CS294 Quantum Computing

Fall 2016 **Lecture 3**

Note: Some of these notes deviate from the lecture a bit; in particular, the optimality argument for Grover's algorithm. These arguments are covered again at the beginning of next week's lecture.

Topics

- Simon's Algorithm (complementary lower bound, classical version)
- Grover's Algorithm (quantum lower bound)

Algorithmic Beginnings

Can quantum computers do what classical computers can do? Last time, we figured that if we have a classical computer that can perform:

$$x \to f(x)$$

then we can construct a quantum circuit that performs:

$$|x\rangle\,|0\rangle \to |x\rangle\,|f(x)\rangle$$

or

$$|x\rangle\,|y\rangle \to |x\rangle\,|f(x)\oplus y\rangle$$

in the general case. We can perform a change of basis with a Hadamard transformation:

$$\begin{aligned} H : |0\rangle &\to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle &\to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

The action of the gate $H^{\otimes n}$ on some example inputs is:

$$\begin{aligned} |00000000\ldots\rangle &\to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \ldots \\ &= \text{uniform superposition over all n-bit states} \\ |10000000\ldots\rangle &\to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \ldots \\ &= \text{superposition over all n-bit states, where qubits with a} \\ &\quad \text{leading 1 are negative} \\ |x_0\, x_1\ \ldots x_n\rangle &\to \sum_y (-1)^{x\cdot y}\,|y\rangle \end{aligned}$$

Hence, the operation $H^{\otimes n}$ is in some sense a Fourier transformation.
The purpose of the Hadamard gate is to take a classical 0-1 state and bring it to a state of maximal quantum superposition. Consider the sequence of operations:

$$\begin{aligned} |0\rangle^{\otimes n} &\xrightarrow{H^{\otimes n}} \sum_{y=[0..2^n-1]} |y\rangle \\ &\xrightarrow{f} \sum |y\rangle\,|f(y)\rangle \end{aligned}$$

where the coefficients were dropped for convenience. Suppose we take the output state $\sum_y |y\rangle\,|f(y)\rangle$, and we perform a measurement on the second ket ($|f(y)\rangle$) If the result of the measurement is a state $|z\rangle$, then we are told something about the first ket. In particular, the first ket is projected onto a space of preimages of z:

$$\sum_y |y\rangle\,|f(y)\rangle \rightarrow \sum_z |f^{-1}(z)\rangle\,|z\rangle$$

We will see that this measurement is useful for Simon's Algorithm.

Simon's Problem

Suppose we have a function $f_s : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$, which is 2-to-1; that is, each output has two possible inputs. We are given the following property, for some s:

$$f_s(x) = f_s(x \oplus s) \qquad s \in \mathbb{Z}_2^n$$

Given that we can only query the function f_s (but cannot look inside it – it is a black box), how fast can we determine s?

Classical Solution

The problem is reduced to finding a collision: we want to find two inputs x and y ($x \neq y$) where $f(x) = f(y)$. Since we are given (above) that the only two inputs that map to the same output are x and $x \oplus s$, we can conclude that $y = x \oplus s$ and hence $s = x \oplus y$, revealing s. So how fast can we find a collision?

The best deterministic bound is $O(2^{n-1}+1)$. We query all numbers from 0 to $2^{n-1}+1$ and look for a collision. Once we find a collision x, y s.t. $f(x) = f(y)$ we can determine $s = x \oplus y$.

If we allow a probablistic algorithm then finding collisions in this way becomes much easier! See: *birthday paradox*. The bound becomes $O(\sqrt{2^n})$. If we know that $f(x) \neq f(y)$ all we can say for sure is that $s \neq x \bigoplus y$. Thus if we have a set of n integers then we can at most eliminate the value given by the XOR of every pair. This is at most $n(n-1)/2$. Thus the number of values eliminated is quadratic in the number of queries done. So it will take at least $O(\sqrt{2^n})$ queries.

Quantum Solution (Simon's Algorithm)

Assume that we have a black box that can perform the oracle f in quantum superposition. We can do much better than $O(\sqrt{2^n})$ with the following procedure:

1. Prepare the state. Start with $|0\rangle^{\otimes n}$ and apply $H^{\otimes n}$ and f:

$$
\begin{aligned}
|0\rangle^{\otimes n} &\xrightarrow{H^{\otimes n}} \sum_y |y\rangle \\
&\xrightarrow{f} \sum_y |y\rangle |f(y)\rangle
\end{aligned}
$$

   Note the following:

$$
\sum_y |y\rangle |f(y)\rangle = \sum_z (|z\rangle + |z \oplus s\rangle) |f(z)\rangle
$$

   The reason the above is true is because for every output $f(z)$ there are two corresponding inputs z and $z \oplus s$ that map to it.

