Quantum Computing: Foundations to Frontier Fall 2018

Lecture 6

Lecturers: Peter Wittek, Henry Yuen *Scribes: Michael Tisi, Yuyang Liu*
David Ledvinka, Robert Lang

1 Overview

In the last lecture we made the transition from discussing the foundations of quantum computing to discussing the frontiers, beginning with Grover's Search Algorithm, Hamiltonians, and open quantum systems. In this lecture we explored the topic of quantum machine learning. At the beginning of this lecture, Peter Wittek from the Rotman School and Creative Destruction Lab introduced the foundations of quantum machine learning and defined realistic expectations of the subject. Afterwards, Henry Yuen lectured about the HHL algorithm.

2 Quantum Machine Learning

Peter Wittek's lecture on quantum machine learning began by defining what classical machine learning is before transitioning to how quantum physics is incorporated into the subject. He first mentioned that quantum machine learning is a very heated subject, as it encompasses both quantum computing and machine learning. He stated that his goal was to define what one could realistically expect from quantum machine learning in its current state.

2.1 Classical Machine Learning

In essence, classical machine learning can be defined by four distinct key elements:

1. Optimization
2. Identifying patterns and data
3. Making predictions
4. Training.

The term *machine learning* was originally coined in the late 1950's as a means to describe a field of artificial intelligence and statistical inference.

2.2 The Roots of Classical Machine Learning

As stated previously, classical machine learning has its roots in the following two fields:

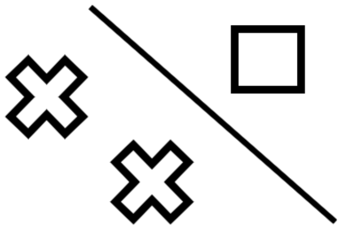
1. Artificial intelligence
2. Statistical inference.

With respect to classical machine learning, **artificial intelligence** is a means of codifying knowledge as logic, and **statistical inference** is observing how some parameter, θ, scales with a number of samples, N.

From artificial intelligence, many complexities arise. In particular, Peter Wittek talked about **model complexity**.

2.3 Model Complexity and the Vapnik-Chervonenkis (VC) Theorem

One particular kind of complexity in classical machine learning discussed by Peter Wittek was model complexity. In particular, if there is a model consisting of two different kinds of points, those points can be separated by a hyperplane, as shown below.

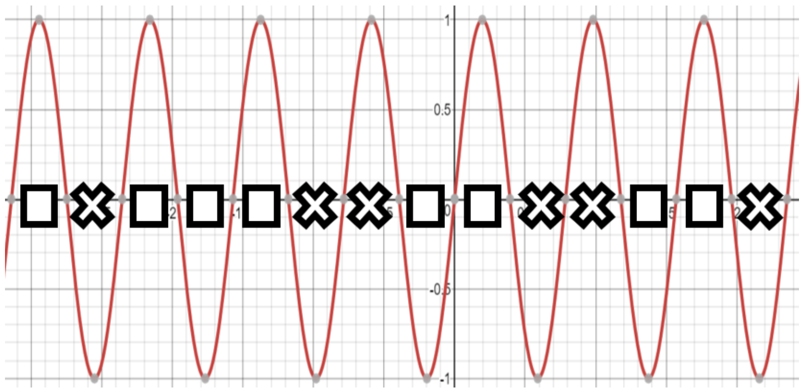


Here, we can see that the two crosses are separated from the square by a single hyperplane, represented by the diagonal line. The complexity of this model, also called the **VC dimension**, can be expressed as follows:

$$h(hyperplane) = d + 1$$

where d is the number of free variables.

Note that, for a model consisting of an arbitrarily large number of points, a sine function with a correspondingly large frequency can be chosen as a hyperplane such that it separates each of the points. This concept is illustrated in the graphic shown below.



Note that each wavelength of the sine function separates two points. It follows that the VC dimension of a sine function with arbitrary frequency α is:

$$h(\sin \alpha x) = \infty$$

The concepts above serve as a foundation for the **Vapnik-Chervonenkis (VC) Theorem**, which defines a probabilistic upper bound on the test error of a classification model. Mathematically, the VC Theorem can be expressed as follows:

$$P(E \leq E_N + \sqrt{\frac{h(\log \frac{2N}{h} - \log \frac{\eta}{4}}{N}}) \geq 1 - \eta$$

where E is the test error, E_N is the training error, N is the sample size, and η is a parameter such that $0 < \eta < 1$.

For a sample, $S = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}$, $y_i \in \{0, 1\}$, we can define the training error as follows:

$$E_N = \frac{1}{N} \sum_{i=1}^{N} (\hat{f}(x_i) - y_i)^2$$

where $\hat{f}(x_i)$ is an arbitrary function of x_i.

In closing of the subject of complexity, Peter Wittek stated that classical machine learning is a "balancing act" between three aspects of complexity:

1. Computational complexity
2. Sample complexity
3. Model complexity.

Wittek also stated that state of the art machine learning comes from principles of deep learning, such as:

- Supervised learning
- Discriminative problems
- Determinism, and
- Parallelizability of neural networks.

