_j t)^2} |0\rangle + A/\lambda_j t |1\rangle \right)_S |\lambda_j t\rangle_C |u_j\rangle_I.$$

$R_i = \begin{pmatrix} \sqrt{1 - A^2/i^2} & A/i \\ -A/i & \sqrt{1 - A^2/i^2} \end{pmatrix}$
 - We measure the ancilla and succeed when getting 1, the post-measurement state becomes:

$$|\psi_3\rangle \propto \left(\frac{A}{t}\right) |1\rangle_S \otimes \sum_j \frac{b_j}{\lambda_j} |\lambda_j t\rangle_C |u_j\rangle_I$$

$|\lambda_j t\rangle$ = the binary representation of $\lambda_j t$ in the computational basis:
e.g., $\lambda_j t = 0.1101 \mapsto |1101\rangle$.



Decoupled with C and I .
Can be discarded without affecting the other two registers.

Step 4: Uncomputation

- Question: Are we done?
- **Answer: No!** Our final target is

$$\sum_j \frac{b_j}{\lambda_j} |u_j\rangle_I$$

- Now the state is proportional to

$$\sum_j \frac{b_j}{\lambda_j} |\lambda_j t\rangle_C |u_j\rangle_I$$

and the clock register is **entangled** with the input register!

- Solution: Our goal is to make the clock register $|0\rangle$.
We will need to **uncompute** the QPE by applying its inverse!

Note: Inversion of a quantum circuit

- Notice that a quantum circuit is represented as a unitary, consisting of a sequence of gates: $U = V_1 V_2 V_3 V_4 \dots V_k$.
- To invert the quantum circuit, we have to execute U^\dagger .
- By the definition of adjoint, $U^\dagger = (V_1 V_2 V_3 V_4 \dots V_k)^\dagger = V_k^\dagger \dots V_2^\dagger V_1^\dagger$.
- Therefore, to invert a quantum circuit, we can execute the inverse of each gate in the reversed order.

Step 4: Uncomputation

- Our final target is $\sum_j \frac{b_j}{\lambda_j} |u_j\rangle_I$. Now the state is proportional to $\sum_j \frac{b_j}{\lambda_j} |\lambda_j t\rangle_C |u_j\rangle_I$ and the clock register is **entangled** with the input register!
- We will need to **uncompute** the QPE by applying its inverse:
 1. Apply QFT on C . $\sum_j \frac{b_j}{\lambda_j} |\lambda_j t\rangle_C |u_j\rangle_I \mapsto \sum_j \frac{b_j}{\lambda_j} |e_{\lambda_j t}\rangle |u_j\rangle$
 2. Apply $\sum_k |k\rangle\langle k| \otimes e^{-2\pi i A k t}$. $\sum_j \frac{b_j}{\lambda_j} |e_{\lambda_j t}\rangle |u_j\rangle \mapsto \sum_j \frac{b_j}{\lambda_j} (W_j |e_{\lambda_j t}\rangle) |u_j\rangle$ with $W_j = \sum_k e^{-2\pi i \lambda_j k t} |k\rangle\langle k|$. The state is thus equal to $\sum_j \frac{b_j}{\lambda_j} |e_0\rangle |u_j\rangle$
 3. Apply $H^{\otimes n}$ on C . The final state is $\sum_j \frac{b_j}{\lambda_j} |0\rangle |u_j\rangle$ as desired.
- **Now all done!** We have $|x\rangle \propto \sum_j \frac{b_j}{\lambda_j} |u_j\rangle$ on the input register.

Review: new tools we learned in this lecture ...

- **Rejection sampling:**

Realize a nonlinear function (“amplitude kickback”) probabilistically.

$$\sum_j b_j |j\rangle \rightarrow \sum_j b_j f(j) |j\rangle.$$

- **Uncomputation:**

Get rid out undesired correlation between the system and the ancilla.

