Week 4

The Negative-weight Adversary

4.1 Limitations of the positive adversary method

The adversary method we saw last week is arguably the most commonly used lower bound technique for quantum query complexity. However, despite being relatively easy to use, there are specific scenarios in which it is known to fail. Two such barriers are the property testing barrier, and the certificate barrier.

4.1.1 Property testing barrier

Suppose a function f is a promise problem with the property that all 0-inputs are far from all 1-inputs. More specifically, suppose that f acts on n bits, and every pair $x, y \in \mathrm{Dom}(f)$ with $f(x) \neq f(y)$ satisfies $|x-y| \geq d$, where $|x-y|$ denotes the Hamming distance between x and y (the number of bits where they differ).

Using this property of f, we can immediately come up with a feasible solution to the primal version of the positive adversary bound. Recall that this only requires coming up with a set of weights $w_{x,i} \geq 0$ such that $\sum_{i:x_i \neq y_i} \sqrt{w_{x,i} w_{y,i}} \geq 1$ for all x and y with $f(x) \neq f(y)$. Since for our function f the distance between x and y with $f(x) \neq f(y)$ is always at least d, we can set $w_{x,i} = 1/d$ for all x and i, and this will clearly be feasible. The objective value of this feasible solution is the maximum over $x \in \mathrm{Dom}(f)$ of $\sum_{i \in [n]} w_{x,i}$, which is n/d. Hence we immediately conclude that $\mathrm{Adv}(f) \leq n/d$.

Theorem 4.1. *Let f be a partial function on n bits such that for $x, y, \in \mathrm{Dom}(f)$, we have $f(x) \neq f(y) \Rightarrow |x-y| \geq d$. Then $\mathrm{Adv}(f) \leq n/d$.*

In particular, if $d = \Omega(n)$, the property testing barrier says that $\mathrm{Adv}(f) = O(1)$. That is, whenever all the 0-inputs are very far from all the 1-inputs (so that they differ on a constant fraction of the indices), the adversary method cannot prove any lower bound better than constant.

However, there exist functions with large gaps between 0- and 1-inputs that do not have fast quantum algorithms. In fact, any "property testing" problem is of this form. A property testing problem is a special type of query problem in which we care about testing whether the input is in some set $S \subseteq \{0,1\}^n$. Generally speaking, the setting we care about in property testing is that the input x will either be from S, or else will be very far from being in S (it will differ on a constant fraction of the indices from every string in S). The goal is to distinguish between these two extreme cases (being in S or being very far from everything in S). What we have just seen is that the positive adversary method can never give super-constant lower bounds on the query complexity of property testing problems.

Other problems of interest also have their 0- and 1-inputs be very far from each other. A famous example is the collision problem. In this problem, the input string is a subset of $[m]^n$, where $m \geq n$ is the size of the alphabet. The promise is that the input string will either have n distinct alphabet symbols, or else will have $n/2$ distinct alphabet symbols occurring twice each (we assume n is a multiple of 2). The task is to distinguish between these (for example, output 0 in the first case and 1 in the second case). It is not hard to see that every 0-input and every 1-input have distance at least $n/2$ (they disagree on at least $n/2$ indices). This means the positive adversary method once again cannot prove a lower bound better than constant.

4.1.2 The certificate barrier

While the property testing barrier says that the adversary method won't work well on certain types of partial functions, the certificate barrier says that it also does not work well on certain types of total functions. In particular, the adversary bound is always at most the certificate complexity of f.

To see this, we again construct a feasible solution to the primal formulation of the positive adversary bound. For each input x of a total Boolean function f, pick a certificate c_x consistent with x of size at most $\mathrm{C}(f,x)$. Then set $w_{x,i} = 1$ if i is a non-$*$ index of the certificate c_x, and set $w_{x,i} = 0$ otherwise. Now, recall that each 0-certificate and each 1-certificate of a total Boolean function must contradict each other; that is, there must be some index i on which their disagree (and both are not $*$ on this bit). Therefore, if x is a 0-input and y is a 1-input of f, then their certificates c_x and c_y overlap (and disagree) on some bit i, which means $x_i \neq y_i$ and $w_{x,i} = wy, i = 1$. It follows that $\sum_{i:x_i \neq y_i} \sqrt{w_{x,i} w_{y,i}} \geq 1$, which means this weight scheme is feasible. Moreover, the objective value of it is the maximum over x of $\sum_{i \in [n]} w_{x,i} = |c_x| = \mathrm{C}(f,x)$, which is $\mathrm{C}(f)$.

We can actually improve this upper bound, which will show that the adversary bound is even weaker. First, consider dividing all the weights $w_{x,i}$ for $x \in f^{-1}(0)$ by some positive constant A and multiplying the weights $w_{y,i}$ for $y \in f^{-1}(1)$ by A. This does not change any of the products $w_{x,i} w_{y,i}$, and hence does not change the feasibility of the solution. However, the objective value becomes the maximum of $\mathrm{C}_0(f)/A$ and $\mathrm{C}_1(f) \cdot A$, where recall that $\mathrm{C}_0(f)$ is the maximum of $\mathrm{C}(f,x)$ over 0-inputs x and $\mathrm{C}_1(f)$ is the maximum over 1-inputs. By picking $A = \sqrt{\mathrm{C}_0(f)/\mathrm{C}_1(f)}$, we get that $\mathrm{Adv}(f) \leq \sqrt{\mathrm{C}_0(f)\,\mathrm{C}_1(f)}$.