3 Paradigms of Quantum Machine Learning

Peter Wittek discussed three paradigms of quantum computing: the **Gate Model**, **Quantum Annealing**, and **Quantum Simulators**.

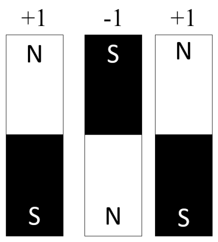
3.1 The Gate Model

The Gate Model paradigm of quantum machine learning is, in essence, what we'eve already been dealing with in this course. The Gate Model deals with qubits and examines photonic circuits which deal with qumodes.

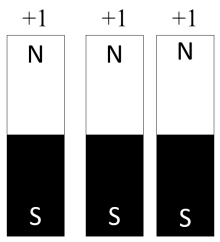
3.2 Quantum Annealing

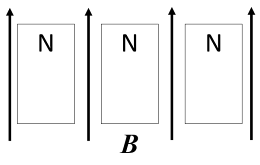
In essence, Quantum Annealing is a means by which a global minimum is found from a set of potential candidates for some object function. In a classical system, annealing can be represented by the following example.

Suppose you have three magnets, with their poles oriented as shown below.



One can apply an external magnetic field to the system such that, if it's strong enough, it can flip the polarity of the magnets.





As can be seen from the image above, the applied magnetic field, $\boldsymbol{B}$, is strong enough to flip the polarity of the middle magnet from -1 to $+1$.

The **Ising Model** can be used to model the total energy of the system. Mathematically, the Ising Model can be represented as follows:

$$\mathcal{H} = \sum_{i,j} \exists_{i,j} \sigma_i \sigma_j + \sum_i h_i \sigma_i$$

where $\mathcal{H}$ is the Hamiltonian and $\exists_{i,j}$ is the interaction between states σ_i and σ_j. The Ising Model looks to minimize $\mathcal{H}$ in order to get the lowest energy configuration.

When dealing with quantum mechanical systems, the Ising Model becomes:

$$\mathcal{H} = \sum_{i,j} \exists_{i,j} \sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^z$$

where the σ's are now replaced with σ^z's, which are Pauli Z matrices.

Recall:

$$\sigma^z |0\rangle = |0\rangle$$

and

$$\sigma^z |1\rangle = -|1\rangle.$$

In order to implement Quantum Annealing, let's start with the Hamiltonian in the ground state:

$$\mathcal{H}_0 = \sum_i \sigma_i^x$$

where σ^x is the Pauli X matrix.

Recall:

$$\sigma^x |0\rangle = |1\rangle$$

and

$$\sigma^x |1\rangle = |0\rangle.$$

Note that, in the ground state

$$\mathcal{H}_0 = |+\rangle_1 \otimes |+\rangle_2 \otimes \cdots \otimes |+\rangle_i.$$

In the first excited state, the Hamiltonian becomes

$$\mathcal{H}_1 = \sum_{i,j} \exists_{i,j} \sigma_i^z \sigma_j^z + \sum_i h_i \sigma_i^z$$

From the expressions for $\mathcal{H}_0$ and $\mathcal{H}_1$, we obtain the expression for the (simple) composite time-dependent Hamiltonian for idealized quantum computing:

$$H(t) = (1-t)\mathcal{H}_0 + t\mathcal{H}_1$$

where $t \in [0,1]$. The general expression for $H(t)$ can be written as follows:

$$H(t) = f(t)\mathcal{H}_0 + g(t)\mathcal{H}_1$$

where $f(t)$ and $g(t)$ are functions such that $f(t) + g(t) = 1$. The expression $f(t) + g(t) = 1$ is known as the **annealing schedule**.

Within $H(t)$ there are many excited states and ground states, however, the aim of quantum annealing is to find the smallest energy gap between an excited state and a ground state.

3.3 Quantum Simulators

Quantum Simulators are not as universal as the Gate Model or Quantum Annealing as quantum simulators are very difficult (if not, impossible) to implement without a working quantum computer.