- **Partial measurement post-measurement state:**

Measuring the first register of a bipartite state $|\psi\rangle$ in basis $\{|\phi_x\rangle\}$, if we get outcome x , the state collapses (partially) to:

$$\frac{(|\phi_x\rangle\langle\phi_x| \otimes I)|\psi\rangle}{\sqrt{P(x)}}.$$

Complexity of HHL

- Previously we assumed QPE has no error. In practice, we cannot guarantee $\lambda_j t$ to be exactly n -bit. But we can ensure this up to a small error by running QPE with more qubits. This leads to a tradeoff between error and complexity.
- Recall: QPE achieves accuracy $1/\delta$ ($|\hat{\varphi} - \varphi| < \delta$ almost surely) of estimating λt with $O(1/\delta)$ queries to U .
- For the output to be ϵ -close to $|x\rangle = \sum_i \frac{b_i}{\lambda_i} |u_i\rangle$, λ_i should have error $O(\epsilon \cdot \kappa^{-1})$. Then we should choose $\delta = \epsilon t \cdot \kappa^{-1}$.
- For HHL, each U has $\tilde{O}(\log(N) s^2 t)$ time complexity (**bottleneck = quantum simulation**). In total, the time complexity of QPE is $\tilde{O}(\log(N) s^2 \kappa / \epsilon)$.
- The success probability of rejection sampling is $\Omega(\kappa^{-2})$, and the algorithm succeeds with high probability within $O(\kappa^2)$ iterations.
- Overall, the complexity of HHL is proportional to $\log N$ instead of N . There is also a dependence on s , so if the matrix is non-sparse ($s = O(N)$)/ill-behaved (large κ) it is not a good idea to use HHL.

HHL vs Shor



- Discussion:

HHL is reminiscent of Shor. What are the similarities and differences?

- **Similarities:**

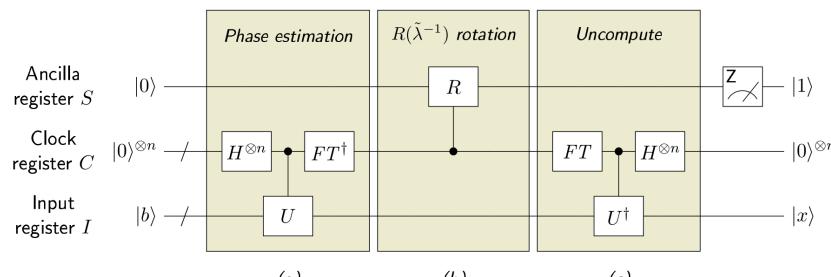
Classical information (Shor: order/period; HHL: eigenvalues)

→ encoded into phase of a unitary

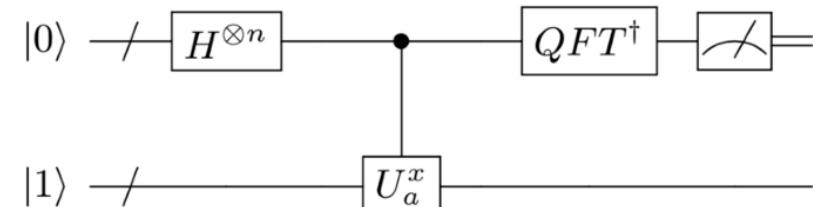
→ processed via QPE (on a superposition of difference eigenvectors).

- Differences:

1. HHL uses an additional qubit ancilla and requires rejection sampling.
 2. HHL requires undo the QPE.



HHI



Short

Improvement of HHL*

- Replacing $|e_0\rangle$ by the optimal clock state (*Buzek-Derka-Massar'98*; see Assignment 2, Q3) improves the accuracy of QPE, and the efficiency of HHL as well.
- Using a technique named amplitude amplification, we can improve the efficiency of the rejection sampling (the success probability). We will learn it in “*Grover's algorithm*” next time.
- There are methods of improving the quantum simulation subroutine and to replace it with the state-of-the-art. (This is beyond the scope of this course.) But the $\log N$ scaling cannot be further improved.