We can strengthen this further by replacing certificate complexity with fractional certificate complexity, which is smaller. To do so, pick an optional solution $w(x,i)$ for fractional certificate complexity, so that $\sum_i w(x,i) = \mathrm{FC}(f,x)$ for all x and $\sum_{i:y_i \neq x_i} w(x,i) \geq 1$ for all x, y such that $f(x) \neq f(y)$. We claim that $w(x,i)$ is also feasible for the adversary bound. To show this, we need to lower bound $\sum_{i:x_i \neq y_i} \sqrt{w(x,i) w(y,i)}$ when $f(x) \neq f(y)$. To analyze this, we consider a new string z defined as follows: for i such that $x_i = y_i$, set $z_i = x_i$; for i such that $x_i \neq y_i$, set $z_i = x_i$ if $w(x,i) > w(y,i)$, and set $z_i = y_i$ otherwise. Now, since f is a total function, the string z is a valid input to f, and since $f(x) \neq f(y)$, we have either $f(z) \neq f(x)$ or $f(z) \neq f(y)$.

Without loss of generality, assume $f(z) \neq f(x)$. Then by the feasibility of the fractional certificate of x, we have $\sum_{i:z_i \neq x_i} w(x,i) \geq 1$. Note that for i in this sum, we have $z_i = y_i$, so $w(x,i) \leq w(y,i)$ by the definition of z. Hence $\sqrt{w(x,i)w(y,i)} \geq w(x,i)$ for such i. This means that $\sum_{i:z_i \neq x_i} \sqrt{w(x,i)w(y,i)} \geq 1$. Expanding the sum so that it sums over all i such that $y_i \neq x_i$ only makes it larger. This shows that $\sum_{i:x_i \neq y_i} \sqrt{w(x,i)w(y,i)} \geq 1$, so that $w(x,i)$ is a feasible weight scheme for the adversary bound. Its objective value is $\mathrm{FC}(f) = \mathrm{fbs}(f)$, so $\mathrm{Adv}(f) \leq \mathrm{fbs}(f)$. Moreover, by doing the rebalancing trick (scaling the weights of 0-inputs by A and of 1-inputs by $1/A$), we can also get $\mathrm{Adv}(f) \leq \sqrt{\mathrm{fbs}_0(f)\,\mathrm{fbs}_1(f)}$. This holds for all total Boolean functions f.

Theorem 4.2. *Let f be a total Boolean function. Then* $\mathrm{Adv}(f) \leq \sqrt{\mathrm{fbs}_0(f)\,\mathrm{fbs}_1(f)}$.

We can also get a certificate barrier for partial functions, although it is not as strong of a barrier. To do so, we will set $w(x,i)$ to be $w'(x,i)^2$ when x is a 0-input, where $w'(x,i)$ is the weights of the optimal fractional certificate for x; however, we will set $w(y,i) = 1$ for all i when y is a 1-input. Then $\sum_{i:x_i \neq y_i} \sqrt{w(x,i)w(y,i)} = \sum_{i:y_i \neq x_i} w'(x,i) \geq 1$, so this is feasible for the adversary bound. The objective value is n when maximizing over 0-inputs; for 1-inputs, we have $\sum_i w(x,i) = \sum_i w'(x,i)^2 \leq \sum_i w'(x,i) = \mathrm{FC}(f,x)$, since $w'(x,i) \leq 1$ for the optimal fractional certificate (there is no reason to ever use weights above 1). This means the maximum cost of a 0-input is $\mathrm{fbs}_0(f)$, and the maximum cost of a 1-input is n. We can rebalance these as before, by scaling the 0-input and 1-input weights in opposite directions. This gives $\mathrm{Adv}(f) \leq \sqrt{\mathrm{fbs}_0(f)n}$. We could also have done this with 1-inputs, getting $\mathrm{Adv}(f) \leq \sqrt{\mathrm{fbs}_1(f)n}$. This is weaker than the bound $\sqrt{\mathrm{fbs}_0(f)\,\mathrm{fbs}_1(f)}$, but works for partial functions.

Theorem 4.3. *Let f be a (possibly partial) Boolean function. Then* $\mathrm{Adv}(f) \leq \sqrt{\mathrm{fbs}_{\min}(f) \cdot n}$, *where* $\mathrm{fbs}_{\min}(f)$ *denotes the minimum of* $\mathrm{fbs}_0(f)$ *and* $\mathrm{fbs}_1(f)$.

The certificate barrier mostly comes up for total functions. An example of a total function for which we might like to prove a large quantum lower bound, but are blocked by the certificate barrier, is the element distinctness function. This function has alphabet $[m]$ with $m \geq n$, and the goal is to determine whether all the symbols in the input string x are distinct, or whether there is at least one pair (i,j) with $i \neq j$ such that $x_i = x_j$. In the former case, we need to output 0, and in the latter case we output 1. This is a total function, although its alphabet is not Boolean; we can convert to a Boolean alphabet if we wish by using $O(\log m)$ bits to represent each alphabet symbol. This will change the quantum query complexity of the function by at most a factor of $O(\log m)$. We will generally take m to be only slightly larger than n, say $m = n$ or $m = 2n$, so this is only a logarithmic factor in the input size (which we usually ignore).

The certificate complexity of element distinctness on the 1-input side is $\mathrm{C}_1(f) = 2$ (using the alphabet size m formulation; it would be $O(\log m)$ if we convert to Boolean alphabet). Hence the certificate barrier says that $\mathrm{Adv}(f) = O(\sqrt{n})$ for the element distinctness problem. However, it turns out that the true quantum query complexity of element distinctness is $n^{2/3}$, so the positive adversary method is not good enough here.