4 Quantum Software Stack

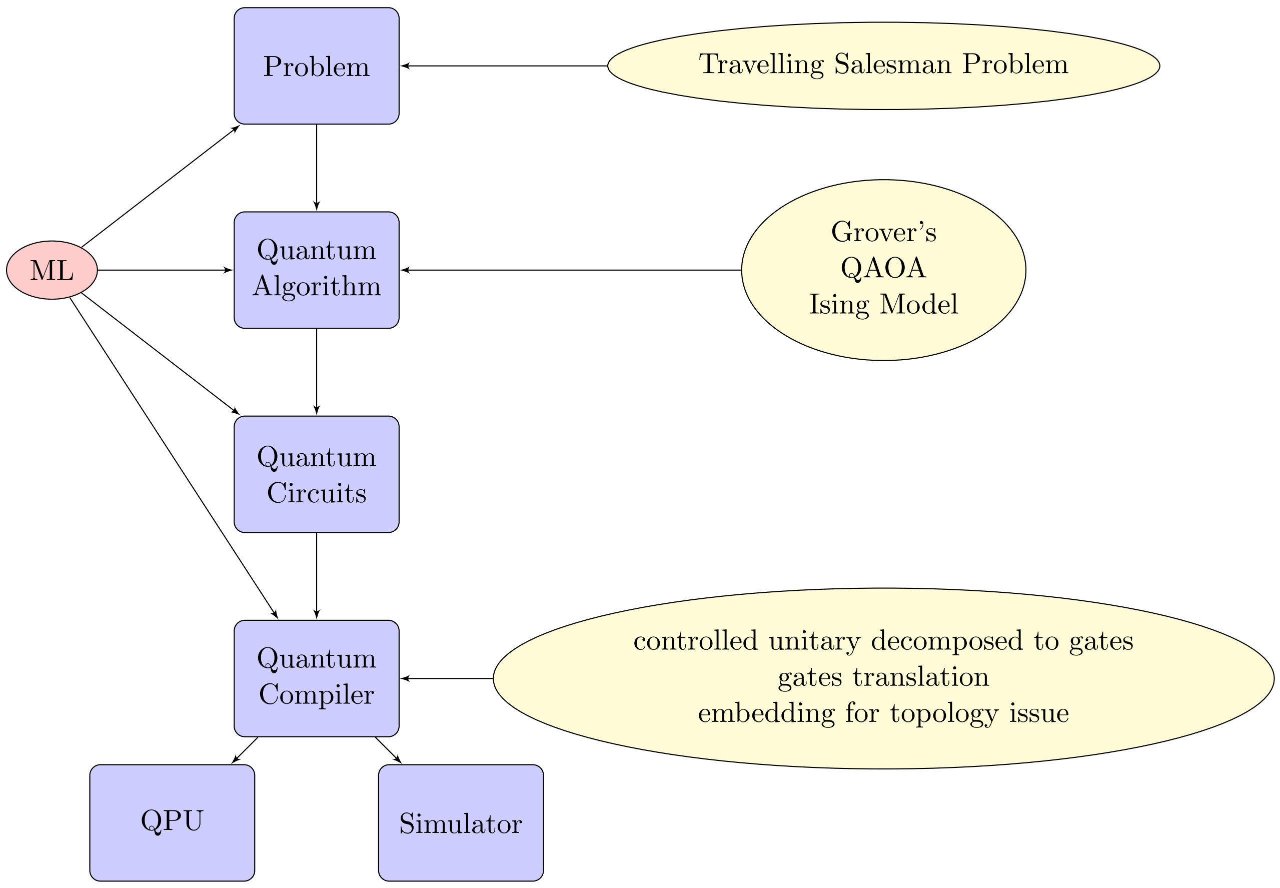


Figure 1: Quantum software stack and machine learning

When it comes to how the quantum software stack is integrated with machine learning, we can see from the figure above that machine learning techniques can be applied at each stage of quantum software stack to solve the challenges in individual domains. Thus, it became the Quantum Machine Learning.

There are still challenges in implementing these quantum software stacks, and we will be looking at a particular issue in the quantum compiler stage. In figure 2, with four qubits in a plane, you can fully connect all of them without crossing any two connections. However, by adding one more qubit to the design will cause a crossing to happen (the dotted line would always intercept with an existing connection) unless you begin to swap qubits. We could in fact swap the positions of qubits to avoid crossing by utilizing C-NOT gates. In the case of swapping 3 qubits, you would need (3+1+3) = 7 gates to realize it. In the current state of quantum computers, using 7 gates to

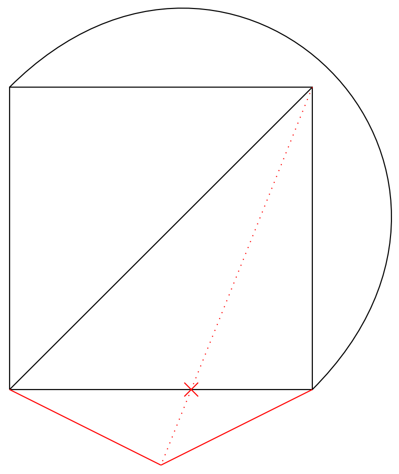


Figure 2: Superconducting share the same topology issue

swap 3 qubits takes up too many valuable resources.

5 Quantum Enhanced Machine Learning

We have two different protocols: the first protocol tries to convert between the classic data and quantum data, whereas the second protocol works fully within quantum world without conversions.

Protocol 1:

$$\text{classical data} \rightarrow \boxed{\text{Quantum stuff}} \rightarrow \text{classical data}$$

This protocol assumes a short time of period for the quantum stuff to run in the quantum computers.

Protocol 2:

$$\text{Q data} \rightarrow \boxed{\text{Quantum stuff}} \rightarrow \text{Q data}$$

Two examples for the quantum stuff in this protocol are exponential speedup and HHL core algorithm [1]. However, based on the paper "Bayesian Deep Learning on a Quantum Computer"[2], HHL is still unrealistic at the moment.

