CMSC 858L: Quantum Complexity

Instructor: Daniel Gottesman

Spring 2023

11 Quantum lower bounds

11.1 References

The hybrid argument showing a lower bound on Grover's algorithm and the separation of BQP from NP is from Bennett, Bernstein, Brassard, and Vazirani, "Strengths and Weaknesses of Quantum Computing," quant-ph/9701001 (often known as "BBBV"), although I have modified the argument slightly. (The BBBV paper also proved $\mathrm{BQP}^{\mathrm{BQP}} = \mathrm{BQP}$.) Lower bounds using polynomials were developed by Beals, Buhrman, Cleve, Mosca, and de Wolf, "Quantum Lower Bounds by Polynomials," quant-ph/9802049.

We now embark on studying methods of lower bounding the quantum query complexity. We'll apply each method to unstructured search, but we will look at some other applications as well.

11.2 Hybrid argument

But so far we have just an upper bound on the quantum query complexity. Can we set a lower bound on the quantum query complexity? Yes, in fact, we can, but it requires some work. The first argument I will show you is known as a *hybrid* argument, but then we will cover some other techniques for proving quantum lower bounds and we'll try them out on this same problem.

Theorem 1. *The quantum query complexity for unstructured search is $Q = \Omega(\sqrt{N})$, and thus in fact $Q = \Theta(\sqrt{N})$.*

Proof. Suppose we have an algorithm that makes T queries. That algorithm can be written as an alternating sequence of unitaries and oracle calls $U_0, O, U_1, O, \ldots, U_{T-1}, O, U_T$. The unitaries don't have to all be the same and they can be many-qubit very complex unitaries — we only care about the query complexity, not the circuit complexity. Importantly, the sequence $U_0, U_1, \ldots, U_T$ of unitaries is the same regardless of the oracle used, since the algorithm knows nothing about the oracle beyond what it learns in oracle calls, and its reaction to those oracle calls is subsumed into the unitaries U_i.

Let

$$V_0 = U_T \prod_{i=T-1}^{0} O_0 U_i \tag{1}$$

and

$$V_{x_0} = U_T \prod_{i=T-1}^{0} O_{x_0} U_i. \tag{2}$$

We are using the same notation for oracles as before: O_0 is the oracle with no marked elements and O_{x_0} is the oracle with only x_0 marked. We wish to show that $V_0|\psi\rangle$ is close to $V_{x_0}|\psi\rangle$ for any $|\psi\rangle$ unless $T = \Omega(\sqrt{N})$.

To do this, we will proceed through a sequence of hybrid circuits (thus the name of the method)

$$V_{x_0}^{(j)} = U_T \prod_{i=T-1}^{j} O_{x_0} U_i \prod_{i=j-1}^{0} O_0 U_i. \tag{3}$$

Thus $V_{x_0}^{(0)} = V_{x_0}$ and $V_{x_0}^{(T)} = V_0$. Each successive $V_{x_0}^{(j)}$ has one less call to O_{x_0} and one more call to O_0 instead. We will actually show that $V_{x_0}^{(j)}$ and $V_{x_0}^{(j+1)}$ are very close for almost all x_0, and then the total distance between V_0 and V_{x_0} can be at most T times the maximum distance between adjacent $V_{x_0}^{(j)}$'s.

Since the only difference between $V_{x_0}^{(j)}$ and $V_{x_0}^{(j+1)}$ is a single oracle call, we can simplify the comparison into one between $V_{x_0}^f U_0 V^i$ and $V_{x_0}^f U_{x_0} V^i$, where V^i is the product of unitaries (including oracles) performed before the $(j+1)$th oracle call and $V_{x_0}^f$ is the product of unitaries (including oracles) performed after the $(j+1)$th oracle call. Note in particular that while $V_{x_0}^f$ can depend on x_0, V^i does not. Let $|\psi\rangle$ be the initial state of the algorithm and let

$$|\phi\rangle = V^i|\psi\rangle = \sum_x \alpha_x |x\rangle \otimes |\phi_x\rangle. \tag{4}$$

Here the first register is the query that will be given to the oracle at step $j+1$ and we can assume that the states $|\phi_x\rangle$ are normalized.

We can simplify this by noting that by permuting the inputs to the oracle, we can exchange any O_{x_0} with any other $O_{x_0'}$. In particular, the algorithm has no reason to prefer any x_0's over any others. Indeed, any algorithm can be symmetrized by choosing random permutation of inputs and applying it everywhere without altering the average success probability, averaged over x_0. We can purify this procedure (so that we can use unitaries everywhere), and the conclusion is that since $|\phi\rangle$ doesn't depend at all on the actual x_0 in the oracle, the amplitudes $|\alpha_x|$ must have the same magnitude $1/\sqrt{N}$. Then

$$O_0|\phi\rangle = \sum_x \alpha_x |x\rangle \otimes |\phi_x\rangle \tag{5}$$

$$O_{x_0}|\phi\rangle = \sum_{x \neq x_0} \alpha_x |x\rangle \otimes |\phi_x\rangle - \alpha_{x_0}|x_0\rangle \otimes |\phi_{x_0}\rangle. \tag{6}$$

