CS 880: Quantum Information Processing 09/28/2010

Lecture 11: Applications of Eigenvalue Estimation

Instructor: Dieter van Melkebeek Scribe: Balasubramanian Sivan

Last class we saw eigenvalue estimation as an application of phase estimation. We also discussed an application of eigenvalue estimation to Grover's algorithm. Today, we complete our discussion of Grover's algorithm from the perspective of eigenvalue estimation, and see two other applications, namely, quantum Fourier transform over $\mathbb{Z}_M$ when $M \neq 2^m$, and solving systems of linear equations over the reals.

1 Grover's algorithm - an eigenvalue estimation perspective

Recall that in the eigenvalue estimation problem, we are given a unitary operator U and one of its eigenvectors. Our goal is to get a good approximation for the eigenvalue of U corresponding to this eigenvector. Some applications of eigenvalue estimation are:

1. An implementation of Grover's algorithm.
2. Approximate quantum Fourier transform over $\mathbb{Z}_M$ when M is not a power of 2 (We will see that the approach we used when M was a power of 2 doesn't quite work here.)
3. An efficient algorithm for "solving" sparse and well-conditioned systems of linear equations in time poly-logarithmic in the number of dimensions of the system. We mentioned this application in our first lecture. See Section 3 for why we put solving between quotes.
4. Order finding – Given m and a, find the order of a in $\mathbb{Z}_m$. We remark that order finding is a critical component in Shor's algorithm for factoring integers.

We now proceed to discuss Grover's algorithm from the eigenvalue estimation perspective. Recall that the task of Grover's algorithm is to compute an input x such that $f(x) = 1$, where $f : \{0,1\}^m \to 1$. In Grover's algorithm that we saw in lecture 6, we had to know t – the number of strings x such that $f(x) = 1$ – in order to run in time $O(\sqrt{N/t})$ and find an x that maps to 1. In yesterday's class, we mentioned a way to compute the value of t approximately using eigenvalue estimation. The circuit we used yesterday for evaluating t in Grover's algorithm, was an $n+m$ qubit circuit. We refer to the first n bits as the top part, and the remaining m qubits as the bottom part. Yesterday we used the result obtained from the first n qubits to evaluate t. Today, we will observe the other part, namely the remaining m qubits.

The circuit we used yesterday (see Figure 1) was simply the circuit for eigenvalue estimation procedure, with the operator being the Grover iterate, and the input being $|+\rangle^{\otimes n} |+\rangle^{\otimes m}$. From Exercise 2 in lecture 10, we know that the Grover iterate has two eigenvectors, namely, $|\varphi_+\rangle = \frac{1}{\sqrt{2}}|B\rangle + \frac{i}{\sqrt{2}}|C\rangle$ and $|\varphi_-\rangle = \frac{i}{\sqrt{2}}|B\rangle + \frac{1}{\sqrt{2}}|C\rangle$, with respective eigenvalues $e^{2i\theta}$ and $e^{-2i\theta}$, where θ is the angle made by the uniform superposition state $\frac{1}{\sqrt{N}}\sum_x |x\rangle$ with the B axis (i.e., the horizontal axis), and $\sin(\theta) = \sqrt{\frac{t}{N}}$. Here $|B\rangle$ and $|C\rangle$ are respectively, the superpositions of all the inputs that map to zero, and all the inputs that map to one, i.e., we have

$$|B\rangle = \frac{1}{\sqrt{M-t}} \sum_{f(x)=0} |x\rangle$$

$$|C\rangle = \frac{1}{\sqrt{t}} \sum_{f(x)=1} |x\rangle$$

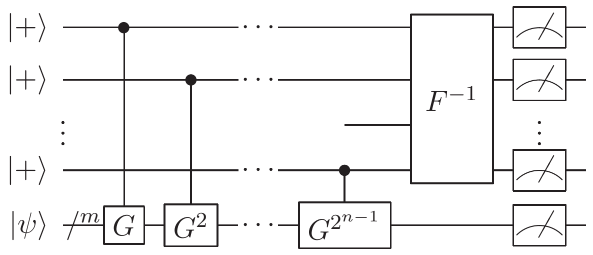

Figure 1: Grover's algorithm through the eigenvalue estimation circuit. The first n qubits of the circuit act as control wires for Grover iterates. The inverse Fourier transform is applied only on these first n qubits. The Grover iterates act on the remaining m qubits.