4.2 The negative adversary

The negative-weight adversary is a generalization of the positive adversary method, originally introduced in [HLŠ07]. We will start by writing down the primal formulation of it. In the primal formulation, instead of requiring

$$\sum_{i:x_i \neq y_i} \sqrt{w(x,i)w(y,i)} \geq 1,$$

(as in the positive adversary), we require exact equality: that the sum on the left is precisely 1 for all x and y with $f(x) \neq f(y)$. Actually, such a requirement cannot always be achieved. Instead, we allow a generalization of the weights $w(x,i)$. Instead of being non-negative real numbers, they are now allowed to be vectors of real numbers. We will denote these vectors by $v_{x,i}$. The vector $v_{x,i}$ will actually be a generalization of $\sqrt{w(x,i)}$, so $\sqrt{w(x,i)}$ would be a one-dimensional version of the vector $v_{x,i}$.

These vectors will be real-valued, and for each i they will all have the same dimension d_i; however, we place no bound on the dimension of the vectors. The constraint will then be that for all x and y with $f(x) \neq f(y)$, we have

$$\sum_{i:x_i \neq y_i} \langle v_{x,i}, v_{y,i} \rangle = 1.$$

The objective value, which we wish to minimize, is the maximum over $x \in \mathrm{Dom}(f)$ of

$$\sum_{i \in [n]} \|v_{x,i}\|_2^2.$$

Definition 4.4. *The negative-weight adversary bound of a (possibly partial) function f, denoted $\mathrm{Adv}^{\pm}(f)$, is the objective value of the optimal solution to the above minimization problem.*

Note that if we were to relax this from equalling 1 to being at least 1, we would get exactly the positive-weight adversary. Why? To convert from a feasible solution $w(x,i)$ to $v_{x,i}$, we will set $v_{x,i}$ to be a 1-dimensional vector which equals $\sqrt{w(x,i)}$. To convert from a feasible solution $v_{x,i}$ for the vector formulation to the weights $w(x,i)$, we set $w(x,i) = \|v_{x,i}\|_2^2$, and Cauchy-Schwartz would give us $\sqrt{w(x,i)w(y,i)} \geq \langle v_{x,i}, v_{y,i} \rangle$. Both conversions preserve feasibility and the objective value, which means that relaxing the $= 1$ to a ≥ 1 in the negative-weight adversary formulation gives exactly the positive-weight adversary.

Why is this the terminology about "negative weights" and "positive weights" when the only difference is the equality or inequality in the constraint? The terminology comes from the dual formulation. An inequality constraint in the primal corresponds to a positive variable in the dual, and an equality constraint in the primal corresponds to an unconstrained (possibly negative) variable in the dual. Hence, in the dual, we really will have positive and negative weights. In the primal, we just have equality or inequality with 1.

4.2.1 SDP formulation

Before we go further, let's rephrase the negative-weight adversary bound as a semidefinite program. For each $i \in [n]$, we will have a symmetric real matrix X_i with rows and columns indexed by $\mathrm{Dom}(f)$. The entries of this matrix will correspond to the inner products of the vectors $v_{x,i}$; that is, $X_i[x,y]$ will correspond to $\langle v_{x,i}, v_{y,i} \rangle$. Because X_i is made of such inner products, it will be positive semidefinite, so we will write $X_i \succeq 0$. The constraints are that for each $x, y \in \mathrm{Dom}(f)$ with $f(x) \neq f(y)$, we have

$$\sum_{i:x_i \neq y_i} X_i[x,y] = 1.$$

The objective value is the maximum over $x \in \mathrm{Dom}(f)$ of $\sum_{i \in [n]} X_i[x,x]$, which is the maximum diagonal entry of $\sum_i X_i$. We introduce a new variable T to represent this maximum. We can then write this as

$$\begin{array}{lllll} \min & & T & & \\ \text{s.t.} & \sum_{i \in [n]} & X_i \circ I & \leq T \cdot I & \\ & \sum_{i:y_i \neq x_i} & X_i[x,y] & = 1 & \forall x, y \in \mathrm{Dom}(f) : \ f(x) \neq f(y) \\ & & X_i & \succeq 0 & \forall i \in [n]. \end{array}$$

It should be clear that a feasible solution $\{v_{x,i}\}$ can be converted to a feasible solution $\{X_i\}$ simply by taking $X_i[x,y] = \langle v_{x,i}, v_{y,i} \rangle$; forming matrices via inner products like this always gives

rise to positive semidefinite matrices. We can also convert in the other direction. A positive semidefinite matrix X_i can always be decomposed into $X_i = MM^T$ for some real matrix M of the same dimensions as X_i. The rows of M then give vectors $v_{x,i}$ such that $X_i[x,y] = \langle v_{x,i}, v_{y,i} \rangle$, and these vectors have the desired properties. This means the above semidefinite program exactly captures $\mathrm{Adv}^{\pm}(f)$. We can also get a program which capture $\mathrm{Adv}(f)$ by relaxing the $= 1$ constraint to a ≥ 1 constraint.

In particular, the dimensions d_i of the vectors $v_{x,i}$ can always be taken to be at most $|\mathrm{Dom}(f)|$; while larger dimensions are allowed, they are never necessary. This semidefinite program formulation also makes it clear that the optimal value can actually be attained; from the original definition of $\mathrm{Adv}^{\pm}(f)$, it might sound like we need to define $\mathrm{Adv}^{\pm}(f)$ as an infimum rather than a minimum, because it is not clear why the optimal value is attained (the set we are optimizing over did not look compact, since the dimension of the vectors was not bounded). With the semidefinite program formulation, we can see that this minimization problem is over a compact convex set, and the minimum is attained.