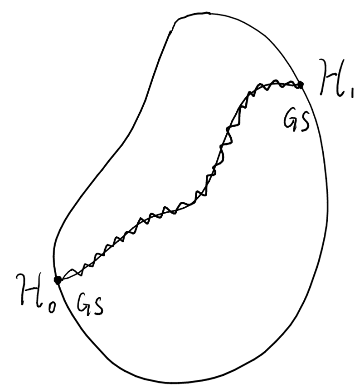
6 Optimization

The quantum annealing is one way to obtain the smallest energy gap between two states:

$$H = (1-t)\mathcal{H}_0 + t\mathcal{H}_1$$

where $t \in [0,1]$ We could always find a way of bridging between finding this smallest energy gap and finding the optimal solution for a hard classic question.

6.1 Quantum Approximate Optimization Algorithm(QAOA)



Let's assume we have a quantum state space, in which we have two Hamiltonians(describing only local effects) in the ground states: $\mathcal{H}_0$ and $\mathcal{H}_1$. They are describing two different sets of constraints in the system. By slowly adjusting in the ground state $\mathcal{H}_0$ and $\mathcal{H}_1$ under a discrete way, we could reach the state of the final Hamiltonians where an approximation exists.

We first define an alternating sequence of operations to our Hamiltonians $\mathcal{H}_0$ and $\mathcal{H}_1$. We are considering within a radius of p in the Hamiltonians:

$$P : U(\mathcal{H}_1) \dots U(\mathcal{H}_p)$$

and

$$U(\mathcal{H}_i) = U(\mathcal{H}_0, \beta_i)U(\mathcal{H}_1, \gamma_i)$$

where β and γ are angles to tell us how long to apply the above operators.

$$\mathcal{H}_0 = \sum \sigma_i^x$$

Researches on finding the optimal sequence of the operators are ongoing. Some work related to use gradient decent has been done. But the results are unclear.[4]

7 Quantum Machine Learning with Ensemble Methods

The ensemble method takes a bunch of ML classifiers that try to solve the same problem together, and aggregate them by assigning weights to each individual to make a prediction. QBoost [3] is an application to use a type of ensemble method and provides an option to improve QML.

8 Sampling

Assume we have the following system where interactions are happening between the system S and the environment E.

Thermal State:

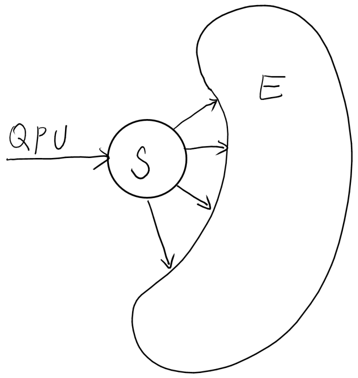
$$P(\mathcal{E}_i) = \frac{1}{\mathcal{Z}} e^{-\frac{\mathcal{E}_i}{T}}$$

where $\mathcal{E}_i$ is the energy, and T is the temperature.

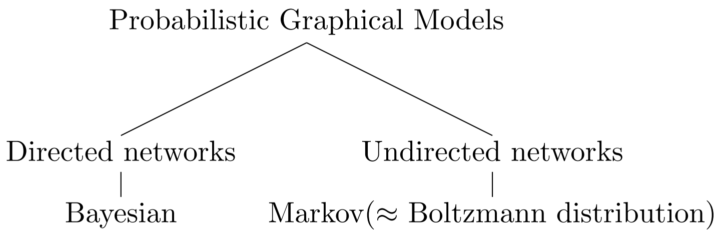
Partition function:

$$\mathcal{Z} = \sum e^{-\frac{\mathcal{E}_i}{T}}$$

We call this distribution the Boltzmann distribution.



9 Probabilistic Graphical Models


Probabilistic Graphical Models
Directed networks
Undirected networks
Bayesian
Markov(≈ Boltzmann distribution)

We show a special case for a Markov network where it is essentially the Boltzmann machine.

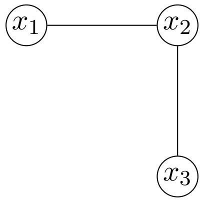


Figure 3: A simple Markov network

If we define the $x_i = \{+1, -1\}$, it becomes a Ising model and also a Boltzmann Machine.

10 Restricted Boltzmann Machine(RBM)

The figure represented a Restricted Boltzmann Machine. It consists of m visible units $\mathbf{V} = (V_1, V_2, \ldots, V_m)$ to represent observable data and n hidden units $\mathbf{H} = (H_1, H_2, \ldots, H_n)$ to be the latent variables. Each unit in the visible layer is fully connected to all the units in the hidden layer and vice versa. However, units in the same layer are not connected to each other, and this is why it is called the Restricted Boltzmann Machine. In terms of probability, without connections in the same layer makes the units independent from others.[5]