2. Apply $H^{\otimes n}$ to the state.

$$
\begin{aligned}
&H^{\otimes n}\left[\sum_z (|z\rangle + |z \oplus s\rangle) |f(z)\rangle\right] \\
=&\sum_y \sum_z \left((-1)^{y\cdot z} + (-1)^{y\cdot(z\oplus s)}\right) |y\rangle |f(z)\rangle
\end{aligned}
$$

   If $y \cdot z = 0$:

$$
(-1)^0 + (-1)^{y\cdot s}
$$

   If $y \cdot z = 1$:

$$
\begin{aligned}
&(-1)^1 + (-1)^{1+y\cdot s} \\
=&-(1+(-1)^{y\cdot s})
\end{aligned}
$$

   In both cases, if $y \cdot s = 1$, the amplitude vanishes! Hence, if after we perform the Hadamard transformation, we measure the first ket to be $|y\rangle$, then we can guarantee that the resulting state has the property $y \cdot s = 0$ since all other amplitudes vanish.

3. If we repeat this measurement multiple times we get outputs:

$$
y_1, y_2, y_3, \ldots \qquad y_i \cdot s = 0 \quad \forall i
$$

   Now if at some iteration we have $n-1$ linearly independent vectors, then they will span the entire vector space perpendicular to s, and we can fully determine s. Hence, all we need to complete the analysis is the expected time for this process to give $n-1$ independent vectors. For that analysis, let S_i denote the Span$(y_1, y_2, \ldots y_i)$ and D_i denote Dimension(S_i).

   We note that $P(D_{i+1} = k+1 | D_i = k) = (2^n - |S_i|)/2^n$ since each vector has a $1/2^n$ probability of being picked. Also, $P(D_{i+1} = k | D_i = k) = |S_i|/2^n$ and since one vector can increase the dimension of a space by at most one, there is no other value that D_{i+1} can take.

Note that since $\mathbb{Z}_2$ is a field of cardinality 2, and S_i is a vector space over it, the number of elements in S_i given that $D_i = k$ is simply 2^k. Thus the process from the state $D_i = k$ can also be viewed in the following way: Toss a coin with probability of failure as $2^k/2^n$. On failure D_{i+1} remains k, on success it gets updated to $k+1$. Thus the expected waiting time at state k is $2^n/(2^n - 2^k)$. Hence the total expected time to hit $n-1$ is:

$$\sum_{i=0}^{n-1} \frac{2^n}{2^n - 2^i} < \sum_{i=0}^{n-1} 2 = 2n$$

So we can see that the quantum algorithm operates in $O(n)$ queries while the classical algorithm operates in $O(2^n)$ queries. Note that this is only true given the "black box" assumption at the beginning – that we can apply the oracle in superposition easily.

Grover's Search Problem

We have a function $f : \{0,1,\ldots,n-1\} \to \{0,1\}$. Find x s.t. $f(x) = 1$. We consider the hardest case, where $f(x) = 1$ for only one input. That is:

$$\begin{cases} f(x) = 1 & \text{if } x = a \\ f(x) = 0 & \text{else} \end{cases}$$

Classically we can do it in $O(n)$ time – we just query every n! And we can't do better, since we know nothing else about f. Can we do better quantumly?

Quantum Lower Bound

To analyse this problem firstly let us initiate a state space. But this time instead of having the oracle registry initially as $|0\rangle$, we will have it to be $|0\rangle - |1\rangle$. Then as the oracle performs a XOR of the original registry value and $f(i)$ we have:

$$\begin{aligned} \sum_i \alpha_i |i\rangle (|0\rangle - |1\rangle) \xrightarrow{f} & \sum_i \alpha_i |i\rangle (|0 \oplus f(i)\rangle - |1 \oplus f(i)\rangle) \\ &= \sum_{i:f(i)=0} \alpha_i |i\rangle (|0\rangle - |1\rangle) + \sum_{i:f(i)=1} -\alpha_i |i\rangle (|0\rangle - |1\rangle) \end{aligned}$$

Thus the oracle essentially reverses the amplitude of the basis vector containing $|i\rangle$ where $f(i) = 1$. This was the reason for initialising the system to have registry values as $|0\rangle - |1\rangle$. For clarity, we will simply denote the state $|i\rangle (|0\rangle - |1\rangle)$ as $|i\rangle$, and define the oracle operator F, where:

$$F|i\rangle = \begin{cases} -|i\rangle & \text{if } \; f(i) = 1 \\ |i\rangle & \text{if } \; f(i) = 0 \end{cases}$$

Now, the essential idea is that if we could somehow amplify the negative vectors enough then we could measure the required state with high probability. Grover's algorithm is thus less of a search algorithm and more of an amplification algorithm.