(Assume we use the oracle in the phase form, but the standard form doesn't materially alter this argument.) Thus,

$$\langle\phi|O_0^\dagger O_{x_0}|\phi\rangle = \sum_{x \neq x_0} |\alpha_x|^2 - |\alpha_{x_0}|^2 = (N-2)/N. \tag{7}$$

This is indeed the lowest fidelity we can achieve between these two states for any ancilla state for the oracle output. Because the operation $V_{x_0}^f$ is unitary, even though it depends on the marked value x_0, the fidelity between the two possible final states for $V_{x_0}^{(j)}$ and $V_{x_0}^{(j+1)}$ has to remain the same as the fidelity between $O_0|\phi\rangle$ and $O_{x_0}|\phi\rangle$, that is to say at least $1 - 2/N$.

The next step is to add up all these distances to get the distance between the $V_{x_0}^{(j)}$'s in order to find the distance between V_0 and V_{x_0}. However, infidelity is not a metric and doesn't need to satisfy the triangle inequality, so we will switch to trace distance. The trace distance is at most

$$\sqrt{1-F^2} \leq \sqrt{1-(1-2/N)^2} = \sqrt{4/N - 4/N^2} \leq 2/\sqrt{N}. \tag{8}$$

The trace distance does satisfy the triangle inequality, so

$$D(V_0|\psi\rangle, V_{x_0}|\psi\rangle) \leq \sum_{j=0}^{T-1} D(V_{x_0}^{(j+1)}|\psi\rangle, V_{x_0}^{(j)}|\psi\rangle) \leq 2T/\sqrt{N}. \tag{9}$$

(Trace distance is actually between density matrices, so consider the above pure states as shorthand for the corresponding density matrices.) Thus, if $T = o(\sqrt{N})$, the outcome distributions of the two states must be very close for large N.

Because of the symmetrization, we have made all the trace distances $D(V_0|\psi\rangle, V_{x_0}|\psi\rangle)$ the same, but before symmetrization, some could be higher than that while others would be lower. However, by the Markov inequality, the number of x_0 such that $D(V_0|\psi\rangle, V_{x_0}|\psi\rangle) > 2kT/\sqrt{N}$ is at most N/k. In particular, when $T = o(\sqrt{N})$, the fraction of values of x_0 for which the trace distance is not small is itself vanishing in the limit of large N.

Let $P_b(O)$ be the probability that this algorithm outputs b to the decision problem when given oracle O. Since for almost all x_0, $D(V_0|\psi\rangle, V_{x_0}|\psi\rangle)$ is small when $T = o(\sqrt{N})$, $P_b(O_0)$ must be close $P_b(O_{x_0})$ and therefore, the algorithm fails to be correct with probability 2/3. In order to achieve this level of accuracy, we must have $T = \Omega(\sqrt{N})$.

□

11.3 Separating BQP from NP relative to an oracle

We then get

Theorem 2. *Relative to a random oracle O, $NP^O \not\subseteq BQP^O$ with probability 1.*

This gives some evidence, for what it's worth, that BQP can't solve NP-complete problems.

Proof. Given O, we can define another oracle $\tilde{O} : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ (for all n) as follows: the ith bit of $\tilde{O}(x)$ is $O(x, i)$, where the input (x, i) is the concatenation of x and i, where i is written using exactly $\lceil \log_2 n \rceil$ bits. $\tilde{O}$ can be computed using $O(\log n)$ calls to O and a particular value of O for a given input of size $n + \lceil \log_2 n \rceil$ can be computed using one call to $\tilde{O}$. Thus, querying O and querying $\tilde{O}$ are essentially equivalent up to possible logarithmic factors. Furthermore, every function $\tilde{O}$ can be defined in this way from an O, and all are equally likely for random O, so $\tilde{O}$ is a random functional oracle.

Define the language L as follows: $y \in L$ iff $\exists\, x$ such that $\tilde{O}(x) = y$. This language can be decided in NP^O using just one oracle call to $\tilde{O}$ and therefore just logarithmically many to O.

However, it cannot be decided in BQP^O: Any algorithm to decide it using T queries to O can be coverted to an algorithm using at most T queries to $\tilde{O}$. Queries to $\tilde{O}$ are not going to be any more effective for this problem that queries to a decision version which returns 0 if $\tilde{O} \neq y$ and returns 1 if $\tilde{O} = y$. Since $\tilde{O}$ is random, for most values of y, there is at most one x_0 such that $\tilde{O}(x_0) = y$, and we know from the previous proof that for any quantum query algorithm using $o(\sqrt{N})$ queries, for almost all x_0, the algorithm does not succeed with probability at least 2/3. That is, for most values of $\tilde{O}$ for this particular value of n, the algorithm fails. By letting $n \to \infty$, we get that $L \notin \mathrm{BQP}^O$ with probability 1.

