Lecture 15 - Quantum algorithms for linear algebra

Outline

- block encodings
- linear combination of unitaries
- Harrow, Hassidim, Lloyd '09 algorithm
- generalizations

Block encodings

We've seen that linear algebra plays an important role in Q.C. States are vectors, operations are matrices. Can we encode general vectors and matrices into quantum states and unitary operations?

Yes! However, depending on how we do it

we might have to pay a price in either computational resources (specifically runtime) or precision.

Encoding vectors

This will mostly recap what you had to do in HW4

Let $x \in \mathbb{R}^N$, we want to create

$$|\varphi(x)\rangle = \sum_{i=0}^{N-1} \frac{x_i}{\|x\|_2} |i\rangle$$

where $|\varphi(x)\rangle$ is on $\lceil \log_2 N \rceil$ qubits.

(note: we'll assume N is a power of 2; if this is not the case, we can simply pad our input vector with 0's to the nearest power of 2)

(another note: we're assuming $x \neq 0$, since we can't encode the zero vector)

E.g.: $x = (1, 2, 3)$

First we pad x to make it of length 4

↳ $x = (1, 2, 3, 0)$

Compute the L_2 norm of x

$$\|x\|_2 = \sqrt{1 + 2^2 + 3^2} = \sqrt{14}$$

⇒ The state we should prepare is

$$|\psi(x)\rangle = \frac{1}{\sqrt{14}}|00\rangle + \frac{2}{\sqrt{14}}|01\rangle + \frac{3}{\sqrt{14}}|10\rangle$$

In vector form $|\psi(x)\rangle = \frac{1}{\sqrt{14}} \begin{pmatrix} 1 & |00\rangle \\ 2 & |01\rangle \\ 3 & |10\rangle \\ 0 & |11\rangle \end{pmatrix}$

We can do this recursively

encode(x, qs)

if $len(x) = 1$

Done

else

$X_L = X[0 \,..\, len(x)/2 - 1]$

$X_R = X[len(x)/2 \,..\, len(x) - 1]$

$\theta = 2 \cdot \arccos\left(\frac{\|X_L\|}{\|X\|}\right)$

(if $\|x\| = 0$ take $\theta = 0$)

Apply $R_Y(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ -\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$

to $qs[0]$

$CTRL_0\text{-}encode(X_L)(qs[0], qs[1..])$

$CTRL_1\text{-}encode(X_R)(qs[0], qs[1..])$

operations are controlled on 0 and 1 respectively.

Idea is if we can recursively prepare $|\varphi(x)\rangle$ then we just need to combine 2 such states:

$|\varphi(x_L)\rangle$ and $|\varphi(x_R)\rangle$

$$|\varphi(x)\rangle = \frac{\|x_L\|}{\|x\|}|0\rangle|\varphi(x_L)\rangle + \frac{\|x_R\|}{\|x\|}|1\rangle|\varphi(x_R)\rangle$$

Can get this from $\frac{\|x_L\|}{\|x\|}|0\rangle + \frac{\|x_R\|}{\|x\|}|1\rangle$ using controlled versions of encode