4.3 An error-dependent version

Before we see why the negative-weight adversary lower bounds quantum query complexity, we will introduce a variant of this adversary bound that allows for errors. We will define $\mathrm{Adv}^{\pm}_{\epsilon}(f)$ to be similar to $\mathrm{Adv}^{\pm}(f)$, except that there is an extra set of vector $v_{x,n+1}$ for each $x \in \mathrm{Dom}(f)$. This extra vector will correspond to querying $f(x)$ directly. That is, suppose that when computing $f(x)$ on input x, you are allowed to "cheat" and directly see the answer. Let's say we place the answer $f(x)$ in position $n+1$ of the vector x. Moreover, while querying normal indices of x costs 1 each, querying position $n+1$ is free. However, there is a limit on the probability with which you can cheat and query $n+1$.

Then $\mathrm{Adv}^{\pm}_{\epsilon}(f)$ effectively allows you to cheat with probability mass ϵ. Motivated by this, we will set $\mathrm{Adv}^{\pm}_{\epsilon}(f)$ to be the minimum of

$$\max_{x \in \mathrm{Dom}(f)} \sum_{i=1}^{n} \|v_{x,i}\|_2^2$$

such that for all $x, y \in \mathrm{Dom}(f)$ with $f(x) \neq f(y)$, we have

$$\sum_{i \in [n+1] : x_i \neq y_i} \langle v_{x,i}, v_{y,i} \rangle = 1,$$

and also, for each $x \in \mathrm{Dom}(f)$,

$$\|v_{x,n+1}\|_2^2 \leq \epsilon.$$

We now claim that $\mathrm{Adv}^{\pm}_{\epsilon}(f)$ can be exactly characterized in terms of $\mathrm{Adv}^{\pm}(f)$.

Theorem 4.5. *Let f be a (possibly partial) function, and let $\epsilon \in [0,1)$. Then*

$$\mathrm{Adv}^{\pm}_{\epsilon}(f) = (1-\epsilon)\,\mathrm{Adv}^{\pm}(f).$$

Proof. In one direction, if we have a feasible solution $\{v_{x,i}\}$ to $\mathrm{Adv}^{\pm}(f)$, we can multiply all the vectors $v_{x,i}$ by $\sqrt{1-\epsilon}$ and add new one-dimensional vectors $v_{x,n+1} = \sqrt{\epsilon}$. This gives a feasible solution for $\mathrm{Adv}^{\pm}_{\epsilon}(f)$ with objective value $(1-\epsilon)\,\mathrm{Adv}^{\pm}(f)$.

In the other direction, fix a solution $\{v_{x,i}\}$ for $\mathrm{Adv}^{\pm}_{\epsilon}(f)$. Further, let $v'_{x,i}$ be any solution for $\mathrm{Adv}^{\pm}(f)$. Then define $u_{x,i} = v_{x,i} \oplus (v'_{x,i} \otimes v_{x,n+1})$ for $x \in \mathrm{Dom}(f)$ and $i \in [n]$. Here $\oplus$ refers to

appending two vectors together, and $\otimes$ refers to the Kronecker (tensor) product, meaning $v'_{x,i} \otimes v_{x,n+1}$ is a vector with one entry for each pair of entry in $v'_{x,i}$ and entry in $v_{x,n+1}$, where this entry equals the product of the two. In other words, if the dimension of $v_{x,i}$ is d for all i and the dimension of $v'_{x,i}$ is d' for all i, then the dimension of $u_{x,i}$ is $d + dd'$ for all i, and the first d entries of $u_{x,i}$ are just the entries of $v_{x,i}$ while the rest of the entries of $u_{x,i}$ are the products between entries of $v_{x,n+1}$ and entries of $v'_{x,i}$.

We claim that $u_{x,i}$ is feasible for $\mathrm{Adv}^{\pm}(f)$: for each x, y with $f(x) \neq f(y)$, we have

$$\sum_{i:x_i \neq y_i} \langle u_{x,i}, u_{y,i} \rangle = \sum_{i:x_i \neq y_i} \langle v_{x,i}, v_{y,i} \rangle + \langle v_{x,n+1}, v_{y,n+1} \rangle \sum_{i:x_i \neq y_i} \langle v'_{x,i}, v'_{y,i} \rangle$$

$$= \sum_{i:x_i \neq y_i} \langle v_{x,i}, v_{y,i} \rangle + \langle v_{x,n+1}, v_{y,n+1} \rangle = 1.$$

The objective value of this solution is the maximum over $x \in \mathrm{Dom}(f)$ of

$$\sum_i \|u_{x,i}\|_2^2 = \sum_i \|v_{x,i}\|_2^2 + \sum_i \|v_{x,n+1}\|_2^2 \|v'x, i\|_2^2 \leq \mathrm{Adv}^{\pm}_{\epsilon}(f) + \epsilon \cdot T,$$

where T is the objective value of $\{v'_{x,i}\}$. Hence we have

$$\mathrm{Adv}^{\pm}(f) \leq \mathrm{Adv}^{\pm}_{\epsilon}(f) + \epsilon \, \mathrm{Adv}^{\pm}(f),$$

or $\mathrm{Adv}^{\pm}(f) \leq \mathrm{Adv}^{\pm}_{\epsilon}(f)/(1-\epsilon)$. □

An astute reader might notice that the above proof assumes the existence of some feasible solution for $\mathrm{Adv}^{\pm}(f)$. That is, the above proof does not eliminate the possibility that $\mathrm{Adv}^{\pm}(f) = \infty$ because the minimization problem has no feasible solution, even though $\mathrm{Adv}^{\pm}_{\epsilon}(f)$ is finite. To truly show that $\mathrm{Adv}^{\pm}(f) = (1-\epsilon)\,\mathrm{Adv}^{\pm}_{\epsilon}(f)$ in all cases, we must show that $\mathrm{Adv}^{\pm}(f)$ always has a feasible solution, which we do in the following lemma.