□

11.4 Bounds from polynomials

Another approach to setting lower bounds to query complexity is to approximate the functions we need to compute by polynomials. In fact, if we think of the oracle as an N-bit string with ith bit X_i, we can think of the X_i's as variables that take on the value 0 or 1 as we vary the oracle. Then the function $f(O)$ we are supposed to compute in the query complexity problem is a polynomial in the X_i's of degree at most N: Any function on a finite number of points (in this case 2^N points) can be written as a polynomial. A priori, this polynomial would have very high degree, as high as $2^N - 1$. However, f is only evaluated for $X_i = 0$ or $X_i = 1$, so any factor $X_i^a = X_i$ for $a > 1$. Thus, the highest degree term possible is $\prod_i X_i$, which is degree N.

The function f itself is only defined at points where $X_i = 0, 1$, but the polynomial can be sensibly interpolated to arbitrary real values for X_i between 0 and 1. There might be a much lower degree polynomial than the obvious one that exactly matches f or gets close to it at all points where all $X_i = 0, 1$ but behaves very differently elsewhere.

Definition 1. *The* degree $\deg f$ *of a function f on N variables is the smallest value d such that there exists polynomial $p(X_0, \ldots, X_{N-1})$ of degree d (with real coefficients) with $f(x_0, \ldots, x_{N-1}) = p(x_0, \ldots, x_{N-1})$ for all $(x_0, \ldots, x_{N-1}) \in \{0, 1\}^N$. The* approximate degree $\widetilde{\deg} f$ *of f is the smallest degree d such that there exists polynomial p of degree d (with real coefficients) with*

$$|f(x_0, \ldots, x_{N-1}) - p(x_0, \ldots, x_{N-1})| \leq 1/3 \tag{10}$$

for all $(x_0, \ldots, x_{N-1}) \in \{0, 1\}^N$.

It turns out that the approximate degree will give us a lower bound on the quantum query complexity:

Theorem 3. $Q(f) \geq (\widetilde{\deg} f)/2$

Proof. The main observation is that if we have a quantum query algorithm with T queries, the amplitudes are given by polynomials of degree at most T. To prove this claim, consider again the algorithm as an alternating sequence of unitaries and oracle calls. Before the first oracle call, the state is $\sum_{i,j} \alpha_{i,j}|i\rangle|j\rangle$, with the first register being the one used to query the oracle. After the first oracle call, the state is now

$$\sum_{i,j}(-1)^{O}(i)\alpha_{i,j}|i\rangle|j\rangle = \sum_{i,j}(1-2X_i)\alpha_{i,j}|i\rangle|j\rangle. \tag{11}$$

Here we are substituting $O(i) = X_i$ and using the fact that the oracle takes values only at 0 and 1. At this point, therefore, the amplitudes are linear functions of the variables X_i as the oracle changes.

By the same argument, if just before the jth oracle call, the amplitudes are degree $j-1$ polynomials of the X_i variables, then after the jth oracle call, the amplitudes are degree j polynomials. We therefore can prove the claim that the final amplitudes are degree T polynomials by induction, provided that the unitaries between the oracle calls don't mess up this property. That is easy to check: If before U, the state is $\sum_a q_a(X_0,\ldots,X_{N-1})|a\rangle$, with q_a polynomials of degree j, then after U, the state is

$$\sum_b \left[\sum_a q_a(X_0,\ldots,X_{N-1})U_{ba}\right]|b\rangle, \tag{12}$$

and $\sum_a q_a(X_0,\ldots,X_{N-1})U_{ba}$ is also a polynomial of degree j.

So now we know the final state has the form $\sum_a q_a(X_0,\ldots,X_{N-1})|a\rangle$, where the q_a's are polynomials of degree at most T. Note, however, that q_a could have complex coefficients even though for $\widetilde{d}(f)$ we care about polynomials over the real numbers; but this will get taken care of soon.

At the final stage, we make a measurement of the first qubit to determine the outcome of the algorithm. The probability that the outcome is 1 is

$$\text{Prob}(1) = \sum_{a=1a'} |q_{1a'}(X_0,\ldots,X_{N-1})|^2. \tag{13}$$

When the q_a's are complex polynomials of degree at most T, Prob(1) is therefore a real polynomial $p(X_0,\ldots,X_{N-1})$ of degree at most $2T$.

Now, since the algorithm decides f correctly, that means that $\text{Prob}(1) \geq 2/3$ if $f(O) = 1$ and $\text{Prob}(1) \leq 1/3$ if $f(O) = 0$. In particular, in either case,

$$|f(O) - p(X_0,\ldots,X_{N-1})| \leq 1/3. \tag{14}$$

The algorithm must work for all oracles, so this equation holds for all $(X_0,\ldots,X_{N-1}) \in \{0,1\}^N$.

Since $\widetilde{\deg}(f)$ is the smallest degree of a polynomial satisfying this condition, and p is a polynomial of degree $2T$ satisfying the condition, it follows that $\widetilde{\deg}(f) \leq 2T$. Thus, $Q(f) \geq \widetilde{\deg}(f)/2$. □