What is the state of the system before observation? It is the state resulting from applying the eigenvalue estimation circuit to $|+\rangle^{\otimes n} |+\rangle^{\otimes m}$, i.e.,

$$\alpha_+ \left|\widetilde{2\theta}\right\rangle |\varphi_+\rangle + \alpha_- \left|\widetilde{-2\theta}\right\rangle |\varphi_-\rangle \tag{1}$$

where $|\widetilde{x}\rangle$ is a superposition concentrated around x, which results from phase/eigenvalue estimation. What do we get if we observe the second register? First, note that if we were to observe $|\varphi_+\rangle$ alone or $|\varphi_-\rangle$ alone, then with a probability of half, we will observe an input that maps to 1. This is because both $|\varphi_+\rangle$ and $|\varphi_-\rangle$ have an amplitude square of 1/2 for $|C\rangle$. But we only get to observe the second register for the state in (1). If $\left|\widetilde{2\theta}\right\rangle$ and $\left|\widetilde{-2\theta}\right\rangle$ were orthogonal, the probability of observing an input that maps to one, is the sum of the probabilities of observing the same along with a constituent base state of $\left|\widetilde{2\theta}\right\rangle$ and observing the same along with a constituent base state of $\left|\widetilde{-2\theta}\right\rangle$, which is exactly $\alpha_+^2/2 + \alpha_-^2/2$, which is 1/2. But since $\left|\widetilde{2\theta}\right\rangle$ and $\left|\widetilde{-2\theta}\right\rangle$ are not necessarily orthogonal, to obtain the probability with which we observe an input that maps to 1, we find an upper bound on the overlap between $\left|\widetilde{2\theta}\right\rangle$ and $\left|\widetilde{-2\theta}\right\rangle$. From our analysis of the phase estimation procedure, we know that the probability that the value $\widetilde{2\theta}$ that we observe is more than δ away from 2θ is $O(\frac{1}{\delta N})$. Using this, we note that

$$\Pr[\widetilde{2\theta} \notin (0, 4\theta)] = O\left(\frac{1}{\theta N}\right)$$

$$\Pr[\widetilde{-2\theta} \notin (-4\theta, 0)] = O\left(\frac{1}{\theta N}\right)$$

If $0 < \theta < \pi/4$, then, $(0, 4\theta)$ and $(-4\theta, 0)$ don't overlap. From the above equations, the total probability mass of base states constituting $\left|\widetilde{2\theta}\right\rangle$ that lie outside $(0, \pi)$ is at most $O(\frac{1}{\theta N})$. Similarly, the total probability mass of the base states constituting $\left|\widetilde{-2\theta}\right\rangle$ that lie outside $(-\pi, 0)$ is at most $O(\frac{1}{\theta N})$. Hence $\Pr[\text{success}] \geq \frac{1}{2} - O(\frac{1}{\theta N})$. So our algorithm is the following:

1. For $n = 1, 2, \ldots$
   (a) Run eigenvalue estimation circuit on $|+\rangle^{\otimes n} |+\rangle^{\otimes m}$, with the Grover iterate operator.
   (b) If $f(\text{observed } y) = 1$, then halt.

Once N exceeds $O(\frac{1}{\theta})$, for some value of θ, we have a good probability of success, say 1/3. If we denote by i^* the trial number after which the probability of success exceeds 1/3, then the number of oracle calls till i^*, as we saw in Lecture 6, is $O(\sqrt{\frac{2^m}{t}})$. The expected number of oracle calls after i^* is given by

$$\sum_{i>i^*} 2^i \left(\frac{2}{3}\right)^{i-i^*}$$

i.e., the event that we do trial number $i > i^*$, happens only when we fail in all the previous trials. In particular, we fail in all the trials after i^*, each happens with probability at most 2/3. Hence the above expression. This series does not converge. The trick we adopted to in Lecture 6 was to change our doubling to "λ-ing" for some appropriate λ that ensures convergence of the series. But we cannot do that here because we can only decide n here. Once we increase n by 1, we are automatically doubling. So what we do is, instead of going from n to $n+1$ immediately after failing at n, we try each n twice before going to next n. Now, the trial number after which the probability of success exceeds 1/3 is $2i^*$. Till $2i^*$ trials, the number of oracle calls is $O(\sqrt{\frac{2^m}{t}})$. The expected number of Oracle calls after trial number $2i^*$ is as follows:

$$\sum_{i>2i^*} 2^{\lceil \frac{i}{2} \rceil} \left(\frac{2}{3}\right)^{i-2i^*}.$$