Part III * :

Application of HHL

* Bonus content

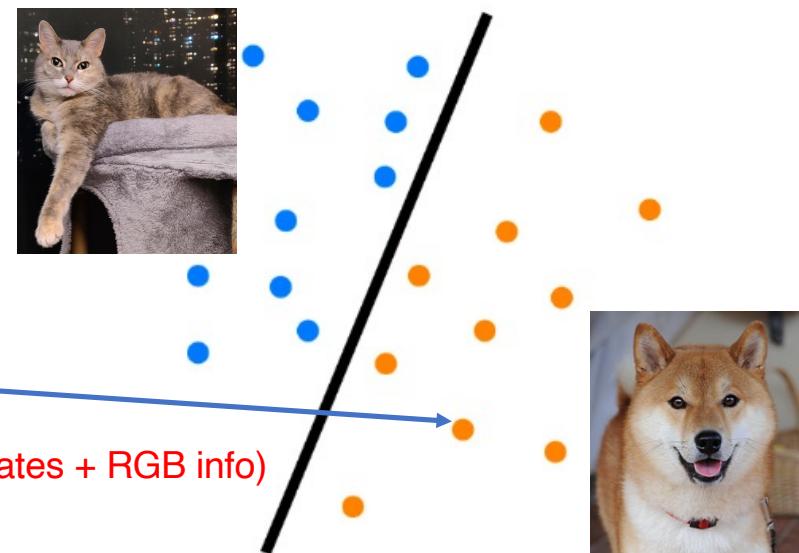
Classification

- **Classification:**

Given M training data of dimension N as well as their labels, the goal is to construct a classifier f



Encoded into \mathbb{R}^N
(for instance, pixel coordinates + RGB info)



Support vector machines (SVMs)

- Goal:

Create classifier $g(\vec{x}) = \vec{w} \cdot \vec{x} + b$ so that

$$g(\vec{x}) \cdot y > 0$$

for any testing datum \vec{x} .

- The choice of $g(\vec{x})$ is not unique.

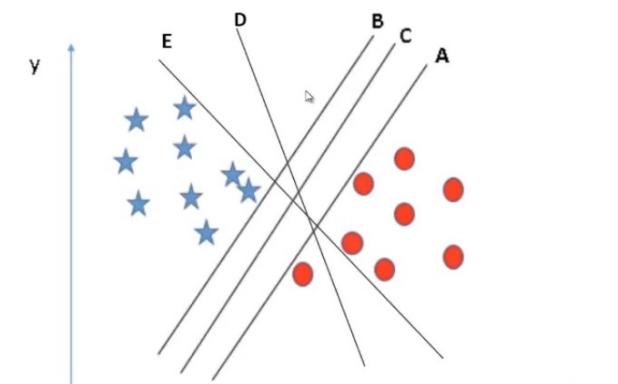
How should we choose the classifier so that it behaves well for future data?

- Solution: maximize the margin!

Margin = minimal distance between any datum and the hyperplane

- Equivalent to minimizing $|\vec{w}|$, subject to the constraint: $g(\vec{x}) \cdot y \geq 1$ for any testing datum \vec{x} .

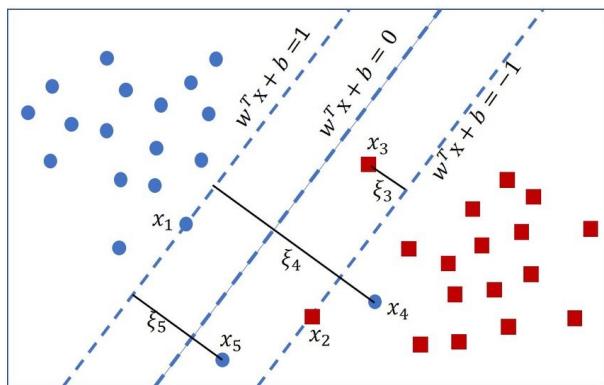
Equivalent to $g(\vec{x}) > 0 \Rightarrow y = 1$ and $g(\vec{x}) < 0 \Rightarrow y = -1$!