Lemma 4.6. *Let f be a (possibly partial) function on n bits. Then $\mathrm{Adv}^{\pm}(f) \leq n$.*

Proof. It suffices to find a feasible set of vectors $v_{x,i}$ with objective value n. For each $i \in [n]$, let S_i be the set partial assignments which could occur in the first i positions of an input $x \in \mathrm{Dom}(f)$. For example, if the input alphabet is m, then S_i will consist of strings in $[m]^i$ with $n-i$ stars appended at the end (to form a partial assignment specifying the first i positions in a string of length n). The set S_0 will be $\{*^n\}$.

Then let $v_{x,i}$ be a vector of dimension $|S_{i-1}|$ such that $v_{x,i}[p] = 1$ if p is the partial assignment consisting of the first $i-1$ bits of x, and $vx, i[p] = 0$ for all other partial assignments. This means $\|v_{x,i}\|_2^2 = 1$ for all x and i, so $\sum_i \|v_{x,i}\|_2^2 = n$ for all $x \in \mathrm{Dom}(f)$.

We claim that this solution is feasible. To see this, let $x, y \in \mathrm{Dom}(f)$ be such that $f(x) \neq f(y)$, and consider the smallest $i \in [n]$ such that $x_i \neq y_i$ (such a position i must exist since $f(x) \neq f(y) \Rightarrow x \neq y$). Then x and y agree on their first $i-1$ bits, which means that $v_{x,i}$ and $v_{y,i}$ have a 1 in the same position and $\langle v_{x,i}, v_{y,i} \rangle = 1$. On the other hand, consider any position j such that $j > i$ and $x_j \neq y_j$. Then x and y disagree on their first $j-1$ positions, which means that the vectors $v_{x,j}$ and $v_{y,j}$ have a 1 in different coordinates and $\langle v_{x,j}, v_{y,j} \rangle = 0$. It follows that

$$\sum_{i:x_i \neq y_i} \langle v_{x,i}, v_{y,i} \rangle = 1,$$

so this solution is feasible, as desired. □

4.4 Negative-weight adversary vs. quantum query complexity

We next prove that the negative-weight adversary bound really lower bounds quantum query complexity.

Theorem 4.7. *For all (possibly partial) Boolean functions f, we have*

$$\mathrm{Q}_\epsilon(f) \geq \frac{1}{2}\,\mathrm{Adv}^{\pm}_{2\sqrt{\epsilon(1-\epsilon)}}(f) \geq \frac{(1-2\epsilon)^2}{4}\,\mathrm{Adv}^{\pm}(f).$$

Proof. The second inequality follows from $1 - 2\sqrt{\epsilon(1-\epsilon)} \geq (1-2\epsilon)^2/4$, which we've shown previously in the context of the adversary and hybrid lower bounds. To prove the first inequality, let Q be a T-query quantum algorithm achieving worst-case error ϵ, where $T = \mathrm{Q}_\epsilon(f)$. Use the same notation as in the hybrid and positive adversary proofs. Recall from the positive adversary proof that we saw

$$\| \, |\psi^x_{T+1}\rangle - |\psi^y_{T+1}\rangle \, \|^2 = 2 \sum_{i:x_i \neq y_i} \sum_{t=1}^{T} \Re(\langle \psi^x_t | \Pi_i (I - (U^x)^\dagger U^y) | \psi^y_t \rangle).$$

Also recall that

$$\| \, |\psi^x_{T+1}\rangle - |\psi^y_{T+1}\rangle \, \|^2 = \langle \psi^x_{T+1} - \psi^y_{T+1} | \psi^x_{T+1} - \psi^y_{T+1} \rangle = 2 - 2\Re(\langle \psi^x_{T+1} | psi^y_{T+1} \rangle)$$

$$= 2 - 2\Re(\langle \phi^x_0 | \phi^y_0 \rangle) - 2\Re(\langle \phi^x_1 | \phi^y_1 \rangle),$$

where ϕ^x_0 is the component of ψ^x_{T+1} in which the output register is 0 (that is, the algorithm outputs 0) and ϕ^x_1 is the component of ψ^x_{T+1} in which the output register is 1. Hence we can write

$$1 = \Re(\langle \phi^x_0 | \phi^y_0 \rangle) + \Re(\langle \phi^x_1 | \phi^y_1 \rangle) + \sum_{i:x_i \neq y_i} \sum_{t=1}^{T} \Re(\langle \psi^x_t | \Pi_i (I - (U^x)^\dagger U^y) | \psi^y_t \rangle).$$