This is a geometric series with a ratio of $\frac{2\sqrt{2}}{3} < 1$, and thus converges. The leading term $\frac{2}{3}^{-2i^*}$ is $O(\sqrt{\frac{2^m}{t}})$. Thus, our final algorithm is as follows:

1. For $n = 1, 2, \ldots$
   Repeat twice:
   (a) Run eigenvalue estimation circuit on $|+\rangle^{\otimes n} |+\rangle^{\otimes m}$, with the Grover iterate operator.
   (b) If $f(\text{observed } y) = 1$, then halt.

Note that this algorithm is the same as our algorithm in Lecture 6, except that

- We did not have the top portion in our old circuit, namely the one corresponding to the first n qubits. The top portion of the new circuit, does two things. One is to serve as control for the Grover iterates, and the other is to do an inverse Fourier transform on the first n qubits.

- In the earlier algorithm, for any given trial, we did not pick the number of iterations of Grover iterate to be 2^n exactly. Instead, we picked a number uniformly at random from 1 to 2^n to be the number of times Grover iterate must be applied.

Now, in our new circuit, since we do not use the first n answer bits, we need not apply the inverse Fourier transform operation. Removing this inverse Fourier transform does not affect the probability of observing y in the last m qubits, for any given base state y. This is because the probability of observing y is the sum of squares of the amplitudes of the states $|x\rangle|y\rangle$, where $|x\rangle$ is an n-qubit state. This quantity is not altered by removing the inverse Fourier transform for the first n qubits. So, the only remaining difference from the algorithm in Lecture 6 is the control part. But note that using wires from a uniform superposition over all states from 0 to $2^n - 1$ as a control for Grover iterates, is just equivalent to picking the number of applications of Grover iterates uniformly at random between 0 and $2^n - 1$. So, both the algorithms are the same.

2 Quantum Fourier transform over $\mathbb{Z}_M$ when M is not a power of two

We saw how to effect the Quantum Fourier transform over $\mathbb{Z}_M$ when M is an exact power of 2, in an earlier lecture. Today we discuss the general M case. When M is not a power of 2, the first thing we do is to embed $\{0, 1, 2, \dots, M-1\}$ into an n qubit system with base states in $\{0,1\}^n$. For example, choose $n = \lceil \log M \rceil$. When M is not an exact power of two, this means that we do not use some base states in $\{0,1\}^n$, namely those states that correspond to x such that $x \geq M$. Our goal is to effect the following transformation F_M:

$$F_M : |x\rangle \longrightarrow \begin{cases} |\widehat{x}\rangle = \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{\frac{2\pi i x y}{M}} |y\rangle & \text{for } 0 \leq x < M \\ |x\rangle & \text{otherwise} \end{cases}$$

We do not care about the action of F_M for $x \geq M$.

It is worth pausing here to reflect that the idea of tensor products that we used for QFT when M is an exact power of 2, does not work here. We wrote the Fourier transform of a state $|x\rangle$ as the tensor product $\otimes_{j=1}^{n} |y_j\rangle$, where

$$|y_j\rangle = \frac{|0\rangle + e^{2\pi i (\cdot x_{n-j+1} \dots x_n)} |1\rangle}{\sqrt{2}}.$$

Note that we needed 2^n base states to use the tensor product idea.