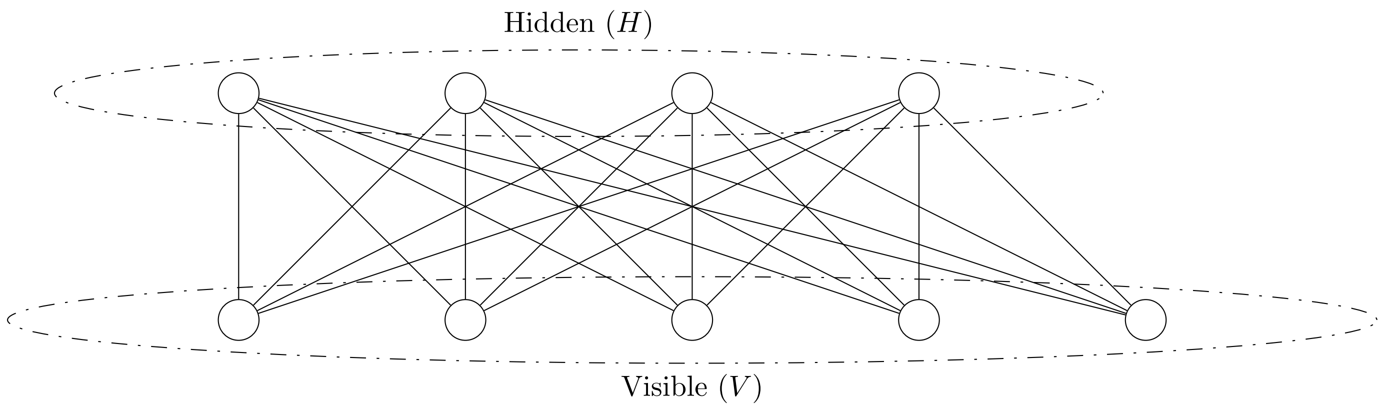


Figure 4: A restricted Boltzmann Machine

We have the probability distribution P:

$$P(\mathbf{V}) = \sum_{\mathbf{H}} P(\mathbf{H}, \mathbf{V}) = \sum_{\mathbf{H}} \frac{1}{\mathcal{Z}} e^{-\frac{\mathcal{E}(\mathbf{H},\mathbf{V})}{T}}$$

and the Partition function:

$$\mathcal{Z} = \sum e^{-\frac{\mathcal{E}_i}{T}}$$

and energy function $\mathcal{E}$.

$$\mathcal{E}(\mathbf{H}, \mathbf{V}) = \sum W_{ij} h_i v_j + \sum a_i v_i + \sum b_j h_j$$

where W_{ij} is the coupling, a weight between unit h_i and unit v_j. a_i is the field(bias) on v_i and b_j is the field(bias) on h_j.

11 The Harrow-Hassidim-Lloyd Algorithm

The Harrow-Hassidim-Lloyd (HHL) algorithm is an algorithm that “solves” exponentially large linear systems in polynomial time. The main idea behind HHL is the following: It looks like quantum computing is nature’s way of doing linear algebra with exponentially large vectors and matrices, so lets put that to use!

11.1 The Problem

Given a matrix $A \in \mathbb{R}^{N \times M}$, and a vector $b \in \mathbb{R}^N$, we can ask for a solution to the system of linear equations

$$Ax = b$$

that is an $x \in \mathbb{R}^M$ which satisfies the above equation, if such an x exists.

In our case, for simplicity we will assume that $N = M$ and A is invertible. Under this assumption

$$x = A^{-1}b$$

so the problem reduces to inverting A. To solve this problem using quantum computing we will want to represent both the input and outputs as quantum states. Thus we assume that b is given in the form $|b\rangle$ as a $\log N$ qubit state, and the goal is to produce an approximation of the following quantum state:

$$|x\rangle = \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|}$$

This is the solution to the linear system $Ax = b$ in "quantum form.

Solving systems of linear equations is one of the most basic problems you can imagine, as well as one of the most important. The HHL algorithm allows us to approximately "solve" this problem in polylog(N) time, given several assumptions are met. This is exponentially faster than any classical algorithm for solving linear systems (which must take at least poly(N) time), but this is not, strictly speaking, a completely fair comparison, because the way the data is represented. Furthermore, the HHL algorithm also requires a few assumption to be satisfied in order to achieve the polylogarithmic running time. Thus you win some, you lose some.

The first major assumption is that the vector $|b\rangle$ is given in quantum state form. How do we obtain such a quantum state? If the vector b is first presented classically, then one can in principle generate the $\log N$ qubit state $|b\rangle$, but this process would take at least $\Omega(N)$ time. Furthermore, we also assume that the HHL algorithm is able to access multiple copies of $|b\rangle$, which is necessary because of the No-Cloning Theorem.

Next, the fact that $|x\rangle$ is given in a quantum state means that we're not getting a solution to the linear system in the traditional sense; in general it is difficult to obtain information about $|x\rangle$ such as a fixed entry of the vector. If we measure $|x\rangle$ in the standard basis we will get one of N classical bit string states and won't be able to ascertain anything about the coefficients of $|x\rangle$. We could try to repeat the algorithm many times and measure the solution until we can obtain some statistical data about $|x\rangle$, but the number of times one would have to repeat the algorithm in order to obtain enough data to estimate a particular entry of $|x\rangle$ will, in general, void our exponential speedup (unless that particular entry had very large magnitude relative to the other entries).