Note that we are assuming, for now, that both the input alphabet is Boolean and that the outputs of f are Boolean. The Boolean inputs assumption means that $(U^x)^\dagger U^y$ maps the query-index and query-output registers from $|i\rangle\,|b\rangle$ to $|i\rangle\,|b \oplus y_i \oplus x_i\rangle = |i\rangle\,|1-b\rangle$. Let $N = I \otimes (|0\rangle\langle 0|_B - |0\rangle\langle 1|_B)$ be the matrix which acts as identity except on the query-output register, and on the latter maps $|0\rangle \to |0\rangle - |1\rangle$ and $|1\rangle \to 0$. Observe that for i such that $x_i \neq y_i$, we have $\Pi_i(I - (U^x)^\dagger U^y) = \Pi_i N^\dagger N$. To see this, note that Π_i commutes with both $(I - (U^x)^\dagger U^y)$ and $N^\dagger N$. Also, we have

$$N^\dagger N = I \otimes ((|0\rangle\langle 0|_B - |1\rangle\langle 0|_B)(|0\rangle\langle 0|_B - |0\rangle\langle 1|_B)) = I \otimes (|0\rangle\langle 0|_B + |1\rangle\langle 1|_B - |0\rangle\langle 1|_B - |1\rangle\langle 0|_B),$$

and $I - (U^x)^\dagger U^y$ maps $|i\rangle_I\,|0\rangle_B \to |i\rangle_I\,(|0\rangle_B - |1\rangle_B)$ and maps $|i\rangle_I\,|1\rangle_B \to |i\rangle_I\,(|1\rangle_B - |0\rangle_B)$, so the action restricted to query register i is the same. Hence we have

$$1 = \Re(\langle \phi^x_0 | \phi^y_0 \rangle) + \Re(\langle \phi^x_1 | \phi^y_1 \rangle) + \sum_{i:x_i \neq y_i} \sum_{t=1}^{T} \Re(\langle N\Pi_i \psi^x_t | N\Pi_i \psi^y_t \rangle).$$

This motivates the following choice of vectors $\{v_{x,i}\}$. For a constant c to be determined later, let c_x be $1/c$ if $f(x) = 0$, and let $c_x = c$ if $f(x) = 1$. Then set

$$v_{x,n+1} = (c_x\,|\Re(\phi^x_0)\rangle \oplus |\Re(i\phi^x_0)\rangle) \oplus ((1/c_x)\,|\Re(\phi^x_1)\rangle \oplus |\Re(i\phi^x_1)\rangle).$$

This way, when $f(x) \neq f(y)$, we have $c_x c_y = 1$, and $\langle v_{x,n+1}, v_{y,n+1} \rangle = \Re(\langle \phi_0^x | \phi_0^y \rangle) + \Re(\langle \phi_1^x | \phi_1^y \rangle)$. Note that we've separated out the real and imaginary components of ϕ_0^x and ϕ_1^x into different coordinates, so that the vectors $v_{x,n+1}$ are real-valued. Further, set

$$v_{x,i} = \bigoplus_{t=1}^{T} |\Re(N\Pi_i \psi_t^x)\rangle \oplus |\Re(iN\Pi_i \psi_t^x)\rangle ,$$

so that $\langle v_{x,i}, v_{y,i} \rangle = \sum_{t=1}^{T} \Re(\langle N\Pi_i \psi_t^x | N\Pi_i \psi_t^y \rangle)$. This way, feasibility is immediately assured. The objective value is the maximum over x of

$$\sum_{i \in [n]} \|v_{x,i}\|^2 = \sum_{t=1}^{T} \|N\psi_t^x\|^2 \leq \|N\|^2 T.$$

This is because $|\psi_t^x\rangle$ is a unit vector, and the spectral norm $\|N\|$ is the largest factor by which N can increase the 2-norm of a vector it is applied to. It is not hard to see that $\|N\| = \sqrt{2}$, so $\sum_{i \in [n]} \|v_{x,i}\|^2 \leq 2T$.

Finally, the error parameter of this feasible solution is the maximum over x of

$$\begin{aligned} \|v_{x,n+1}\|^2 = c_x^2 \|\phi_0^x\|^2 + (1/c_x^2)\|\phi_1\|^2 &= (1/c^2)\Pr[Q(x) = f(x)] + c^2 \Pr[Q(x) = 1 - f(x)] \\ &= 1/c^2 + (c^2 - 1/c^2)\Pr[Q(x) = 1 - f(x)] \leq 1/c^2 + (c^2 - 1/c^2)\epsilon \end{aligned}$$

(when $c \geq 1$). Optimizing over $c \geq 1$, we pick $c = (1-\epsilon)^{1/4}/\epsilon^{1/4}$, which gives $\|v_{x,n+1}\|^2 \leq 2\sqrt{\epsilon(1-\epsilon)}$, as desired. □

Although we did not prove it here, the negative-weight adversary method also works to lower bound $\mathrm{Q}(f)$ when the input and output alphabets of f are not Boolean.

4.5 Duality and tightness

So far, we have seen the primal form of the negative-weight adversary, which takes the form of a minimization program. To prove lower bounds, however, it is more useful to look at the dual, which will be a maximization problem; this way, a lower bound on $\mathrm{Q}(f)$ can be shown by giving a feasible solution to the dual program.

Definition 4.8. *The dual form of the negative-weight adversary for a function f can be defined as*

$$\max_{\Gamma} \min_{i \in [n]} \frac{\|\Gamma\|}{\|\Gamma \circ D_i\|},$$

where Γ ranges over real symmetric matrices with rows and columns indexed by $\mathrm{Dom}(f)$ which satisfy $\Gamma[x,y] = 0$ whenever $f(x) = f(y)$. Here D_i is the $\{0,1\}$-matrix with $D_i[x,y] = 1$ if and only if $x_i \neq y_i$, the notation $\circ$ denotes the Hadamard (entrywise) product, and the norm $\|\cdot\|$ is the spectral norm (i.e. $\|A\|$ is the maximum value of $u^T A v$ over vectors u and v with $\|u\|_2 = \|v\|_2 = 1$).