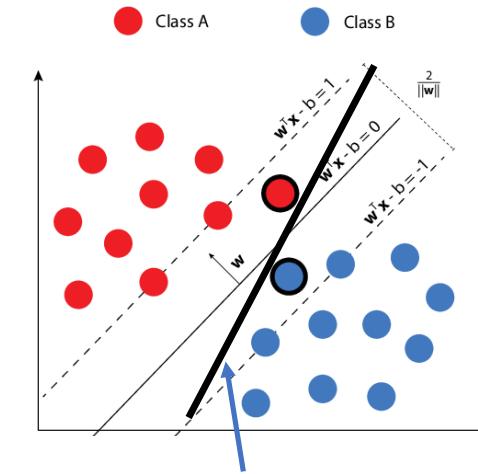
The choice of $g(\vec{x})$ is not unique!

Soft-margin SVMs

- Soft-margin: tolerate a few bad data while giving them some penalties.
- Why soft-margin:
 - Hard-margin SVM requires the data to be perfectly separable, which often cannot be satisfied!
 - Allowing more flexibility often results in more reasonable classifiers!



Hard-margin SVM does not work for data that are not perfectly separable!



Hard-margin SVM would have a much smaller margin and a less reasonable classifier!

Soft-margin SVMs

- Soft-margin SVM optimization problem:

$$\begin{aligned} \min \frac{1}{2} \vec{w} \cdot \vec{w} + \frac{\gamma}{2} \vec{\xi} \cdot \vec{\xi} \\ \text{s.t. } y_k (\vec{w} \cdot \vec{x}_k + b) = 1 - \xi_k, \quad \xi_k \geq 0, \quad k = 1, \dots, M \end{aligned}$$

ξ_k: slack variables
γ: amount of penalty

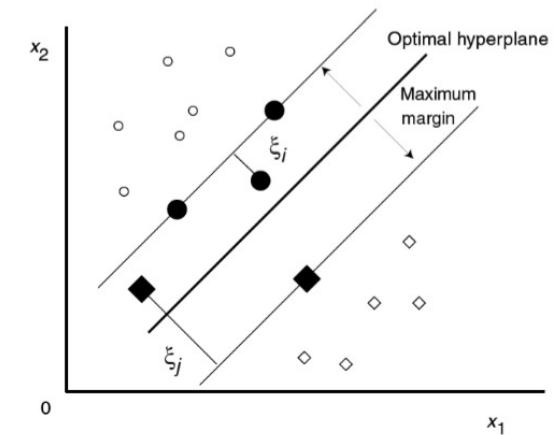
- KKT condition:

$$\vec{w} - \sum_{k=1}^M \alpha_k \vec{x}_k = 0 \quad \sum_k \alpha_k = 0 \quad \xi_k = \frac{1}{\gamma} \cdot y_k \alpha_k$$

$$\vec{w} \cdot \vec{x}_k + b = y_k(1 - \xi_k) \text{ or } \alpha_k = 0$$

- Finding the SVM is a quadratic programming, with complexity $O\left(\log\left(\frac{1}{\epsilon}\right) \text{poly}(N, M)\right)$.

↑
Not very efficient!



Soft-margin SVMs

- Matrix form:

$$F \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix} \quad F := \begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & \gamma^{-1}I + K \end{pmatrix}$$

- Here K is the kernel matrix, defined by $K_{ij} = \vec{x}_i \cdot \vec{x}_j$.
- Goal:
Given \vec{y}, γ , and $\{\vec{x}_i\}$ (and thus K),
we want to derive $\vec{\alpha}, b$, which can be used to construct the classifier as
 $g(\vec{x}) = \vec{w} \cdot \vec{x} + b$ with $\vec{w} = \sum_i \alpha_i \vec{x}_i$.
- Solving soft-margin SVMs → Solving linear systems (does it ring a bell?)

Quantum SVM

- Task: solve the linear system:

$$F \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix} \quad F := \begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & \gamma^{-1}I + K \end{pmatrix}$$

- Steps:

1. Construct quantum states given classical data
2. Perform HHL on F and $\begin{pmatrix} 0 \\ \vec{y} \end{pmatrix}$

- Performance:

Quantum SVM complexity = $O(\log(NM))$, which is **exponentially lower than the classical counterpart** (which requires solving a quadratic programming)!

Exercise: Verify the “w.l.o.g.” claim in the Hermitian assumption.

