Week 5

Polynomials, Part 1: Symmetrization

5.1 Representing functions by polynomials

So far, we have seen the positive adversary method (easy to use but doesn't always work) and the negative adversary method (always works but very hard to use). This week we will introduce a different type of lower bound technique, called the polynomial method. Its power is incomparable to the positive adversary (so it can sometimes be stronger), and using it is usually harder than the positive adversary but easier than the negative adversary.

To start, we will define the polynomial degree of a Boolean function. We start with the following observation.

Lemma 5.1. *Let $f : \{0,1\}^n \to \mathbb{R}$ be any function from the Boolean hypercube to the reals. Then f can be uniquely represented as a multivariate polynomial $p(x_1, x_2, \ldots, x_n)$ with real coefficients, such that $p(x) = f(x)$ for all $x \in \{0,1\}^n$. The polynomial p will be multilinear, which means that each variable x_i only occurs with degree 0 or 1 in each monomial.*

Proof. First, note that for $x_i \in \{0,1\}$, we have $x_i^2 = x_i$, so terms like x_i^2 or x_i^3 don't help – they are equivalent to x_i. With this in mind, there are 2^n different subsets of the set $[n]$, and for each subset $S \subseteq [n]$ there is one potential monomial $m_S = \prod_{i \in S} x_i$. Any polynomial p over $\{0,1\}^n$ is therefore a linear combination of these 2^n monomials m_S for different subsets S.

Consider the monomials m_S as functions; that is, for each S let $m_S(x)$ denote the product $\prod_{i \in S} x_i$, so that $m_S : \{0,1\}^n \to \{0,1\}$ is a Boolean function. Further, associate with this function m_S a long vector v_S representing the truth table of m_S with one entry for each $x \in \{0,1\}^n$, i.e. the vector with $v_S[x] = m_S(x)$. We claim that the 2^n vectors v_S for $S \subseteq [n]$ are all linearly independent. To see this, consider a linear combination of them that equals the all-zero vector. This linear combination is some polynomial q in the variables $x_1, x_2, \ldots, x_n$ such that $q(x) = 0$ for all $x \in \{0,1\}^n$. Let m_S be a monomial of q that has non-zero coefficient and such that $|S|$ is as small as possible. Consider the string x with $x_i = 1$ for $i \in S$ and $x_i = 0$ otherwise, and consider the value of $q(x)$. Each monomial of q other than m_S must use some variable outside of S, by the minimality of the choice of S; hence each monomial of p other than m_S evaluates to 0 on x. However, $m_S(x) = 1$, and since m_S has a non-zero coefficient in p, we have $p(x) \neq 0$. This is a contradiction, which means that the monomials are linearly independent, as desired.

Finally, consider the arbitrary function $f : \{0,1\}^n \to \mathbb{R}$. The truth table of this function is a vector in $\mathbb{R}^{2^n}$ with one entry for each $x \in \{0,1\}^n$. Since $\mathbb{R}^{2^n}$ has dimension 2^n, and since the vectors v_S are 2^n independent vectors, they are a basis. This means the truth table of f can be uniquely written as a linear combination of the vectors v_S, which means f can be uniquely represented as a real polynomial p. □

Polynomials of this form, in which each variable occurs with degree 0 or 1, are called *multilinear*.

We now define the degree of a Boolean function, as follows.

Definition 5.2. *For a function $f\colon \{0,1\}^n \to \mathbb{R}$, define its degree $\deg(f)$ to be the degree of its corresponding real polynomial. That is, the maximum value of $|S|$ for monomials m_S that have a non-zero coefficient in the real polynomial p computing f.*

Note that the degree of a function on n bits is always at most n. Also note that although we allowed f to output real numbers, this definition also works perfectly well for Boolean functions that have outputs in $\{0,1\}$. The input and output alphabets should be Boolean for the degree to make sense, at least as we've currently defined it.

We can also extend this definition to partial functions. There are actually two ways of doing so.

Definition 5.3. *Let f be a possibly partial Boolean function on n bits. Then $\deg(f)$ is the minimum degree of a real polynomial p satisfying $p(x) = f(x)$ for $x \in \mathrm{Dom}(f)$.*

An alternative definition is to take the minimum degree of p only over polynomials p satisfying both $p(x) = f(x)$ for all $x \in \mathrm{Dom}(f)$, and also $p(x) \in [0,1]$ for all $x \in \{0,1\}^n$.

So far we have taken polynomial degree where the inputs and outputs are $\{0,1\}$-valued. It is often useful to consider polynomials that take inputs in $\{+1,-1\}^n$ and give outputs in $\{+1,-1\}$ instead. As it turns out, the degree of a function when represented over $\{+1,-1\}$ is the same as when it is represented over $\{0,1\}$. We will associate $+1$ with 0 and -1 with 1, so that we can convert from $\{0,1\}$ to $\{+1,-1\}$ by taking $(-1)^b$ or $1-2b$, and by taking $(1-b)/2$ to convert back from $\{+1,-1\}$ to $\{0,1\}$.

We refer these as different *bases*: the $\{0,1\}$ basis and the $\{+1,-1\}$ basis. The word "basis" comes from the fact that, as we've seen, the monomials m_S form a basis when considered as vectors over $\mathbb{R}^{2^n}$ (with these vectors representing the truth table of the function m_S, i.e. the vectors $v_S[x] = m_S(x)$). It turns out that monomials m_S over $\{+1,-1\}$ variables also form a basis for $\mathbb{R}^{2^n}$. Note that these are different functions now: the function $m_S(x) = \prod_{i \in S} x_i$ is an AND function when x is a $\{0,1\}^n$ vector, but it is a PARITY function when x is a $\{+1,-1\}^n$ vector.