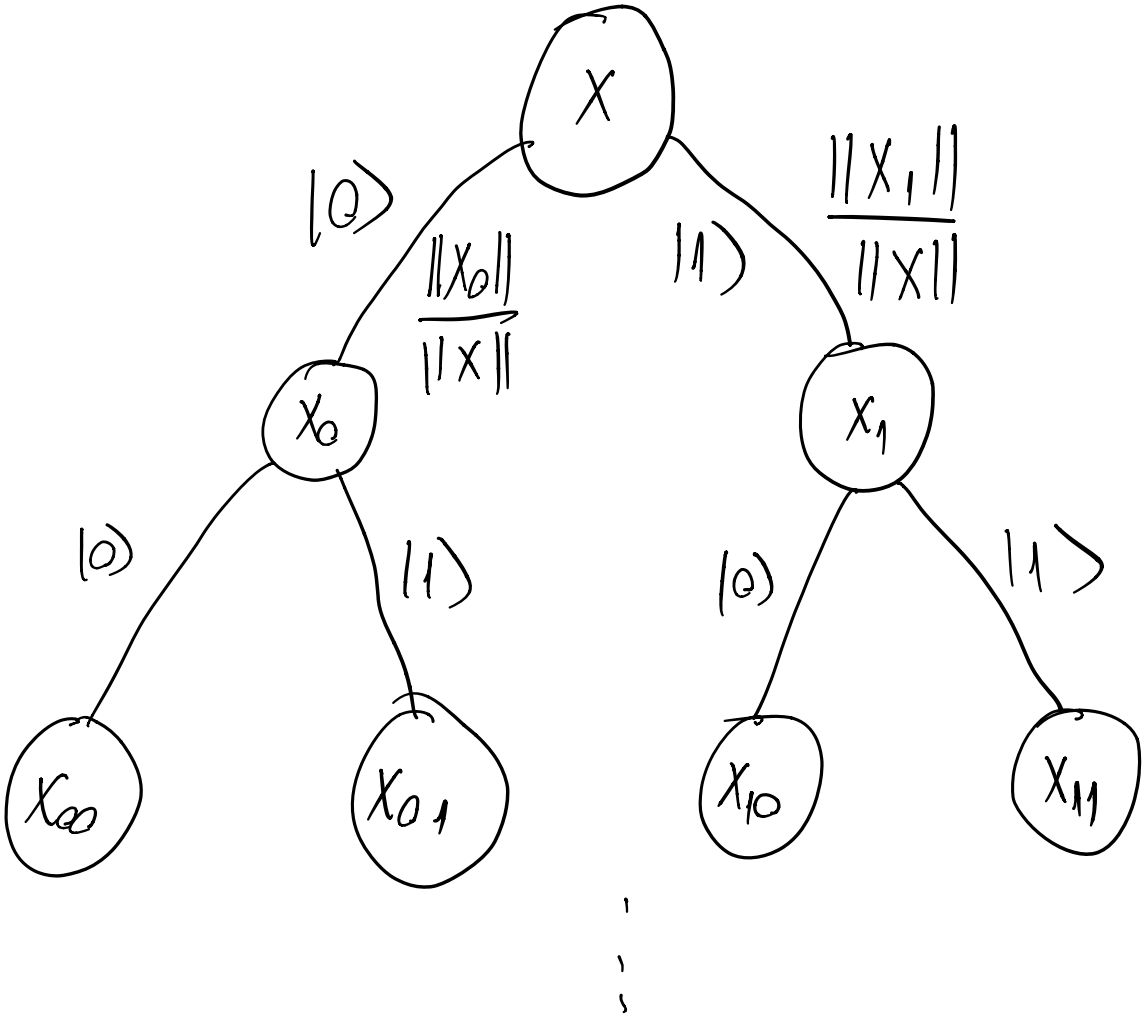
Schematically, suppose we label components of x by their binary representations

$$x = (x_{00\ldots0}, x_{00\ldots1}, \ldots x_{11\ldots1})$$

Let $x_0 = (x_{00\ldots0}, \ldots x_{011\ldots1})$

$$x_1 = (x_{10\ldots0}, \ldots x_{111\ldots1})$$

Same for X_{00}, X_{01} etc (prefix notation)



What about the signs of the X_i components?

With the above approach we're only preparing

$$|\varphi(|X|)\rangle = \sum \frac{|X_i|}{\|X\|_2} |i\rangle$$

To get the signs as well we can perform a

useful trick inspired by the oracle algorithms we studied.

Consider the operation

$$C_j Z = |j\rangle\langle j| \otimes Z + (I - |j\rangle\langle j|) \otimes I$$

This is the controlled on j Z operation.

$$C_j Z\, |i\rangle|\psi\rangle = \begin{cases} |i\rangle Z|\psi\rangle & \text{if } i=j \\ |i\rangle|\psi\rangle & \text{otw} \end{cases}$$

Suppose $|\psi\rangle = |1\rangle$. We know that $Z|1\rangle = -|1\rangle$

So $$C_j Z\, |i\rangle|1\rangle = (-1)^{\delta_{ij}} |i\rangle|1\rangle$$

($\delta_{ij} = 1$ if $i=j$ on 0 otw is the Dirac delta function)

After preparing $|\varphi(|x|)\rangle$ we can simply add an

ancilla qubit initialized as $|1\rangle$. We then perform $C_j Z\, |\varphi(|x|)\rangle|1\rangle$ for all j for which $x_j < 0$. The resulting state will be $|\varphi(x)\rangle$!