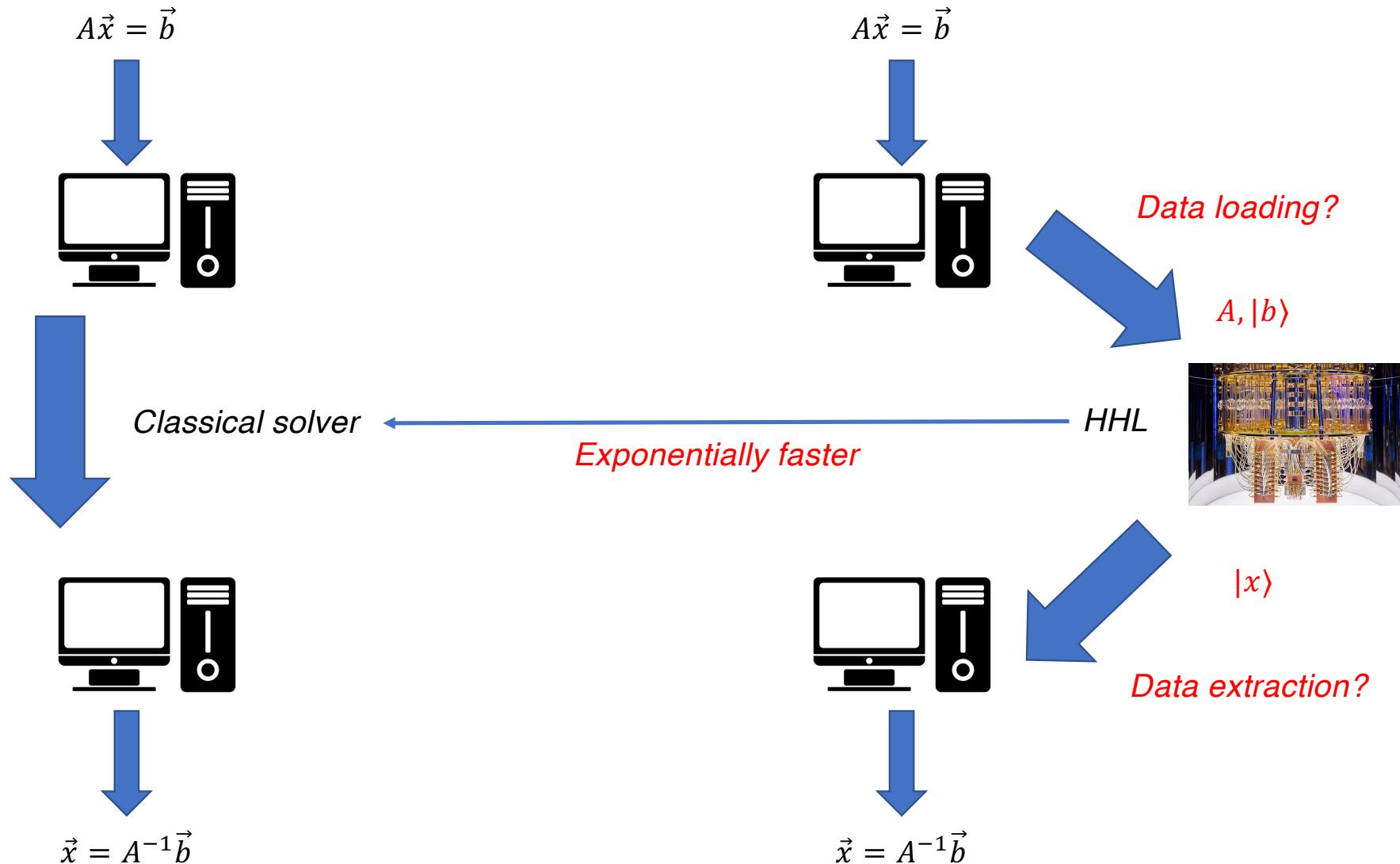
Part IV^{*}:

Loading classical data on a

quantum computer

* Bonus content

Classical solvers vs HHL



Data loading in HHL

- HHL offers quantum acceleration of a fully classical task $A\vec{x} = \vec{b}$.
The classical data need to be loaded on to the quantum computer, e.g.,

$$\vec{b} \mapsto \frac{1}{\|\vec{b}\|} \sum_i b_i |i\rangle.$$

- The output of HHL is a quantum state $|x\rangle$, which must be measured if we want to know x_i for each i .
- (Lazy) remedy: assume that $|b\rangle$ can be given as a quantum input.
- Otherwise, the cost of data loading should be accounted for, when evaluating the efficiency of HHL compared to classical solvers.
- This requires a notion of **quantum memory**.