Lemma 5.4. *A polynomial p in the $\{0,1\}$ basis can be converted into a polynomial q in the $\{+1,-1\}$ basis, and vice versa. These conversions preserve the behavior of the polynomials as functions, and also the degree of the polynomials.*

Proof. If p expects $\{0,1\}$ inputs, we can make it take $\{+1,-1\}$ inputs instead by plugging in $(1-x_i)/2$ into each variable x_i of p. Since $(1-x_i)/2$ converts $\{+1,-1\}$ into $\{0,1\}$, this preserves the behavior of p. When we expand and simplify, it's not hard to see that the degree of p cannot increase, since we are plugging in degree-1 terms into p. We can also change the output of p from $\{0,1\}$ to $\{+1,-1\}$ by applying $1-2p$ to the polynomial, which does not affect its degree.

To convert back, we similarly plug in $(1-2x_i)$ into x_i in q, and apply $(1-q)/2$ on the outside. Once again, this transformation can only decrease the degree. However, since applying both transformations gets us back to exactly the same function (and hence exactly the same polynomial, since each function has a unique polynomial), we conclude that neither transformation can actually decrease the degree, and both must preserve it. □

Note that polynomials taking in $\{+1,-1\}$ variables can still be assumed to be multilinear, since now $x_i^2 = 1$, $x_i^3 = x_i$, etc., and there is still no need for x_i to have degrees other than 0 or 1.

5.2 Approximating polynomials

Having defined the degree of a Boolean function, we now define a related notion called the approximate degree.

Definition 5.5 (Approximating polynomial)**.** *We say that a polynomial p approximates a (possibly partial) Boolean function f to error ϵ if $|p(x) - f(x)| \leq \epsilon$ for all $x \in \mathrm{Dom}(f)$. This is for $\{0,1\}$ outputs; for $\{+1,-1\}$ outputs, the values get stretched by a factor of 2, so we require instead that $|p(x) - f(x)| \leq 2\epsilon$.*

Definition 5.6 (Bounded polynomial)**.** *We say that a real polynomial p in n variables is bounded if $p(x) \in [0,1]$ for all $x \in \{0,1\}^n$. This is when dealing with $\{0,1\}$ outputs; in the context of $\{+1,-1\}$ outputs, we will say p is bounded if $|p(x)| \leq 1$ for all x in the Boolean hypercube.*

With these definitions in hand, we can define the approximate degree of a Boolean function.

Definition 5.7. *Let f be a (possibly partial) Boolean function on n bits. The approximate degree of f to error ϵ, denoted $\widetilde{\deg}_\epsilon(f)$, is the minimum degree of a bounded polynomial p which approximates f to error ϵ. When $\epsilon = 1/3$, we omit it and write $\widetilde{\deg}(f)$.*

It turns out that approximate polynomial degree lower bounds quantum query complexity. Before we see this, let's see why it lower bounds deterministic and randomized query complexities.

Theorem 5.8. *Let f be a (possibly partial) Boolean function. Then $\mathrm{D}(f) \geq \deg(f)$.*

Proof. We will show that each decision tree D has a corresponding polynomial p_D which computes the same function as D and that has degree at most the height of D. We proceed by induction over the height of D. If the height of D is 0, meaning D is a constant, we can represent it by a constant polynomial (either the constant 0 or the constant 1). These polynomials have degree 0, completing the base case.

Suppose all decision trees of height at most k can be represented by polynomials in this fashion, and let D be a decision tree of height $k+1$. Then D starts by querying some position x_i of the input x, and if $x_i = 0$ it goes to the root of subtree D_0, while if $x_i = 1$ it goes to the root of subtree D_1. These subtrees have height at most k, so they are represented by polynomials p_{D_0} and p_{D_1} of degree at most k. We claim that $p = (1 - x_i)p_{D_0} + x_i p_{D_1}$ is a polynomial computing the same function as D. To see this, note that if $x_i = 0$, p outputs the same as p_{D_0}, which is the same as the output of D_0; on the other hand, if $x_1 = 1$, p outputs the same as p_{D_1}, which is the same as the output of D_1. In both cases, $p(x) = D(x)$. Moreover, the degree of p is at most $k+1$, since p_{D_0} and p_{D_1} have degree at most k. This completes the induction argument.

Finally, recall that $\mathrm{D}(f)$ is the minimum height of a decision tree computing f. This minimum decision tree converts into a polynomial computing f, and this polynomial has degree at most $\mathrm{D}(f)$, so $\mathrm{D}(f) \geq \deg(f)$. □

While the exact degree of a Boolean function lower bounds $\mathrm{D}(f)$, to lower bound $\mathrm{R}(f)$ we will need the approximate degree.