Encoding matrices

Given a matrix A, can we encode A into a unitary U?

In other words have something like this:

$$U = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}$$

where $A \in \mathbb{C}^{N \times N}$ and $\alpha \geq \|A\|$

↳ operator norm of A (largest singular value)

Remarkably this is possible and in time that scales like polylog (N). However, there are several caveats

- It's only possible for certain matrices; particularly sparse matrices or linear combinations of unitaries (LCUs). We'll only look at the latter case

- Assumes same quantum data-structure to store entries of A and be able to query them in superposition (QRAM)

$$O_1 |i\rangle|j\rangle|x\rangle \to |i\rangle|j\rangle|A_{i,j} \oplus x\rangle$$

$$O_2 |i\rangle|j\rangle \to |i\rangle|f(i,j)\rangle$$

$f(i,j)$ - column index for j'th non-zero entry in row i.

Linear combination of unitaries (LCU)

Suppose $A = \sum_{i=0}^{m-1} \alpha_i U_i$. How do we block encode A?

Denote vector of coefficients as $\alpha = (\alpha_0 \dots \alpha_{m-1})$

Suppose we have $P_{\sqrt{\alpha}} |0 \dots 0\rangle = |\psi(\sqrt{\alpha})\rangle$

↑ encoding unitary from before

Also consider $C\text{-}U = \sum_{i=0}^{m-1} |i\rangle\langle i|_A \otimes U_{i_B}$

Consider what happens if we perform

$$P_{\sqrt{\alpha}_A}^{\dagger} \; C\text{-}U_{AB} \; P_{\sqrt{\alpha}_A}$$

Let's look at the top left block of this unitary

$$\langle 0 \dots 0|_A P_{\sqrt{\alpha}_A}^{\dagger} \; C\text{-}U_{AB} \; P_{\sqrt{\alpha}_A} |0 \dots 0\rangle_A =$$

$$= \sum_j \langle j | \sqrt{\alpha_j} \cdot \left(\sum_i |i\rangle\langle i| \otimes U_i \right) \cdot \sum_k \sqrt{\alpha_k} |k\rangle$$

$$= \sum_{i,j,k} \underbrace{\langle j | i \rangle}_{\delta_{ij}} \underbrace{\langle i | k \rangle}_{\delta_{ik}} \sqrt{\alpha_j \alpha_k} \cdot U_i =$$

$$= \sum_i \alpha_i U_i = A$$

(note: this will not also encode the signs of the α_i's however we can get those as well with the same trick we used for vector encoding)

Runtime of this is $O(m)$. If $m \ll N$, we have an efficient way of block-encoding matrices.

Note that in principle any matrix could be encoded this way. This is because the set of Pauli matrices is a basis for all matrices.

$\forall A, \exists \alpha_i$ s.t.

$$A = \sum_i \alpha_i P_i$$

↳ log N-qubit Pauli operation

However, the set of n-qubit Paulis has size 4^n. Hence the overhead for preparing an arbitrary A would scale like N^2.

Finally note that having 2 block encodings, one of A and one of B, we can also get $A \cdot B$ and $A+B$.

For $A+B$ use same idea as LCU. For $A \cdot B$ take product of unitaries.

$$\begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \cdot \begin{pmatrix} B & \cdot \\ \cdot & \cdot \end{pmatrix} = \begin{pmatrix} A \cdot B & \cdot \\ \cdot & \cdot \end{pmatrix}$$

HHL (linear systems solver)

Suppose we have $|\psi(x)\rangle$ and a block encoding of A. HHL is an algorithm for obtaining $|\psi(A^{-1}x)\rangle$ or an approx of it to precision ε in time

$$\text{poly}(\log N, \log 1/\varepsilon, K)$$

where K is the condition number of A; $K = \frac{\sigma_{max}}{\sigma_{min}}$

σ - singular value of A.