QRAM

- Quantum Random Access Memory:

$$\sum_j c_j |j\rangle_A \otimes |0\rangle_D \xrightarrow{\text{QRAM}} \sum_j c_j |j\rangle_A \otimes |D_j\rangle_D$$

Address register *Data register* *Arbitrary amplitude*

- We also require the ability to reset it:

$$\sum_j c_j |j\rangle_A \otimes |D_j\rangle_D \xrightarrow{\text{QRAM}^{-1}} \sum_j c_j |j\rangle_A \otimes |0\rangle_D$$

- Let's now focus on $|b\rangle$ and see how QRAM contributes to its preparation.

Data loading with QRAM [Grover, Rudolph'02]

- Task: prepare $|b\rangle = \frac{1}{\|\vec{b}\|} \sum_i b_i |i\rangle$ given the vector $\{b_i\}_{i=1}^N$.
- Procedure:

1. Initialization: $|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle_A \otimes |0\rangle_D$.

2. Apply QRAM on $\{b_i\}$: $|\psi_2\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle_A \otimes |b_i\rangle_D$

3. Add an ancilla $|0\rangle_a$ and perform on it $R_i = \begin{pmatrix} \sqrt{1 - \frac{b_i^2}{b_{max}^2}} & \frac{b_i}{b_{max}} \\ -\frac{b_i}{b_{max}} & \sqrt{1 - \frac{b_i^2}{b_{max}^2}} \end{pmatrix}$ conditioned on the value b_i of the data register.

$$|\psi_3\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle_A \otimes |b_i\rangle_D \otimes \left(\sqrt{1 - \frac{b_i^2}{b_{max}^2}} |0\rangle + \frac{b_i}{b_{max}} |1\rangle \right)$$

Succeed w. prob. $\geq \left(\frac{b_{min}}{b_{max}}\right)^2$

- 4. Measure ancilla in $|0\rangle, |1\rangle$. Succeed when getting $|1\rangle$: $|\psi_4\rangle = \frac{1}{\sqrt{N}} \sum_i \frac{b_i}{\|\vec{b}\|} |i\rangle_A \otimes |b_i\rangle_D$
- 5. Uncompute data register by QRAM^{-1} .

Steps 3-4 = rejection sampling!

How QRAM complexity affects HHL

$$\kappa' = b_{max}/b_{min}$$

- Conclusion: We can **load classical data efficiently**, with complexity $O(\kappa'^2)$, if we are **given the QRAM primitive**.

- Overall complexity of solving linear system with HHL

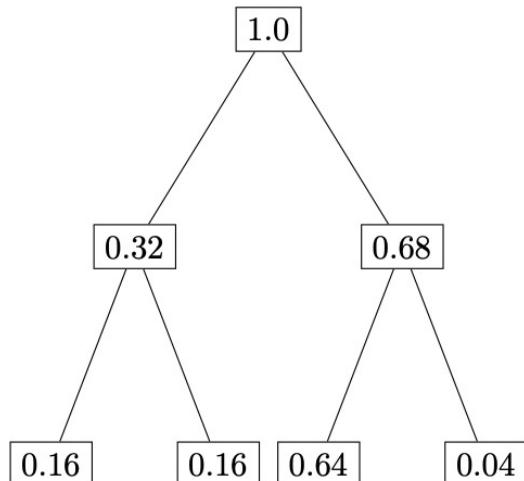
$$C_{HHL} = O(C_{QRAM}) + O(\log(N))$$

- The exponential speedup is preserved if $C_{QRAM} = O((\log N)^a)$ for some $a > 0$.