Theorem 5.9. *Let f be a (possibly partial) Boolean function. Then $\mathrm{R}_\epsilon(f) \geq \widetilde{\deg}_\epsilon(f)$.*

Proof. Recall that a randomized query algorithm is a probability distribution over decision trees. Let R be the one that minimizes $\mathrm{R}_\epsilon(f)$, so that each decision tree in the support of R has height at most $\mathrm{R}_\epsilon(f)$. Let $p = \sum_D \Pr[D] \cdot p_D$, where the sum ranges over decision trees D in the support of R, $\Pr[D]$ denotes the probability of D in R, and p_D is the polynomial computing the same function

as the decision tree D. Then p_D has degree at most $\mathrm{R}_\epsilon(f)$ for all D, and p is a linear combination of such polynomials, so it has degree at most $\mathrm{R}_\epsilon(f)$.

Moreover, for each $x \in \mathrm{Dom}(f)$, we have $\Pr[R(x) \neq f(x)] \leq \epsilon$, which means $\Pr_{D \sum R}[p_D(x) \neq f(x)] \leq \epsilon$. It is not hard to see that $\Pr_{D \sum R}[p_D(x) \neq f(x)] = |p(x) - f(x)|$ when $f(x)$ is $\{0,1\}$-valued, so we have $|p(x) - f(x)| \leq \epsilon$ for all $x \in \mathrm{Dom}(f)$. Finally, it is clear that $p(x) \in [0,1]$ for all $x \in \{0,1\}^n$, because each $p_D(x)$ is in $\{0,1\}$ and $p(x)$ is a convex combination of $p_D(x)$ terms. □

As it turns out, the approximate degree also lower bounds quantum query complexity, not just randomized query complexity.

Theorem 5.10. *Let f be a (possibly partial) Boolean function. Then $\mathrm{Q}_\epsilon(f) \geq \widetilde{\deg}_\epsilon(f)/2$.*

Proof. Let Q be a T-query quantum algorithm computing f to error ϵ, where $T = \mathrm{Q}_\epsilon(f)$. Then Q is a sequence of unitaries $U_0, U_1, \ldots, U_T$, and the output of $Q(x)$ is the result of querying the output register of

$$U_T U^x U_{T-1} U^x \ldots U^x U_1 U^x U_0 \left|\psi\right\rangle$$

for some initial state $|\psi\rangle$. Using our usual notation, let $|\psi_t^x\rangle$ be the state of the algorithm right before query number t, so $|\psi_t^x\rangle = U_{t-1} U^x U_{t-2} U^x \ldots U^x U_0 |\psi\rangle$.

Consider the amplitudes of $|\psi_t^x\rangle$, that is, the entries of the vector. When $t = 1$, no queries have been made yet, and so these entries have no dependence on x; they are constants relative to x. When a query is made, that is, when U^x applied, we map $|i\rangle_I |b\rangle_B \to |i\rangle_I |b \oplus x_i\rangle_B$, which is the same as mapping

$$|i\rangle_I |b\rangle_B \to x_i |i\rangle_I |1-b\rangle_B + (1-x_i) |i\rangle_I |b\rangle_B .$$

If the amplitudes of $|\psi_t^x\rangle$ where $\{\alpha\}$, the amplitudes of $U^x |\psi_t^x\rangle$ are combinations of $x_i\alpha$ and $(1-x_i)\alpha$ for different values of i and α. That is, applying U^x multiplies some of the amplitudes by x_i or $1 - x_i$, and potentially recombines the resulting complex numbers in a linear combination. The unitary U_t applies a linear map to the amplitudes that does not depend on x. Hence, going from $|\psi_t^x\rangle$ to $|\psi_{t+1}^x\rangle$, the amplitudes got multiplied by x_i or $(1 - x_i)$, and then rearranged in a linear combination.

Since the amplitudes of $|\psi_1^x\rangle$ are constants, we conclude that the amplitudes of $|\psi_t^x\rangle$ are all polynomials over the variables $x_1, x_2, \ldots, x_n$ with degree at most $t - 1$. The final state, $|\psi_{T+1}^x\rangle$, has amplitudes that are polynomials of degree at most T. Although they have potentially complex coefficients, these can be separated into a real polynomial coefficient plus i times another real polynomial coefficient. When the output register is measured, these real polynomials get squared and added together to form the acceptance probability. Squaring polynomials can double their degree, so the final acceptance probability of $Q(x)$ is a polynomial p in the variables $x_1, x_2, \ldots, x_n$ of degree at most $2T$.

Since p computes an acceptance probability, we have $p(x) \in [0,1]$ for all $x \in \{0,1\}^n$, even when x is not in $\mathrm{Dom}(f)$ (since in all cases the quantum algorithm does *something* and has *some* probability of accepting). Moreover, since Q computes f to error ϵ, we have $|p(x) - f(x)| \leq \epsilon$, which means that p approximates f to error ϵ. This shows that $\widetilde{\deg}_\epsilon(f) \leq 2T$, as desired. □

5.3 Symmetrization

Now that we've seen that $\widetilde{\deg}(f)$ lower bounds $\mathrm{Q}(f)$, it remains to get a handle on determining the approximate degree of Boolean functions. This is often tricky to do, but it becomes much easier if the function is symmetric, due to a trick called symmetrization.