But there is still hope that the algorithm can be used as a subroutine in another quantum algorithm. Generally when we are interested in solving linear systems of equations we don't just want the solution, but we want to do something with the solution. For example after using HHL perhaps we would want to compute $\langle x|G|x\rangle$ for some other matrix G – if G is an efficiently implementable measurement, then we could estimate this quadratic form efficiently.

This would be like the use of the Quantum Fourier Transform. Even though the Fourier transform is given as a quantum state in which it is difficult to extract any particular Fourier coefficient, it is still remarkably useful for solving problems like factoring integers. If the algorithm using HHL also has an efficient way to generate the $|b\rangle$ needed during HHL then the representation of the data is no longer an issue, and could in-fact be seen as a benefit.

11.2 The Set Up

Now we will list the assumptions we'll need for the algorithm to work

1. A is a Hermitian matrix (as well as square and invertible as previously assumed).
2. If $\lambda_1 \leq \ldots \leq \lambda_N$ are the eigenvalues of A, we assume that $\min_j |\lambda_j| \geq \alpha$, and $\max_j |\lambda_j| \leq 1$
3. The algorithm has access to many copies of $|b\rangle$ as a quantum state.
4. The algorithm is given the ability to implement the unitary e^{-iAt} for any value of t such that $0 \leq t \leq \text{polylog} N$.

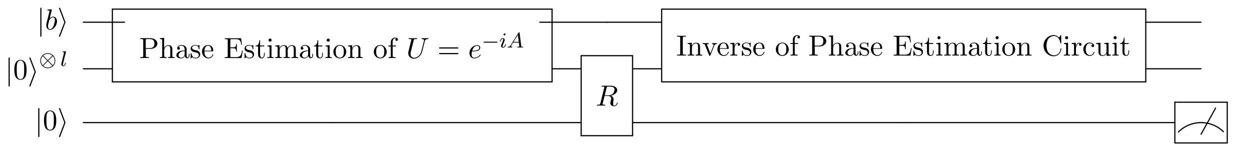
Assumption 1 can be relaxed and assumption 2 is reasonable, but assumptions 3 and 4 are quite strong. We will discuss the implications of these assumptions more after presenting and analyzing the algorithm.

For the algorithm we will need an algorithmic primitive which was discussed in lecture 4; phase estimation. To recall, given a unitary U and one of it's eigenvectors $|\psi_j\rangle$ in a quantum state we know that $U|\psi_j\rangle = e^{i\theta_j}U|\psi_j\rangle$ for some θ_j but we don't know what θ_j is. If we are able to apply controlled-U^{2^k} for a wide range of k then we can estimate the value of θ_j.

More precisely the phase estimation circuit takes the input state $|\psi_j\rangle|0\rangle^{\otimes l}$ (where $l = O(\log N)$) to the state $|\psi_j\rangle|\tilde{\theta_j}\rangle$, where $|\tilde{\theta_j}\rangle$ is length l binary expansion of an approximation of θ_j. For simplicity we will assume that phases estimation works exactly, and ignore approximation errors.

11.3 The Algorithm

The algorithm uses the following basic subroutine:


$|b\rangle$
$|0\rangle^{\otimes l}$
$|0\rangle$
Phase Estimation of $U = e^{-iA}$
R
Inverse of Phase Estimation Circuit

where $|b\rangle$ is n-qubit state. We call the first n qubits $|b\rangle$ the "eigenvector register", the next l qubits the "eigenvalue register" and the last qubit the "flag register".

The unitary R is defined by

$$R(|\lambda\rangle \otimes |a\rangle) = \begin{cases} |\lambda\rangle \otimes (R_\lambda |a\rangle) & \text{if } \lambda \geq \alpha \\ |\lambda\rangle \otimes |a\rangle & \text{otherwise} \end{cases}$$

where $|\lambda\rangle$ is an l-qubit state, and

$$R_\lambda = \begin{pmatrix} \sqrt{1 - \frac{\alpha^2}{\lambda^2}} & -\frac{\alpha}{\lambda} \\ \frac{\alpha}{\lambda} & \sqrt{1 - \frac{\alpha^2}{\lambda^2}} \end{pmatrix}$$

(i.e R_λ is just a rotation of the qubit by $\arcsin \frac{\alpha}{\lambda}$).

Now let's analyze this basic subroutine:

$$|\Psi_0\rangle = \underbrace{|b\rangle}_{n \text{ qubits}} \otimes \underbrace{|0\rangle^{\otimes \ell}}_{\ell \text{ qubits}} \otimes \underbrace{|0\rangle}_{1 \text{ qubit}}$$

We're going to perform phase estimation of the vector $|b\rangle$ with respect to the unitary e^{-iA}, we need to understand the eigenvectors and eigenvalues of it.

Recall the spectral theorem from last time.

$\exists$ orthonormal basis for $\mathbb{C}^N : \{|u_1\rangle, ..., |u_N\rangle\}$, and real eigenvalues: $\lambda_1 \leq ... \leq \lambda_N$ s.t.