For simplicity, we consider the reduced problem where there is only one satisfying predicate. That is, $f(i) = 1$ iff $i = a$, and $f(i) = 0$ elsewhere.

We can write the operator F as:

$$F = I - 2\left|a\right\rangle\left\langle a\right|$$

(You should confirm that this agrees with our previous definition above). We also define an initial state $\left|s\right\rangle$:

$$\left|s\right\rangle = \frac{1}{\sqrt{N}}\sum_i \left|i\right\rangle$$

representing maximal ignorance of the correct element. Note that the action of F on $\left|s\right\rangle$ is:

$$\begin{aligned} F\left|s\right\rangle &= (I - 2\left|a\right\rangle\left\langle a\right|)\left|s\right\rangle \\ &= \left|s\right\rangle - \frac{2}{\sqrt{N}}\left|a\right\rangle \\ &= \frac{1}{\sqrt{N}}\left[\left(\sum_{i\neq a}\left|i\right\rangle\right) - \left|a\right\rangle\right] \end{aligned}$$

That is, **applying the oracle to the initial state negates the amplitude of the satisfying element**.

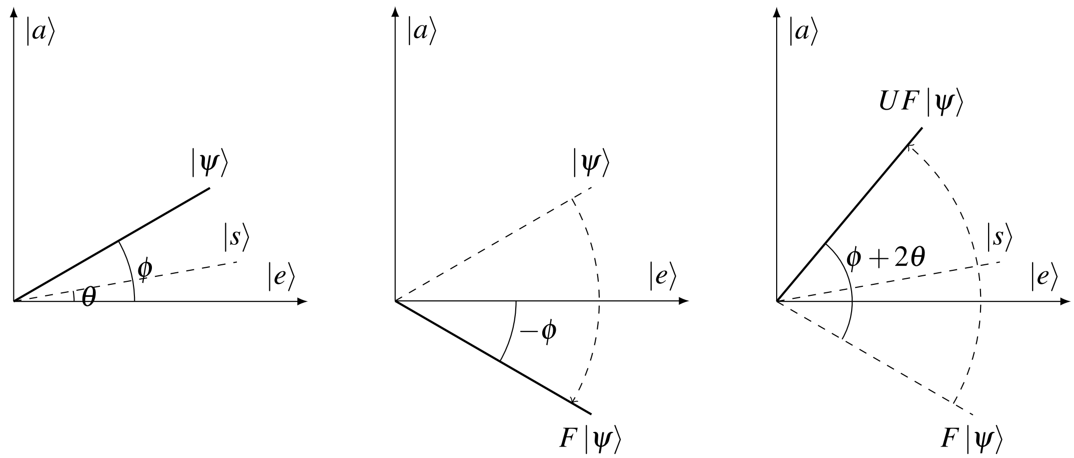
We now introduce the Grover diffusion operator:

$$U = 2\left|s\right\rangle\left\langle s\right| - I$$

One iteration of the algorithm consists of applying the operator $A = UF$ (that is, querying the oracle and then applying the diffusion operator). After one step of amplification:

$$\begin{aligned} A\left|s\right\rangle &= UF\left|s\right\rangle \\ &= (2\left|s\right\rangle\left\langle s\right| - I)(I - 2\left|a\right\rangle\left\langle a\right|)\left|s\right\rangle \\ &= \left(2\left|s\right\rangle\left\langle s\right| + 2\left|a\right\rangle\left\langle a\right| - \frac{4\left|s\right\rangle\left\langle a\right|}{\sqrt{N}} - I\right)\left|s\right\rangle \\ &= \left(1 - \frac{4}{N}\right)\left|s\right\rangle + \frac{2}{\sqrt{N}}\left|a\right\rangle \end{aligned}$$

We see that after one iteration, the probability of measuring a has increased. With some calculation, it can be checked that the transformation A rotates the state vector by $2\sqrt{N-1}/N \approx 2/\sqrt{N}$. Since we start out almost orthogonal to $\left|a\right\rangle$ (assuming N is large), we need to rotate an angle of $\pi/2$ in total, and thus we need to apply A about $\pi\sqrt{N}/4$ times to get a vector close enough to $\left|a\right\rangle$. Hence the algorithm runs in $O(\sqrt{N})$ queries of the oracle.



(a) Start of iteration (b) After F (c) After U

Figure 1: One iteration of Grover's algorithm

Geometric Visualization of Grover's Algorithm

We can visualize the operators in Grover's algorithm as reflections in state space. Consider the target vector $|a\rangle$ and the hyperplane of all other vectors $|e\rangle \equiv \frac{1}{\sqrt{N-1}} \sum_{i \neq a} |i\rangle$ (see figure).