Theorem 5.11. *Let p be an n-variate multilinear polynomial. Then there is a single-variate polynomial q such that for all $t \in \{0, 1, 2, \ldots, n\}$, $q(t)$ evaluates to the average of $p(x)$ over all strings $x \in \{0,1\}^n$ of Hamming weight t. Further, the degree of q is at most the degree of p.*

Proof. The trick is to "symmetrize" the polynomial p by averaging over all permutations of the variables. That is, for each permutation $\sigma \in S_n$, let p_σ be the polynomial $p_\sigma(x) = p(x_\sigma)$, where x_σ is the string $(x_\sigma)_i = x_{\sigma(i)}$. That is, p_σ is the polynomial we get if we permute the input string x according to σ before we applying p to the result. Then let

$$p_{sym}(x) = \frac{1}{n!} \sum_{\sigma \in S_n} p_\sigma(x).$$

Note that each p_σ has the same degree as p, so the degree of p_{sym} is at most the degree of p (it might be lower due to cancellations). Moreover, the degree p_{sym} is symmetric in its variables; that is, $p_{sym}(x_\sigma) = p_{sym}(x)$ for all σ.

It turns out that symmetric multilinear polynomials can always be written as a linear combination of the elementary symmetric polynomials. The first elementary symmetric polynomial is $J_0 = 1$, the second is $J_1 = \sum_i x_i$, the third is $J_2 = \sum_{i<j} x_i x_j$, and so on. Note that the degree of J_t is t. Since p_{sym} is symmetric, we can write

$$p_{sym} = \sum_{t=0}^{\deg(p)} c_t J_t$$

for some real coefficients c_t. Next, note that $J_1^2 = J_2 + \sum_i x_i^2 = J_2 + J_1$, since $x_i^2 = x_i$. Hence $J_2 = J_1^2 - J_1$. We similarly have $J_3 = J_2 J_1 - 2J_2 = J_1^3 - 3J_1^2 + 2J_1$. More generally, each J_t can be written as a degree-t polynomial in J_1. Hence p_{sym} can be written as $q(J_1)$ with q being the polynomial $q(t) = \sum_{j=0}^{\deg(p)} d_j t^j$ for some real coefficients d_j. We conclude that $q(t)$ computes the average of $p(x)$ over strings of Hamming weight t, as desired. □

This theorem is primarily useful when analyzing the approximate degree of symmetric functions. For example, let's use it to lower bound the approximate degree of PARITY.

Lemma 5.12. $\widetilde{\deg}_\epsilon(\text{PARITY}) = n$ *for all* $\epsilon < 1/2$.

Proof. Suppose p approximates PARITY, and consider its outputs in $\{+1, -1\}$ form. Let q be the single-variate symmetrization of p, so that $q(t)$ is the average of $p(x)$ over $x \in \{0,1\}^n$ with Hamming weight t. Note that for x with Hamming weight t, $p(x)$ must approximate $(-1)^t$, so $p(x)(-1)^t \in [1 - 2\epsilon, 1]$. Hence the average of $p(x)$ over x of Hamming weight t still has this property, so $q(t)(-1)^t \geq [1 - 2\epsilon]$ for all $t = 0, 1, \ldots n$. In particular, $q(t)$ must be positive when t is even and negative when t is odd. This means that in the range $[0, n]$, $q(t)$ must cross the x-axis at least n times. It follows that $q(t)$ must have degree at least n, and since its degree is at most that of p, p must also have degree at least n. Hence $\widetilde{\deg}_\epsilon(\text{PARITY}) = n$. □

Note that this lemma implies that $\mathrm{Q}_\epsilon(\text{PARITY}) \geq n/2$ for all $\epsilon < 1/2$; that is, a quantum algorithm cannot even achieve a small bias towards computing the parity of a string in fewer than $n/2$ queries (it turns out that $n/2$ can be achieved by an exact quantum algorithm when n is even, so this is tight).

Next, we consider the approximate degree of PROMISEOR. Recall that PROMISEOR is the function which outputs 0 when the Hamming weight of the string is 0, outputs 1 when the Hamming weight of the string is 1, and is undefined elsewhere. A polynomial p which approximates

PROMISEOR, and which is bounded, turns into a single-variate polynomial $q(t)$ with $q(0) \leq \epsilon$, $q(1) \geq 1-\epsilon$, and $q(t) \in [0,1]$ for all $t = 0, 1, 2, \ldots, n$. This polynomial q moves quickly between $q(0)$ and $q(1)$ (going up by $1-2\epsilon$ in the y-axis for a shift of 1 in x-axis, meaning its derivative was at least $1-2\epsilon$ at some point). However, this polynomial is also stuck inside $[0,1]$ on all the points $\{0, 1, 2, \ldots, n\}$.

The following property of single-variate real polynomials will come in handy.

Theorem 5.13 (Markov Brothers' inequality)**.** *Let $p\colon \mathbb{R} \to \mathbb{R}$ be a real polynomial of degree d. If $p(x) \in [b, b']$ for all $x \in [a, a']$, then $|p'(x)| \leq \frac{b'-b}{a'-a} d^2$ for all $x \in [a, a']$.*

This theorem says that if a polynomial is stuck in a rectangular box, which is $a'-a$ and $b'-b$ high, then it cannot have a high derivative unless it has high degree. In particular, if the width of the box is n and the height is 1, then the derivative of the polynomial in the box must be at most d^2/n, which means that if the polynomial moves sharply—getting derivative $\Omega(1)$—then it must have degree at least $\Omega(\sqrt{n})$.

Markov brothers' inequality was proven in the 1800s, though its proof is too technical to include here. Instead, we will see how it can be used to imply a lower bound on the approximate degree of PROMISEOR. First, we will need the following corrolary, first shown by [EZ64; RC66]. This corollary gives a lower bound on the degree of a polynomial even if it is only known to stay in the box on integer points, rather than on all points.