- HHL remains as **powerful**, if there is an **efficient realization of QRAM**.
- HHL is **not so powerful** as we thought, if $C_{QRAM} = O(\text{poly}(N))$ or there is any **caveat** for QRAM.

Implementation of QRAM

- **Question:** What is the requirement and complexity of implementing QRAM?
- There are various proposals ...
Let us check out one exemplary proposal (Kerenidis-Prakash'16) that achieves $O(\log N)$ complexity, given **free access** to a tree-shaped classical data structure:



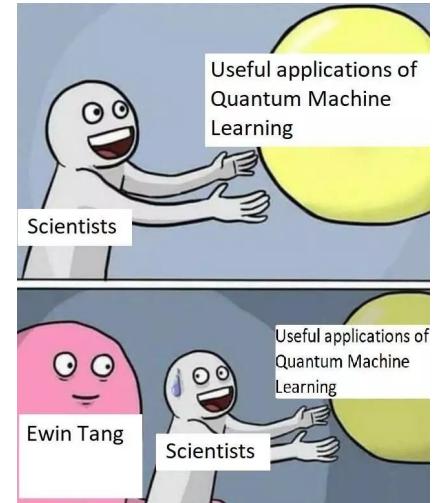
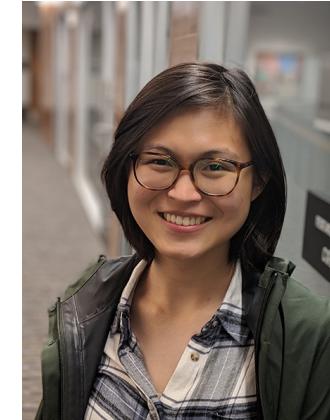
Let $|\phi\rangle = 0.4|00\rangle + 0.4|01\rangle + 0.8|10\rangle + 0.2|11\rangle$.

- Rotation on qubit 1:
 $|0\rangle|0\rangle \rightarrow (\sqrt{0.32}|0\rangle + \sqrt{0.68}|1\rangle)|0\rangle$
- Rotation on qubit 2 conditioned on qubit 1:

$$\begin{aligned} & (\sqrt{0.32}|0\rangle + \sqrt{0.68}|1\rangle)|0\rangle \rightarrow \\ & \sqrt{0.32}|0\rangle \frac{1}{\sqrt{0.32}}(0.4|0\rangle + 0.4|1\rangle) + \\ & \sqrt{0.68}|1\rangle \frac{1}{\sqrt{0.68}}(0.8|0\rangle + 0.2|1\rangle) \end{aligned}$$