Note that $F = I - 2\,|a\rangle\,\langle a|$ corresponds to a flip over $|e\rangle$ and $U = 2\,|s\rangle\,\langle s| - I$ corresponds to a flip over $|s\rangle$. If we define θ as the angle between $|s\rangle$ and $|e\rangle$, and ϕ as the angle between $|\psi\rangle$ and $|e\rangle$ (where $|\psi\rangle$ is the state at the current iteration), we see that the transformations perform the following rotations:

$$\phi \xrightarrow{F} (-\phi) \xrightarrow{U} \phi + 2\theta$$

Hence, after one iteration we rotate our state forward by $2\theta = 2\arcsin\frac{1}{\sqrt{N}} \approx \frac{2}{\sqrt{N}}$. Given that we start with $\phi = \theta \approx \frac{1}{\sqrt{N}}$ and want to rotate by $\pi/2$ in total, we will need $\frac{\pi/2-\theta}{2\theta} \approx O(\sqrt{N})$ iterations (for large N).

Optimality of Grover's Algorithm

Note: We will sometimes abbreviate state vectors with their labels, e.g. $v \iff |v\rangle$.

To prove optimality, we show that any sequence of unitary operators (combined with calls to the oracle) that distinguish between the function that has 0 everywhere and the function which is 1 at the a'th position requires at least $O(\sqrt{N})$ calls of the oracle. So let $U_1, U_2, \ldots$ be some unitary operators and $|v_{a,k}\rangle = U_k F_a U_{k-1} F_a \ldots U_1\,|s\rangle$, where F_a is the oracle corresponding to the function $f(x)$ which is 1 iff $x = a$. Also, let $|s_k\rangle = U_k U_{k-1} \ldots U_1\,|s\rangle$. Essentially, s_k is the state we would have been in if f had no 1's. Note that since the oracle depends on a, F changes with a and thus so does a_k. However since no measurement is done before the end, we can glean no information about a a priori, and thus the $U_i's$ are independent of a, implying that s_k is independent of a.
Now define $t_{a,k} = |v_{a,k} - s_k|$. Essentially, $t_{a,k}$ is a measurement of the error between a run of the algorithm with a function which is always 0 or a function which is 1 at position a.

Then we are first going to prove that $t_{a,k} \leq \sum_{i=1}^{k-1} 2(a \cdot s_i)$. We will prove this by induction. Note that at $k = 1$ $v_{a,1} = U_1\,|s\rangle = s_1$. Thus $t_{a,1} = 0$ and the inequality is trivially true.

Now let us assume by induction that for some k, $t_{a,k} \leq \sum_{i=1}^{k-1} 2(a \cdot s_i)$. Then for $k+1$:

$$\begin{aligned}
t_{a,k+1} &= |v_{a,k+1} - s_{k+1}| \\
&= |U_{k+1} F_a v_{a,k} - U_{k+1} s_k| \\
&= |F_a v_{a,k} - s_k| \quad [U_{k+1} \text{ is unitary}] \\
&= |F_a v_{a,k} - F_a s_k + F_a s_k - s_k| \\
&\leq |F_a(v_{a,k} - s_k)| + |(F_a - I)s_k| \quad [\text{by triangle inequality}] \\
&= |v_{a,k} - s_k| + 2(a.s_k) \\
&= t_k + 2(a \cdot s_k) \\
&\leq \sum_{i=1}^{k} 2(a \cdot s_i) \quad [\text{by Induction on k}]
\end{aligned}$$

Thus by induction we have that:

$$t_{a,k} \leq \sum_{i=1}^{k-1} 2(a \cdot s_i) \tag{1}$$

Now since our algorithm needs to distinguish between the function which is 0 everywhere and the one which has 1 at a, $t_{a,T}$ must be large, since this is the difference between the output vectors when these two functions are used as inputs. So for the sake or argument say $t_{a,T} > 1/2$.

Now applying Cauchy Schwartz on the R.H.S. of (1) we get:

$$\begin{aligned}
1/2 &< t_{a,T} \\
&\leq \sum_{i=1}^{T-1} 2(a \cdot s_i) \\
&\leq \sqrt{T-1}\sqrt{\sum_{i=1}^{T-1} (a \cdot s_i)^2}
\end{aligned} \tag{2}$$

But as the different vectors a are an orthonormal basis for the space and as each s_i is unit norm we have $\sum_a (a \cdot s_i)^2 = 1 \Rightarrow \sum_a \sum_{i=1}^{T-1} (a \cdot s_i)^2 = T-1$. Since there are N such vectors a, there is at least one choice of a for which $\sum_{i=1}^{T-1} (a \cdot s_i)^2 < (T-1)/N$.

Plugging this value of a into (2), we get that there is a choice of a, such that $1/2 < \sqrt{T-1}\sqrt{(T-1)/N} = (T-1)/\sqrt{N}$.

Thus we get that $T = O(\sqrt{N})$ as desired.