Corollary 5.14. *Let $p\colon \mathbb{R} \to \mathbb{R}$ be a real polynomial. Let $n \in \mathbb{N}$ be a positive integer, and suppose $p(x) \in [0,1]$ for all $x \in \{0, 1, 2, \ldots n\}$. Then $\deg(p) \geq \sqrt{nc/(1+c)}$, where c is the maximum value of $|p'(x)|$ for $x \in [0, n]$.*

Proof. Let c be the maximum value of $|p'(x)|$ for $x \in [0, n]$. Note that if $p(x)$ is ever outside of $[0,1]$ by more than δ for some $x \in [0, n]$, then x has distance at most $1/2$ to some $i \in \{0, 1, 2, \ldots, n\}$, which means that p changed by at least δ from $p(i)$ to $p(x)$ (with an x-axis change of at most $1/2$). By the mean value theorem, we have $|p'(y)| \geq 2\delta$ for some y between x and i, so $y \in [0, n]$. This means that $c \geq 2\delta$, or $\delta \leq c/2$. In other words, we must have $p(x) \in [-c/2, 1+c/2]$ for all $x \in [0, n]$.

We now have bounds on $p(x)$, so we use Markov brothers' inequality. This tells us that $|p'(x)| \leq (1+c)\deg(p)^2/n$ for $x \in [0, n]$, so $c \leq (1+c)\deg(p)^2/n$. Rearranging, we get $\deg(p) \geq \sqrt{nc/(1+c)}$, as desired. □

With this tool in hand, we can prove a lower bound on $\widetilde{\deg}(\text{PROMISEOR})$.

Theorem 5.15. *For all $\epsilon \in [0, 1/2)$ and all $n \in \mathbb{N}$, we have*

$$\widetilde{\deg}_\epsilon(\text{PROMISEOR}_n) \geq \sqrt{\frac{1-2\epsilon}{2(1-\epsilon)} n}.$$

In particular, $\widetilde{\deg}(\text{PROMISEOR}_n) \geq \sqrt{n}/2$.

Proof. Let p be a bounded polynomial approximating PROMISEOR to error ϵ, and let q be the single-variate symmetrization polynomial with degree at most that of p. Then $q(t) \in [0,1]$ for $t \in \{0, 1, 2, \ldots, n\}$. Let d be the degree of q, so $d \leq \widetilde{\deg}_\epsilon(\text{PROMISEOR})$. By the above corollary, we have $d \geq \sqrt{nc/(1+c)}$, where c is the maximum value of $q'(t)$ for $t \in [0, n]$. Note that $q(0) \leq \epsilon$ and $q(1) \geq 1-\epsilon$, so by the mean value theorem, $q'(t) \geq 1-2\epsilon$ for some $t \in [0,1]$, from which the desired result follows. □

Recall from the first week that if a measure M satisfies some simple properties, then for all f we have $M(f) \geq M(\textsc{PromiseOR}_{\mathrm{bs}(f)})$. Because of this, we get the following corollary, first observed by Beals, Buhrman, Cleve, Mosca, and de Wolf [BBC+01].

Corollary 5.16. *Let f be a (possibly partial) Boolean function. Then $\widetilde{\deg}(f) \geq \sqrt{\mathrm{bs}(f)}/2$.*

We also have a corollary for quantum algorithms, which has a better dependence on ϵ when ϵ is close to $1/2$.

Corollary 5.17. *Let f be a (possibly partial) Boolean function. Then*

$$\mathrm{Q}_\epsilon(f) \geq \sqrt{\frac{1-2\epsilon}{8(1-\epsilon)}\,\mathrm{bs}(f)} \geq \sqrt{(1-2\epsilon)\,\mathrm{bs}(f)/8}.$$

Next, let's lower bound the approximate degree of majority. We can already prove $\mathrm{Q}(\mathrm{MAJ}_n) = \Omega(n)$ using the positive adversary method, but this doesn't directly imply that $\widetilde{\deg}(\mathrm{MAJ}_n) = \Omega(n)$, and the latter can be considered as a question of independent interest. We have $\mathrm{bs}(\mathrm{MAJ}_n) = (n+1)/2$ (where we assume n is odd), so we know that $\widetilde{\deg}(\mathrm{MAJ}_n) = \Omega(\sqrt{n})$. Can we improve this?

If we start with a polynomial p approximating MAJ_n and symmetrize it, we get a polynomial q in one variable with $q(t) \in [0,\epsilon]$ for $t \in \{0,1,\ldots,(n-1)/2\}$ and $q(t) \in [1-\epsilon,1]$ for $t \in \{(n+1)/2,\ldots,n\}$. Using Corollary 5.14 on this polynomial will only give an $\Omega(\sqrt{n})$ lower bound on the degree of q. To improve this, we will use yet another old theorem from approximation theory.

Theorem 5.18. *Let $p\colon \mathbb{R} \to \mathbb{R}$ be a real polynomial of degree d. If $p(x) \in [b,b']$ for all $x \in [a,a']$, then for all $x \in [a,a']$, we have*

$$|p'(x)| \leq \frac{b'-b}{2\sqrt{(x-a)(a'-x)}}d.$$

As before, we need a version of this theorem in which the polynomial is only known to be bounded on integer points. This was shown by Paturi [Pat92] (the proof is quite tricky).