The above definition precisely equals $\mathrm{Adv}^{\pm}(f)$; proving this requires taking the dual of the semidefinite program for $\mathrm{Adv}^{\pm}(f)$, arguing that strong duality holds (so that the dual has the same optimal value as the primal), and then converting the dual formulation into the above form. We will not go over the proof here.

The negative-weight adversary does not suffer from the property testing and certificate barriers. Indeed, it turns out that $\mathrm{Adv}^{\pm}(f)$ exactly captures bounded-error quantum query complexity (up to constant factors).

Theorem 4.9 ([Rei11; LMR+11]). *For all (possibly partial) functions f, we have*

$$\mathrm{Q}(f) = \Theta(\mathrm{Adv}^{\pm}(f)).$$

More explicitly, for any $\epsilon \in (0, 1/2)$, there are constants c_ϵ and C_ϵ such that for all (possibly partial) query functions f, we have

$$c_\epsilon \,\mathrm{Adv}^{\pm}(f) \leq \mathrm{Q}_\epsilon(f) \leq C_\epsilon \,\mathrm{Adv}^{\pm}(f).$$

This remarkable theorem says that bounded-error quantum query complexity is precisely captured by a reasonably simple semidefinite program, up to a constant factor that depends on the error parameter. Note that this property is unique to *bounded-error* quantum query complexity; zero-error quantum query complexity does not have this property!

We've already seen one direction of the proof, showing that $\mathrm{Adv}^{\pm}(f)$ is a lower bound on $\mathrm{Q}(f)$ (we've seen this only for Boolean functions, but the non-Boolean case is not much harder). We will not cover the other direction, which can be found in [LMR+11].

4.6 Composition

For Boolean functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^n \to \{0,1\}$, we denote their block-composition by $f \circ g$; this is the function defined by $f \circ g(x^1x^2 \dots x^n) = f(g(x^1)g(x^2) \dots g(x^n))$, which takes in n separate inputs to g, evaluates g on all of them, and feeds the resulting n-bit string into f.

If f and g are partial functions, we can still define $f \circ g$, though we need to be a little careful. The domain of $f \circ g$ will only contain strings $x^1x^2 \dots x^n \in \{0,1\}^{nm}$ if each x^i is in $\mathrm{Dom}(g)$, and if in addition, the string $g(x^1)g(x^2) \dots g(x^n)$ is in $\mathrm{Dom}(f)$. That is, we will simply promise that each string we encounter along the way will be in the domain of the function it feeds into.

Composition is a common way of constructing Boolean functions, and the query complexity of composed functions is a problem that comes up frequently. It turns out that $\mathrm{D}(f \circ g) = \mathrm{D}(f)\,\mathrm{D}(g)$, which means that at least for deterministic query complexity, the complexity of composed functions is well understood.

For randomized algorithms, things get a little more complicated: in the upper bound direction, we have $\mathrm{R}(f \circ g) = O(\mathrm{R}(f)\,\mathrm{R}(g)\log \mathrm{R}(f))$. The log factor comes from the need to amplify. The idea is that to compute $f \circ g$, we can run an algorithm for f, and whenever it makes a query to a bit of f, we can run the algorithm for g as a subroutine. The issue is that the algorithm for g won't return the right answer with certainty: it will only do so with bounded error. To ensure that the error is small enough that the outer algorithm for f still works, we will need to amplify the algorithm for g so that its error is small, roughly $O(1/\mathrm{R}(f))$. This requires $O(\log \mathrm{R}(f))$ repetitions of the algorithm for g to achieve, and hence $\mathrm{R}(f \circ g) = O(\mathrm{R}(f)\,\mathrm{R}(g)\log \mathrm{R}(f))$. A similar argument means that $\mathrm{Q}(f \circ g) = O(\mathrm{Q}(f)\,\mathrm{Q}(g)\log \mathrm{Q}(f))$. Actually, this is a little subtle, since the quantum algorithm for f will need to call the algorithm for g in superposition; but it turns out that everything works fine and this is not a problem.

What about the other direction: does $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f)\,\mathrm{R}(g))$? This is certainly true for some f and g, and it is even known that the extra log factor is sometimes (but not always) necessary. However, showing this for *all* f and g remains open. Recently, it has been shown that this is false when f and g can be partial functions: $\mathrm{R}(f \circ g)$ can be asymptotically smaller than $\mathrm{R}(f)\,\mathrm{R}(g)$.

As you can see, characterizing composition behavior of query measures can often be tricky; even randomized query complexity is not well-understood in this regard. However, it turns out that the adversary bounds compose perfectly.

Theorem 4.10. *Let f and g be (possibly partial) Boolean functions. Then*

$$\mathrm{Adv}(f \circ g) = \mathrm{Adv}(f)\,\mathrm{Adv}(g),$$

$$\mathrm{Adv}^{\pm}(f \circ g) = \mathrm{Adv}^{\pm}(f)\,\mathrm{Adv}^{\pm}(g).$$

For a proof of the negative-weight composition, see [HLŠ07; Kim12]. The proof of the composition of the positive-weight adversary follows along similar lines, but is easier. Note that in both cases, it's important that f and g are Boolean functions, or at least that the outputs of g and inputs of f are Boolean. This is because in the non-Boolean setting, composition theorems are generally false: one can pick a function g with three possible outputs, say $1, 2, 3$, such that it's hard to determine if $g(x)$ is 1 or 2 but always easy to determine if $g(x) = 3$. One can also pick a function f with input alphabet $1, 2, 3$ which treats input symbols 1 and 2 identically, and cares only about where the 3s are. Composing f and g will then give a function which is much easier to compute than the product of the cost of computing f and the cost of computing g. Composition theorems are therefore only plausible in the Boolean alphabet/output setting.