To realize the transformation F_M, we first achieve a transformation from $|x\rangle|0\rangle|0\rangle$ to $|x\rangle|\widehat{x}\rangle|0\rangle$ as follows:

$$|x\rangle|0\rangle|0\rangle \longrightarrow |x\rangle \left(\frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} |y\rangle \right) |0\rangle \longrightarrow |x\rangle \left(\frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} |y\rangle |xy\rangle \right) \longrightarrow$$

$$|x\rangle \left(\frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{\frac{2\pi i x y}{M}} |y\rangle |xy\rangle \right) \longrightarrow |x\rangle \left(\frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{\frac{2\pi i x y}{M}} |y\rangle \right) |0\rangle = |x\rangle|\widehat{x}\rangle|0\rangle$$

Note that operation one is just producing a superposition of base states from 0 to $M-1$. Operation two is a multiplication operation, and hence is a deterministic one. Operation three is a controlled rotation. Operation four is simply undoing the reversible simulation of the multiplication operation we did.

Thus we have effected a transformation from $|x\rangle|0\rangle|0\rangle$ to $|x\rangle|\widehat{x}\rangle|0\rangle$, which is same as having effected the transformation from $|x\rangle|0\rangle$ to $|x\rangle|\widehat{x}\rangle$. But, this is not enough for us as the $|x\rangle$ in the first register will meddle with the interference pattern that must be exhibited by $|\widehat{x}\rangle$. We need our final state to be $|\widehat{x}\rangle|0\rangle$. Note that it is enough if we get the final state to be $|0\rangle|\widehat{x}\rangle$, as we can flip the two registers. We now show how to transform $|x\rangle|\widehat{x}\rangle$ to $|0\rangle|\widehat{x}\rangle$.

Actually, we show a unitary transformation from $|0\rangle|\widehat{x}\rangle$ to $|x\rangle|\widehat{x}\rangle$, and this is enough, as we know that unitary transformations are reversible. To achieve this, we first note that $|\widehat{x}\rangle$ is an eigenvector of the following operator U.

$$U : |x\rangle \longrightarrow \begin{cases} |x-1 \bmod M\rangle & \text{for } 0 \le x < M \\ |x\rangle & \text{otherwise} \end{cases}$$

To see that $|\widehat{x}\rangle$ is an eigenvector of U, note that

$$\begin{aligned} U|\widehat{x}\rangle &= \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{\frac{2\pi i x y}{M}} |y-1\rangle \\ &= \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{\frac{2\pi i x (y+1)}{M}} |y\rangle \\ &= e^{\frac{2\pi i x}{M}} |\widehat{x}\rangle . \end{aligned}$$

Thus, $|\widehat{x}\rangle$ is an eigenvector of U with eigenvalue $e^{\frac{2\pi i x}{M}}$. In the notation we used for eigenvalue estimation, $\omega = \frac{x}{M}$. So, we run eigenvalue estimation on $|\widehat{x}\rangle$ with U as the operator. This gives us $|\widetilde{x}\rangle|\widehat{x}\rangle$. Recall that the notation $|\widetilde{x}\rangle$ denotes a state that is concentrated around $|x\rangle$. If $|\widetilde{x}\rangle$ was exactly $|x\rangle$, then we have achieved the required transformation. But we will not get $|x\rangle$ exactly. For k bits of accuracy, we require our operator U applied 2^k times. Moreover, for our operator U, iterates are easy to compute because we have

$$U^r : |x\rangle \longrightarrow \begin{cases} |x-r \bmod M\rangle & \text{for } 0 \le x < M \\ |x\rangle & \text{otherwise} \end{cases}$$

So we can afford high values of k. But, is any error tolerated at all? The answer is yes because we know successive application of unitary operators never make error grow, and thus any further operations that we make on this approximate state will only carry the tiny error forward, without blowing it up.

3 "Solving" systems of linear equations

We explain why we wrote solving in quotes by describing the problem and the solution we develop. Consider a system of linear equations $Ax = b$, where A is an $N \times N$ matrix that is invertible. Our goal is to effect the following transformation: start with a normalized state $|b\rangle$, in n $(= \log N)$

qubits, and transform it into the normalized solution $|x\rangle$. As in Fourier transform, what we have at the end is a state that contains in it all the information about the solution to the system, but we do not have access to all that information: after all we can make a measurement. Nevertheless, the transformation to $|x\rangle$ might be useful in some situations. We could for example do a linear transform from $|x\rangle$ to some other state, and then make a measurement.

We also require that the matrix A is Hermitian, i.e., $A^* = A$. Note that A being Hermitian is not a restriction, as we can get around it by solving the following system instead:

$$\begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} x' \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

The coefficient matrix in the above equation is clearly Hermitian. x' is some set of variables we do not care about.