Theorem 5.19 (Paturi)**.** *Let $p\colon \mathbb{R} \to \mathbb{R}$ be a real polynomial. Let $n \in \mathbb{N}$ be a positive integer, and suppose $p(x) \in [0,1]$ for all $x \in \{0,1,2,\ldots,n\}$. Then for all $x \in [0,n]$, we have*

$$\deg(p) \geq \frac{|p'(x)|}{C(1+|p'(x)|)}\sqrt{x(n-x)}$$

where C is some universal constant.

Note that Paturi's theorem improves on Corollary 5.14 when x is near $n/2$; if $|p'(x)|$ is at least a constant for such x, then Paturi gives $\deg(p) = \Omega(n)$ instead of $\Omega(\sqrt{n})$. In other words, we knew from before that if a polynomial stays bounded in a box of width n and height 1, and if at some point it has constant derivative, then it must have degree at least $\Omega(\sqrt{n})$; we now know that unless the constant derivative only happens near the left/right edges of the box, the polynomial must in fact have degree at least $\Omega(n)$. Further, this holds even when the polynomial only stays in the box on integer points.

Using Paturi's theorem, we can prove an approximate degree lower bound for majority.

Theorem 5.20. $\widetilde{\deg}(\mathrm{MAJ}_n) = \Omega(n)$.

Proof. Let p be a polynomial approximating MAJ_n to error ϵ, and let q be the symmetrization of p. Then $q(t) \in [0,1]$ for all $t \in \{0,1,2,\ldots,n\}$. Now, we know that $q((n-1)/2) \leq \epsilon$ and $q((n+1)/2) \geq 1-\epsilon$, so by the mean value theorem, some point t in between them has $q'(t) \geq 1-2\epsilon$. Paturi's theorem then gives $\deg(q) = \Omega((1-2\epsilon)n)$, which is $\Omega(n)$ when $\epsilon = 1/3$. □

5.4 Larger alphabets

The polynomial degree measures we've defined so far are specific to functions that have Boolean inputs and outputs. We can generalize this to a notion of polynomials that can handle larger input and output alphabets.

To handle larger output alphabets, we will use a collection of polynomials $\{p_c\}$ for each symbol c in the output alphabet. The polynomial $p_c(x)$ will represent the probability that a quantum algorithm outputs the symbol c when run on x. Such a collection will be required to have $\sum_c p_c(x) = 1$ for all $x \in \{0,1\}^n$. It will also be required to have $p_c(x) \geq 1 - \epsilon$ if $f(x) = c$. Its degree will be the maximum degree of p_c for any c.

More interesting is the question of how to handle larger input alphabets. It turns out that the natural way to do so is as follows. Instead of defining a polynomial in n variables $x_1, x_2, \ldots, x_n$ where $x \in \{0,1\}^n$ is the input string, we will instead define a polynomial in nm variables, where m is the size of the input alphabet. The variable y_{ij} for $i \in [n]$ and $j \in [m]$ will be in indicator variable for whether $x_i = j$, where $x \in [m]^n$ is the input. That is, $p(y)$ will be a polynomial in nm variables, and applying p to x works by evaluating $p(y_x)$, where y_x is the string with $(y_x)_{ij} = 1$ if $x_i = j$ and $(y_x)_{ij} = 0$ otherwise. As before, we require that $|p(y_x) - f(x)| \leq \epsilon$ for all $x \in \mathrm{Dom}(f)$ (assuming f has Boolean outputs; see above if not). For boundedness, we will require that $p(y_x) \in [0,1]$ for all $x \in [m]^n$, but not necessarily for all $y \in \{0,1\}^{nm}$.

Actually, when dealing with large input alphabets, it is often convenient to consider an extra symbol $\perp$ which is promised never to occur. When $\perp$ does occur in an input, a quantum algorithm accepting that input may output anything. We will replace the above boundedness requirement with the stronger requirement that $p(y_x) \in [0,1]$ for all $x \in ([m] \cup \{\perp\})^n$, where $(y_x)_{ij} = 0$ for all j when $x_i = \perp$.

The above generalizations allow us to write $\widetilde{\deg}(f)$ for functions f with non-Boolean inputs and/or outputs. We will primarily care about the case of non-Boolean inputs, so we will often assume f has Boolean outputs for simplicity. It turns out that $\mathrm{Q}(f) \geq \widetilde{\deg}(f)/2$ still holds for these generalized notions of approximate degree, so we can still use polynomials to lower bound quantum algorithms in this setting.

One thing that will come in handy is a notion of symmetrization for polynomials with non-Boolean input alphabets.

Theorem 5.21. *Let p be a polynomial in the variables y_{ij} for $i \in [n]$, $j \in [m]$. Then there is a polynomial $q(z)$ in m variables such that $\deg(q) \leq \deg(p)$ and for all $z \in [n]^m$ with $\sum_{j=1}^m z_j \leq n$, the value of $q(z)$ equals the average of $p(y_x)$ over all strings $x \in ([m] \cup \{\perp\})^n$ such that z_j is the number of times the symbol j occurs in x.*

This theorem statement is a little confusing. Think of it as follows: instead of having a single new variable t corresponding to the Hamming weight, and have a new polynomial $q(t)$ act only in t, we now have m different "Hamming weights", each for a different alphabet symbol. These Hamming weights are $z_1, z_2, \ldots, z_m$, and must sum to at most n, since there are n total symbols in a string of length n (they need not sum to exactly n due to the possibility of the symbol $\perp$ showing up). The new polynomial q will act in the m variables $z_1, z_2, \ldots, z_m$, and will evaluate to the average of $p(y_x)$ over all strings x that are of type z (i.e. all strings x for which z correctly specifies the number of times each alphabet symbol in $[m]$ occurs). It turns out that q is still a low-degree polynomial in the variables z_j (with degree at most that of p), though it does not need to be multilinear.