"Solving a system of linear equations exp faster than classical algorithms".

Same remarks

- Not actually getting the solution vector $A^{-1}x$ but the state $|\psi(A^{-1}x)\rangle$. Would have to repeat

$O(N)$ times to get $A^{-1} \cdot x$

— Makes the rather strong assumption that we're given encodings of A and x. Where do these come from?

— Problem could became classically easy depending on how input and output are represented

— For sparse A can be shown that performing HHL is BQP-complete

— For low-rank A ($\text{rank}(A) = O(\log(N))$) performing HHL is in BPP! (breakthrough result of Ewin Tang)

— Need to assume an upper bound on K, since we can't compute it efficiently. Must be that $K = \text{polylog}(N)$ to maintain quantum advantage

(for d-sparse A, runtime is

$$\text{poly}(\log N, \log 1/\varepsilon, \kappa, d)$$

)

We're going to look at a version of HHL from Childs, Kothari, Somma '17.

Idea: Given that we know how to do linear combinations and powers of encoded matrices, use that to perform a series expansion of A^{-1} (truncated so as to have precision ε)

Two ways to do this

- Fourier series
- Chebyshev polynomials

We're going to look at the first version, but

the second is similar.

Let

$$h(x) = \frac{i}{\sqrt{2\pi}} \sum_{j=0}^{J-1} \Delta_y \sum_{k=-K}^{K} \Delta_z z_k e^{-z_k^2/2} \cdot e^{-ixy_j z_k}$$

Can be shown that $\left| h(x) - \frac{1}{x} \right| \leq \varepsilon$, for $x \in [-1, -1/K] \cup [1/K, 1]$, where

$$y_j = j \cdot \Delta_y \qquad z_k = k \cdot \Delta_z$$

$$\Delta_y = \theta\left(\varepsilon / \sqrt{\log(K/\varepsilon)}\right), \quad \Delta_z = \theta\left(1/K \cdot \sqrt{\log(K/\varepsilon)}\right)$$

$$J = \theta\left(\frac{K}{\varepsilon} \log(K/\varepsilon)\right)$$

$$K = \theta\left(K \log(K/\varepsilon)\right)$$

The formula for $h(x)$ is obtained from approximating an integral formula for $1/x$ with finite sums.

Specifically, the integral

$$\frac{1}{x} = \frac{i}{\sqrt{2\pi}} \int_0^{\infty} dy \int_{-\infty}^{+\infty} dz \; z \cdot e^{-z^2/2} \cdot e^{-ixyz}$$

Using the LCU ideas, we perform $h(A)$. Note that this requires us to do $e^{i\theta A}$, but we know how to do this from the lectures on quantum simulation! Of course, this requires that A is expressed as a linear combination of Hermitian operators, but that's not an issue (all Paulis are Hermitian, for instance).

$\Rightarrow$ we end up with $\begin{pmatrix} h(A) & \cdot \\ \cdot & \cdot \end{pmatrix} \approx_{\varepsilon} \begin{pmatrix} A^{-1} & \cdot \\ \cdot & \cdot \end{pmatrix}$

If the unitary encoding A^{-1} is U, we require $\langle 0\ldots 0|U|0\ldots 0\rangle$ to get A^{-1}.
This means we need to post-select on outcome $|0\ldots 0\rangle$.

Applications and generalizations

- many ML applications though not clear if there's a q. advantage (PCA, recommendation systems)
- non-ML applications: amplitude amplification, q simulation without phase estimation or trotterization, solving systems of differential equations, computing scattering cross-sections for electrodynamics etc

– We considered a series expansion of $1/x$, but this will work for pretty much any other well behaved function.

→ general way of applying $f(A)$, for same function f (amplitude amplification $f(s)=1$, $f(x_{x\neq s})=0$, least squares solver etc)

– testing differences between solutions of linear systems (SWAP test $|\varphi(A^{-1}x)\rangle$, $|\varphi(B^{-1}x)\rangle$)