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

By our assumption on A,

$$\min_j |\lambda_j| \geq \alpha > 0$$

and

$$\max_j |\lambda_j| \leq 1.$$

Then,

$$e^{-iA} = \sum_j e^{-i\lambda_j} |u_j\rangle\langle u_j|$$

and

$$A^{-1} = \sum_j \frac{1}{\lambda_j} |u_j\rangle\langle u_j|.$$

Returning to the phase estimation, by the orthonormality of the basis, we can write

$$|b\rangle = \sum_j b_j |u_j\rangle,$$

$$\text{s.t. } \sum_j |b_j|^2 = 1.$$

$$\begin{aligned} C|b\rangle|0\rangle^{\otimes \ell} &= C \sum_j b_j |u_j\rangle|0\rangle^{\otimes \ell} \\ &= \sum_j b_j C\big(|u_j\rangle|0\rangle^{\otimes \ell}\big) \\ &= \sum_j b_j |u_j\rangle|\lambda_j\rangle, \end{aligned}$$

where $|\lambda_j\rangle$ is a binary expansion of the eigenvalue λ_j. For simplicity, we're going to pretend this is exact and not an approximation. We can deal with these errors later.

Then,

$$|\Psi_1\rangle = \sum_j b_j |u_j\rangle|\lambda_j\rangle|0\rangle,$$

$$|\Psi_2\rangle = \sum_j b_j |u_j\rangle|\lambda_j\rangle \left(\sqrt{1 - \frac{\alpha^2}{\lambda_j^2}} |0\rangle + \frac{\alpha}{\lambda_j}|1\rangle \right).$$

At this point, we run the inverse of the phase estimation circuit. This is typically referred to as "uncomputing" in quantum information literature.

$$|\Psi_3\rangle = \sum_j b_j |u_j\rangle |0\rangle^{\otimes \ell} \left(\sqrt{1 - \frac{\alpha^2}{\lambda_j^2}} |0\rangle + \frac{\alpha}{\lambda_j} |1\rangle \right).$$

We now measure the last qubit.

$$= \left(\sum_j b_j \sqrt{1 - \frac{\alpha^2}{\lambda_j^2}} |u_j\rangle |0\rangle^{\otimes \ell} \right) |0\rangle + \left(\sum_j b_j \frac{\alpha}{\lambda_j} |u_j\rangle |0\rangle^{\otimes \ell} \right) |1\rangle$$

Partial measurement rule: Probability of obtaining outcome $|1\rangle$ is

$$\left|\left| \sum_j b_j \frac{\alpha}{\lambda_j} |u_j\rangle |0\rangle^{\otimes \ell} \right|\right|^2 = p.$$

This is already written as a linear combination of orthogonal vectors. So the norm is just sum of squares of the amplitudes in front:

$$\begin{aligned} &= \sum_j \left| b_j \frac{\alpha}{\lambda_j} \right|^2 \\ &= \sum_j |b_j|^2 \frac{\alpha^2}{\lambda_j^2} \\ &\geq \Big(\sum_j |b_j|^2 \Big) \alpha^2 = \alpha^2. \end{aligned}$$

Conditioned on outcome being $|1\rangle$, we get that the post measurement state is:

$$\begin{aligned} &\frac{1}{\sqrt{p}} \sum_j b_j \frac{\alpha}{\lambda_j} |u_j\rangle \otimes |0\rangle^{\otimes \ell} \otimes |0\rangle \\ &= \frac{\alpha}{\sqrt{p}} \Big(\sum_j \frac{b_j}{\lambda_j} |u_j\rangle \Big) \otimes |0\rangle^{\otimes \ell} \otimes |0\rangle \\ &= \frac{\alpha}{\sqrt{p}} \Big(\underbrace{A^{-1}|b\rangle}_{\text{our desired } |x\rangle!} \Big) \otimes |0\rangle^{\otimes \ell} \otimes |0\rangle. \end{aligned}$$

Thus, the Basic Subroutine, if when we measure the flag qubit we get the state 1 (which we get with prob $\geq \alpha^2$), we actually have the desired solution state in the first register! We have solved the problem.

If we measure 0, then we just throw everything away and try again. How many times do we have to repeat? $O(\frac{1}{\alpha^2})$ times.

If the ratio between the smallest eigenvalue and the largest eigenvalue of A is not too small, then we don't have to repeat that many times in order to get our solution vector with high probability.

The complexity of this problem is thus $O(\text{polylog } N \cdot \frac{1}{\alpha^2})$.

Now, let us return back to the assumptions we established. Some of them are not so bad.

1. **A is full rank and Hermitian**
   - Full rank: Kind of necessary if we want to guarantee that $Ax = b$ has a solution. If $|b\rangle$ is in the range of A then we can also drop this assumption.
   - Hermitian: Via a trick, we can assume this without loss of generality because solving $Ax = b$ is equivalent to solving:

$$\underbrace{\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}}_{\tilde{A}\in\mathbb{R}^{(N+M)\times(N+M)}} \tilde{x} = \underbrace{\begin{pmatrix} b \\ 0 \end{pmatrix}}_{\tilde{b}\in\mathbb{R}^{N+M}}$$

   where:

$$A \in \mathbb{R}^{N\times M}$$
$$x \in \mathbb{R}^{M}$$
$$b \in \mathbb{R}^{N}.$$

   Any solution $\tilde{x}$ to $\tilde{A}\tilde{x} = \tilde{b}$ will be such that $\tilde{x} = \begin{pmatrix} x \\ \vdots \end{pmatrix}$ has $Ax = b$.