Note that the above theorem, combined with $\mathrm{Q}(f) = \Theta(\mathrm{Adv}^{\pm}(f))$, implies that $\mathrm{Q}(f \circ g) = \Theta(\mathrm{Q}(f)\,\mathrm{Q}(g))$. In particular, this means the log factor from amplification is never necessary! It turns out that quantum algorithms have a magical ability to work with "noisy" queries without suffering the cost of noise reduction, so long as the noise is "coherent" (it can be reversed) and as long as the output at the end is allowed to make bounded error.

This remarkable property of quantum query complexity allows us to easily determine the query complexity of some highly-composed functions. For instance, consider the recursive AND-OR tree, which is $\mathrm{OR}_2 \circ \mathrm{AND}_2 \circ \mathrm{OR}_2 \circ \cdots \circ \mathrm{AND}_2$. Up to negations, this function is also equal to $\textsc{Nand}_2^k$ where NAND is the negation of the AND function, and the exponent k means we compose NAND with itself k times (so the input size is 2^k). What is the quantum query complexity of this function? Well, we know that $\mathrm{Adv}^{\pm}$ composes perfectly (without even a constant factor), so $\mathrm{Q}(\textsc{Nand}_2^k) = \Theta(\mathrm{Adv}^{\pm}(\textsc{Nand}_2)^k)$. We just need to compute $\mathrm{Adv}^{\pm}(\textsc{Nand}_2)$. It is not hard to see that $\mathrm{Adv}^{\pm}$ stays the same if a function is negated or if its inputs are negated, so $\mathrm{Adv}^{\pm}(\textsc{Nand}_2) = \mathrm{Adv}^{\pm}(\mathrm{AND}_2) = \mathrm{Adv}^{\pm}(\mathrm{OR}_2)$. As for the adversary bound of OR_2, we know that

$$\mathrm{Adv}^{\pm}(\mathrm{OR}_2)^k = \mathrm{Adv}^{\pm}(\mathrm{OR}_2^k) = \mathrm{Adv}^{\pm}(\mathrm{OR}_{2^k}) = \Theta(\mathrm{Q}(\mathrm{OR}_{2^k})) = \Theta(\sqrt{2^k}).$$

Hence for all k, $\mathrm{Adv}^{\pm}(\mathrm{OR}_2) = \Theta(\sqrt{2^k})^{1/k}$, which necessarily means that $\mathrm{Adv}^{\pm}(\mathrm{OR}_2) = \sqrt{2}$. Indeed, a similar argument shows that for all n, $\mathrm{Adv}^{\pm}(\mathrm{OR}_n)$ must be exactly $\sqrt{n}$. From this we conclude that the quantum query complexity of $\textsc{Nand}_2^k$ is $\Theta(2^{k/2})$, and more generally, for any AND-OR tree of any form, its adversary bound is always the square root of its input size, and its quantum query complexity is within a constant factor of that.

Another example. Let $f = \mathrm{MAJ}_3$, the majority function on 3 bits. What is the query complexity of f^k, the function on inputs of size 3^k consisting of a depth-k tree of majorities on 3 bits each? Well, we know that $\mathrm{Q}(f^k) = \Theta(\mathrm{Adv}^{\pm}(f^k)) = \Theta(\mathrm{Adv}^{\pm}(f)^k)$, so all we need to do is to compute $\mathrm{Adv}^{\pm}(\mathrm{MAJ}_3)$. It turns out to be 2, so $\mathrm{Q}(f^k) = \Theta(2^k) = \Theta(n^{\log_3 2}) \approx n^{0.631}$, where $n = 3^k$ is the input size. In contrast, the randomized query complexity $\mathrm{R}(f^k)$ is actually open!

Directly using the negative-weight adversary (rather than merely its composition properties) is generally difficult. Pretty much the only example where this has been done is [BŠ13].

References

[BŠ13] Aleksandrs Belovs and Robert Špalek. "Adversary Lower Bound for the K-sum Problem". In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. 2013. DOI: 10.1145/2422436.2422474. URL: http://doi.acm.org/10.1145/2422436.2422474 (p. 10).

[HLŠ07] Peter Høyer, Troy Lee, and Robert Špalek. "Negative weights make adversaries stronger". In: *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC 2007)*. 2007. DOI: 10.1145/1250790.1250867 (pp. 3, 10).

[Kim12] Shelby Kimmel. "Quantum Adversary (Upper) Bound". In: *Automata, Languages, and Programming*. Vol. 7391. 2012. DOI: 10.1007/978-3-642-31594-7_47 (p. 10).

[LMR+11] Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. "Quantum query complexity of state conversion". In: *Foundations of Computer Science (FOCS 2011)*. 2011. DOI: 10.1109/FOCS.2011.75. eprint: 1011.3020 (p. 9).

[Rei11] Ben W Reichardt. "Reflections for quantum query algorithms". In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms (SODA 2011)*. 2011. DOI: 10.1137/1.9781611973082.44. eprint: 1005.1601 (p. 9).