The quantum algorithm for this problem runs in time $\widetilde{O}((\log N)s^2\kappa^2\frac{1}{\epsilon})$, where the $\widetilde{O}$ notation is just the O notation with polylogarithmic terms ignored. In this particular case, polylogarithmic terms in s, κ and ϵ have been ignored. We now explain the parameters in the expression for run-time.

1. s is the sparseness of the matrix A, i.e., A contains at most s nonzero elements per row. We require that these non-zero elements can be located efficiently, i.e., in time negligible compared to the run-time mentioned above.

2. $\Omega(1) \leq ||A|| \leq 1$ is a constant less than 1 and $||A^{-1}|| \leq \kappa$. Note that for a Hermitian matrix the 2-norm just represents the largest eigenvalue. We require the largest eigenvalue of A to be less than 1. If this is not the case, we scale the matrix to achieve this. We need to have good estimates of the largest eigenvalue to do this. κ is the notation for condition number in numerical analysis. We use it here for the following reason: The condition number of a matrix is $||A|| \cdot ||A^{-1}||$, i.e., the ratio of the largest eigenvalue and the smallest eigenvalue. Since we have $||A|| \leq 1$, and $||A^{-1}|| \leq \kappa$, it follows that the condition number is at most κ. We require a system with a small condition number to have an edge over running a classical algorithm for solving the system. This is because κ could potentially be as large as N, in which case, we do not derive any benefit by running a quantum algorithm.

3. ϵ is the error parameter, i.e., we would like our final distribution to be at most ϵ away from the exact distribution.

The run-time of any naive classical algorithm is $O(\text{poly}(N)\kappa\,\text{poly-log}(\frac{1}{\epsilon}))$. Thus, the dependence on κ is better in the classical algorithm than the quantum algorithm. Also the dependence on ϵ is exponentially better in a classical algorithm. But the run-time of the classical algorithm grows polynomially in N, which is exponentially worse than the quantum algorithm's $\log(N)$ growth.

3.1 Idea behind the algorithm

We now discuss the idea behind the algorithm. The first point to observe is that the matrix A is Hermitian, and thus, it has a full basis of eigenvectors, all with real eigenvalues. Thus, any state, in particular our $|b\rangle$, can be written as

$$|b\rangle = \sum_j \beta_j |\varphi_j\rangle \tag{2}$$

where the $|\varphi_j\rangle$'s form the eigenbasis of A, with λ_j's being their eigenvalues. Thus our target state $|x\rangle$ can be written as

$$|x\rangle = \sum_j \beta_j \lambda_j^{-1} |\varphi_j\rangle . \tag{3}$$

To see this is right, note that if we operate with A on both sides, we get (2). Equation (3) suggests that the next goal is to be find these λ_j's. Given $|\varphi_j\rangle$, we need to be able to extract λ_j out. In order to do this, we need an operator U with eigenvector $|\varphi_j\rangle$ and eigenvalue $e^{i\lambda_j} = e^{2\pi i w_j}$. (For phase estimation, we require ω_j to be between 0 and 1. That's why we restricted the largest eigenvalue to be at most 1.) Now, what is the operator with the above described property? It is $U = e^{iA}$, i.e., the matrix iA exponentiated, where e^X is defined through its Taylor expansion:

$$e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}. \tag{4}$$

We know that (4) converges for every point in the complex plane, when X is a number. But in fact, it converges when X is a matrix also.

Exercise 1. *Prove the following statements.*

- *The series defined in* (4) *converges for every matrix X.*
- *If X has an eigenvector $|\varphi_j\rangle$ with eigenvalue λ_j, then, e^X too has $|\varphi_j\rangle$ an eigenvector with e^{λ_j} as the corresponding eigenvalue.*
- *If X and Y commute, i.e., $XY = YX$, then, $e^X e^Y = e^{X+Y}$.*
- *If A is Hermitian, and if $U = e^{iA}$, then $U^* = e^{-iA^*}$ and $UU^* = I$.*

4 Next time

In the next class, we will see how to use e^{iA} in effecting the transformation we need for solving the system. We will also discuss another application of eigenvalue estimation, namely order finding.