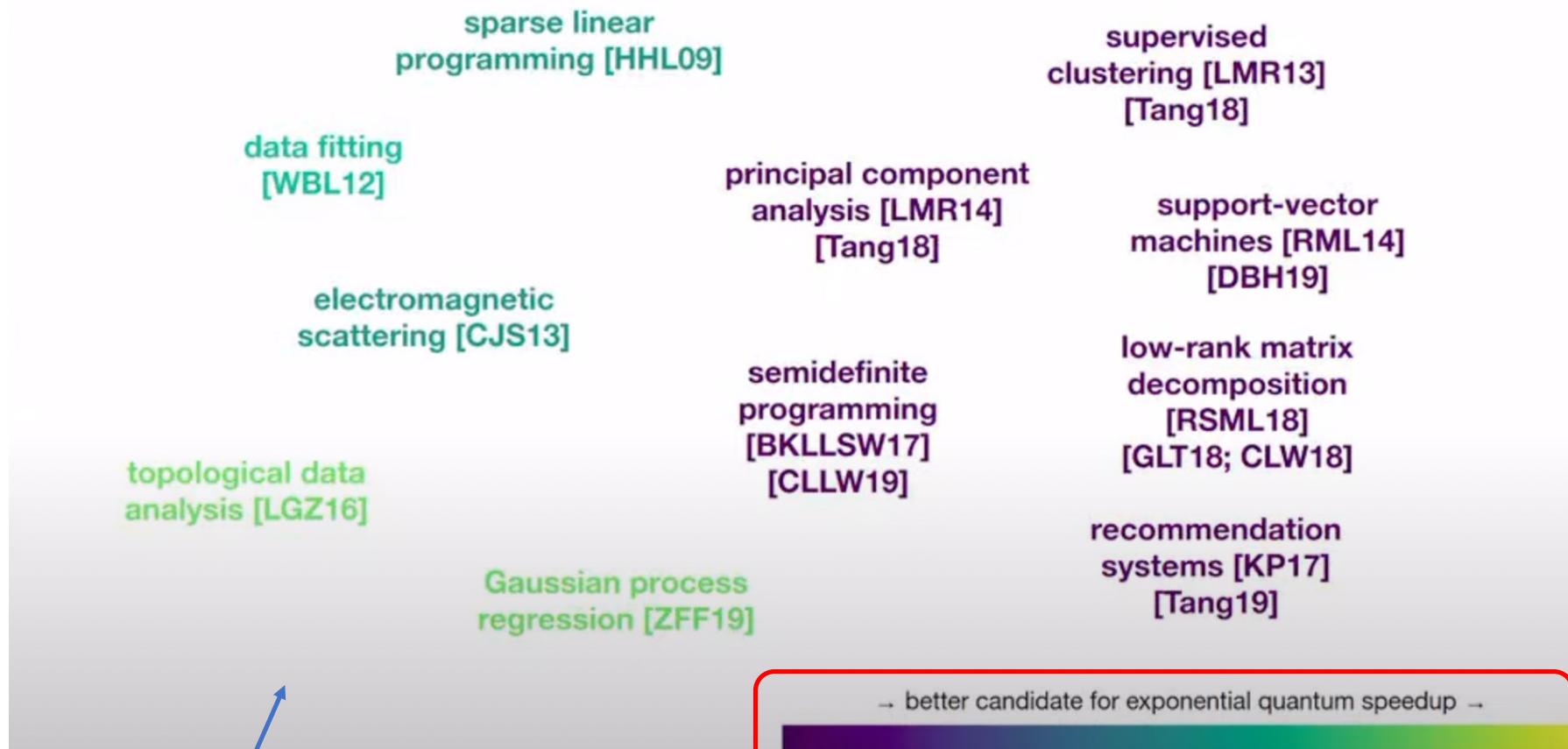
”Quantum-inspired” classical solver

- (Ewin Tang) In the previous implementation of QRAM, it is required (via the “tree”) that we have **Sampling** + **Query** access to \vec{b} :
 1. Sample $j \in \{0, \dots, N - 1\}$ with probability $\frac{|b_j|^2}{||\vec{b}||^2}$.
 2. Query each b_j and $||\vec{b}||$.
- (Shao-Montanaro) If we have S + Q access to \vec{b}, A . There is a **classical** algorithm that returns \vec{x}' with $||\vec{x}' - A^{-1}\vec{b}|| \leq \epsilon ||A^{-1}\vec{b}||$ in $O(\text{polylog}(N))$ time.
- Quantum algorithms are only polynomially faster than quantum-inspired algorithms.
- Still **open to debate**:
We could, e.g., find implementations of QRAM that are efficient and do not require SQ.



Ewin Tang

Landscape: exponential speedups in quantum machine learning



Screenshot from Ewin Tang's talk

Summary

- HHL - an exponentially faster quantum solver for linear systems.
- Rejection sampling & uncomputing.
- Applications of HHL (quantum SVM).
- Caveat of HHL: QRAM and quantum-inspired algorithms.

Homework

- **Review** the lecture slides; you may find the review questions in the next slides helpful.

Try the exercises in the slides and discuss with your classmates.

- Optional: Read the Qiskit tutorial on HHL (VPN required)
https://qiskit.org/textbook/ch-applications/hhl_tutorial.html.

Review questions

- Is it necessary to invert A when solving a linear system?
Do we get A^{-1} in HHL algorithm?
- Why do we need to uncompute in HHL and when loading classical data?
- Given a quantum circuit, how to construct a circuit that inverts the given circuit?
- What is a QRAM? How could it be implemented?
- Why do we say that the QRAM caveat may ruin HHL?