Proof. Start with a polynomial p, and consider any monomial $m = y_{i_1 j_1} y_{i_2 j_2} \ldots y_{i_d j_d}$ of p. We will convert the monomial m into a polynomial q_m in the variables z of degree at most d. We will make

sure q_m has the property that the average, over strings in $x \in [m]^n$ matching the alphabet multiplicities specified by z, of the value $m(y_x)$ is precisely $q_m(z)$. Then since p is a linear combination of its monomials, we can simply define q to be the corresponding linear combination of q_m. Hence we need only worry about the single monomial m.

Note that we can assume all i_ℓ for $\ell \leq d$ are distinct. This is because $y_{ij}^2 = y_{ij}$, and $y_{ij}y_{ik} = 0$ whenever $j \neq k$, at least when the values of y come from a real underlying string $x \in [m]^n$ (this is because $y_{ij} = 0$ unless $x_i = j$, and we cannot have both $x_i = j$ and $x_i = k$ if $j \neq k$).

For each $z \in [n]^m$, let A_z be the set of strings in $[m]^n$ with multiplicities matching z (i.e. A_z is the set of all $x \in ([m] \cup \{\perp\})^n$ such that each $j \in [m]$ occurs exactly z_j times in x). Let e_z be the average of $m(y_x)$ over $x \in A_z$. Note that we can sample randomly from A_z by fixing a single $x \in A_z$, sampling a random permutation $\sigma \in S_n$, and then taking x_σ (the string x shuffled according to σ). Hence

$$e_z = \mathop{\mathbb{E}}_{\sigma \sim S_n}[m(y_{x_\sigma})] = \Pr_{\sigma \sim S_n}[x_{\sigma(i_\ell)} = j_\ell \ \ \forall \ell = 1, 2, \ldots, d] = \prod_{\ell=1}^{d} \Pr[x_{\sigma(i_\ell)} = j_\ell | x_{\sigma(i_k)} = j_k \ \ \forall k < \ell]$$

where x is any fixed string in A_z.

Let s_ℓ be the number of different $k \in \{1, 2, \ldots, \ell - 1\}$ such that $j_k = j_\ell$. Then we have

$$\Pr[x_{\sigma(i_\ell)} = j_\ell | x_{\sigma(i_k)} = j_k \ \ \forall k < \ell] = \frac{z_{j_\ell} - s_\ell}{n - \ell + 1},$$

because there are originally z_{j_ℓ} indices i with $x_i = j_\ell$, s_ℓ of them have already been used, and the denominator is $n - \ell + 1$ as that is the number of indices still unassigned. Hence we get

$$e_z = \prod_{\ell=1}^{d} \frac{z_{j_\ell} - s_\ell}{n - \ell + 1}.$$

Note that s_ℓ depends on the monomial m (because it depends on $j_1, j_2, \ldots, j_d$), but not on z or x or anything else. Hence it is a constant with respect to z. It follows that taking $q_m(z) = e_z$ makes q_m be a degree-d polynomial in the variables z with the desired properties. □

This symmetrization, first provided by Ambainis [Amb05], comes in very useful when analyzing the approximate degree of symmetric functions with large input alphabets. However, as you can see, this symmetrization still leaves us with a polynomial in m variables rather than in 1 variable, so usually additional tricks will be needed to get a handle on the degree.

As an example of such a trick, let us prove a lower bound for the collision problem.

Theorem 5.22. *Let f be the* Collision *problem on n bits; that is, pick $n \in \mathbb{N}$ to be even, and define f on inputs in $[n]^n$ such that $f(x) = 0$ if x is a permutation (i.e. each alphabet symbol in $[n]$ occurs exactly once in x), and $f(x) = 1$ if x is 2-to-1 (i.e. there are $n/2$ alphabet symbols in $[n]$ that each occur exactly twice in x). All other strings in $[n]^n$ are not in the domain of f.*

Then $\widetilde{\deg}(f) = \Omega(n^{1/3})$, and hence $Q(f) = \Omega(n^{1/3})$.

The proof of the collision lower bound was first given by Aaronson and Shi [AS04], though we will go through a simplified version by Kutin [Kut05].

Proof. Let p be a polynomial in variables $\{y_{ij}\}$ which computes f to error ϵ. Let q be the symmetrized polynomial in variables $\{z_j\}$. Then we have $\deg(q) \leq \deg(p)$, $q(z) \in [0, 1]$ for all $z \in [n]^n$ with $\sum_j z_j \leq n$, $q(1^n) \leq \epsilon$, and for any z that has half its coordinates equal to 0 and the other half equal to 2, we have $q(z) \geq 1 - \epsilon$.