2. **Bounds on the eigenvalues**
   - Why can we assume $\lambda_{\max}(A) \leq 1$? If not, just work with $\tilde{A} = \frac{A}{\lambda_{\max}(A)}$ and $\tilde{b} = \frac{b}{\lambda_{\max}(A)}$ instead.
   - What about $\lambda_{\min}(A) \geq \alpha$? This condition is also known as saying that A is a **well-conditioned matrix**.

   In classical computer science and in numerical algorithms, A being a well-conditioned matrix is usually needed anyway since if the minimum eigenvalue is too small, then A gets too close to being a non-invertible matrix and that causes all sorts of problems. Thus, these are so far mild conditions.

3. **Algorithm has ability to efficiently implement e^{-iAt} for a range of t**
   The expression e^{-iAt} should look familiar. This is the solution to the Schrödinger equation where we interpret A as a Hamiltonian!

$$i\frac{d|\Psi\rangle}{dt} = A|\Psi\rangle \qquad\qquad |\Psi(t)\rangle = e^{-iAt}|\Psi(0)\rangle$$

   This is pretty cool. In order to solve this exponentially large linear system, we have to essentially simulate the dynamics of a "Hamiltonian" defined by our system.

   How do we actually implement e^{-iAt}? It depends on how A is specified. We can't be given A explicitly, since it takes too long to write down A. Instead, we have to make several assumptions about A. There are a few classes of As that allow us to efficiently implement e^{-iAt}.

   One is that A is **sparse**: i.e each row of A contains at most $S \ll N$ entries, and each entry can be computed in $O(S)$ time.

This is not an unreasonable class of matrices: all local Hamiltonians (i.e a sum of local terms) are sparse matrices, so this is natural from a physics point of view.

There exist **Hamiltonian simulation algorithms** for sparse As:

Theorem 1. *e^{-iAt} can be implemented with error ϵ by a quantum circuit with size*

$$O((\log N)(\log (\frac{\tau}{\epsilon})\tau)$$

where $\tau = s^2 \cdot ||A||_{\max} \cdot t$

$||A||_{\max}$ is the largest entry in A by magnitude.

Since $t \leq \text{polylog}N$, if $S \ll \text{polylog}N$ we get efficient implementation of e^{-iAt}. We're assuming these entries of A are stored in ***Quantum Random Access Memory*** or ***QRAM***. This is a quantum data structure that allows us to query the inputs in superposition, e.g.

$$\Theta_A|i,j\rangle|\alpha\rangle = |i,j\rangle|\alpha \oplus (j^{th} \text{ entry of } i^{th} \text{ row of A})\rangle$$

in a fast manner, but also allows us to load data into the RAM in a fast manner. If the loading takes N^c time for some c, this will wipe out the exponential speed up of the HHL algorithm.

An important open question is whether there are practical and feasible QRAMS to allow us to exploit exponential speed ups of HHL or QFT, for problems where we have some external data that we want to process. Basically, the only way we know how to exploit HHL is to deal with As that are given not by massive quantities of data, but rather can be implicitly computed from a much smaller piece of data.

4. **We're given vector b in quantum form $|b\rangle$**
   Why is this a reasonable assumption? In general, it's not, but again you just need to look for situations where this could naturally arise. Again. We're going to need to assume that $|b\rangle$ is handed to us, or we have some very efficient way of generating it from scatch. If getting $|b\rangle$ or generating it takes more than polylog$N$, our exponential speed up is limited.

Another major caveat is that we get our solution $|x\rangle$ in quantum form. How useful is that? Even if we wanted to get any particular entry of $|x\rangle$, this takes $\approx N$ time to extract, because in general the amplitudes of $|x\rangle$ are $\approx \frac{1}{\sqrt{N}}$.

We will see next time some examples of problems that satisfy these caveats...

But it's still a compelling open question: find an application of HHL.

References

[1] Aram W. Harrow, Avinatan Hassidim and Seth Lloyd. Quantum algorithm for solving linear systems of equations, 2008, Phys. Rev. Lett. vol. 15, no. 103, pp. 150502 (2009); arXiv:0811.3171. DOI: 10.1103/PhysRevLett.103.150502.

[2] Zhikuan Zhao, Alejandro Pozas-Kerstjens, Patrick Rebentrost and Peter Wittek. Bayesian Deep Learning on a Quantum Computer, 2018; arXiv:1806.11463.

[3] Hartmut Neven, Vasil S. Denchev, Geordie Rose and William G. Macready. Training a Large Scale Classifier with the Quantum Adiabatic Algorithm, 2009; arXiv:0912.0779.

[4] Peter Shor. ISCA 2018 Tutorial: Grand Challenges and Research Tools for Quantum Computing, 2018;

[5] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. In Luis Alvarez, Marta Mejail, Luis Gomez, and Julio Jacobo, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 14–36, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.