We wish to lower bound the degree of such a polynomial q. To do so, we introduce a further symmetrization. We say a string $z \in [n]^n$ has type (A, a, B, b) if there is a set of A/a coordinates $i \in [n]$ on which $z_i = a$, as well as a disjoint set of B/b coordinates $j \in [n]$ on which $z_j = b$, and also that $z_\ell = 0$ for ℓ outside of those sets. Note that only certain tuples (A, a, B, b) have strings z associated with them; we call a tuple *valid* if $A + B \le n$ and $a, A/a, b, B/b$ are all positive integers. Note that the strings z corresponding to a valid tuple (A, a, B, b) represent strings $x \in [n]^n$ that are a-to-1 on some subset of coordinates (of size A), are b-to-1 on a disjoint subset of coordinates (of size B), and are $\perp$ on the rest of the coordinates.

Next, define the function $P(A, a, B, b)$ for valid tuples (A, a, B, b) to be the average of $q(z)$ over strings z of type matching the tuple. We claim this function is a polynomial in the variables A, a, B, b, and that $\deg(P) \le \deg(q)$. To see this, it suffices to take one monomial $m = z_{j_1} z_{j_2} \dots z_{j_d}$ of q, and show that the average of $m(z)$ over strings z of type (A, a, B, b) is a polynomial in A, a, B, b of degree at most d; then since q is a linear combination of such monomials, we can get P by taking the matching linear combination of the corresponding polynomials.

Let $S \subseteq [n]$ be the unique indices in the variables of m, so $|S| \le d$. If z is such that $z_j = 0$ for some $j \in S$, then $m(z) = 0$. Otherwise, let $S = S_A \sqcup S_B$ be a partition of S into two disjoint sets with $|S_A| \le A/a$ and $|S_B| \le B/b$. The probability that a randomly chosen z of type (A, a, B, b) will have $z_j = a$ for $a \in S_A$ and $z_j = b$ for $b \in S_B$ is

$$\frac{A/a}{n} \cdot \frac{A/a - 1}{n-1} \cdots \frac{A/a - |S_A| + 1}{n - |S_A| + 1} \cdot \frac{B/b}{n - |S_A|} \cdot \frac{B/b - 1}{n - |S_A| - 1} \cdots \frac{B/b - |S_B| + 1}{n - |S_A| - |S_B| + 1}$$

and the value of $m(z)$ for such z is $a^\alpha b^\beta$, where α is the total degree in m of the variables in S_A and β is the total degree in m of the variables in S_B; in particular, $\alpha \ge |S_A|$ and $\beta \ge |S_B|$. The contribution of the z that have a in S_A and b in S_B towards the expectation $\mathbb{E}[m(z)]$ over z of type (A, a, B, b) is the product of $a^\alpha b^\beta$ and the above probability; since $\alpha \ge |S_A|$ and $\beta \ge |S_B|$, the number of ratios A/a in this product will be less than α, so they cancel with the a^α term, and similarly for B/b. We conclude that the contribution of the (S_A, S_B) partition towards $\mathbb{E}(z)$ is a polynomial in A, b, B, b, so long as we treat $|S_A|$ and $|S_B|$ as constants. Moreover, the degree of this polynomial will be exactly $\alpha + \beta = d$. Finally, since $\mathbb{E}[m(z)]$ is a sum of such contributions, it is also a polynomial in A, a, B, b of degree at most d, which means $P(A, a, B, b)$ is a polynomial of degree at most $\deg(q)$, as desired.

We have now symmetrized down to a polynomial of four variables. We know that $P(A, 1, B, 1) \le \epsilon$ whenever $A + B = n$, that $P(A, 2, B, 2) \ge 1 - \epsilon$ whenever $A + B = n$, and that $P(A, a, B, b) \in [0, 1]$ whenever $a|A$, $b|B$, $A + B \le n$, and $a, b, A, B \in \mathbb{N}_{\ge 1}$.

In particular, note that $P(n/2, 1, n/2, 1) \le \epsilon$, $P(n/2, 2, n/2, 2) \ge 1 - \epsilon$, and $P(n/2, 1, n/2, 2) \in [0, 1]$. We split into two cases: either $P(n/2, 1, n/2, 2) \le 1/2$, or not.

In the former case, P behaves differently on $P(n/2, 1, n/2, 2)$ and $P(n/2, 2, n/2, 2)$. In this case, we consider $P(n/2, x, n/2, 2)$ as a function of x, and look at how long this function stays reasonably bounded near $[0, 1]$ (we want to use the fact that if a polynomial moves quickly from 1 to 2 but then stays bounded for a long time, it must have high degree). Specifically, let r be the smallest integer such that $P(n/2, r, n/2, 2) \notin [-1, 2]$. Then by Corollary 5.14, we have $\deg(P) = \Omega(\sqrt{r})$ by looking at the single-variate polynomial $P(n/2, x, n/2, 2)$ on the range $x \in [1, r - 1]$, where it is bounded on integer points. This gives a good lower bound on $\deg(P)$ if r is large.

What if r is small? In this case, we look at the polynomial $P(2rx, r, n - 2rx, 2)$. Since n is even, we have $n - 2rx$ be even whenever x is an integer, so the tuple $(2rx, r, n - 2rx, 2)$ is valid for all positive integers $x < n/2r$; hence $P(2rx, r, n - 2rx, 2) \in [0, 1]$ for all positive integers $x \in \{1, 2, \dots, \lceil n/2r \rceil - 1\}$. However, we know that when $x = n/4r$, the above function lies outside

the range $[-1, 2]$, which means it goes out of $[0, 1]$ by at least 1. This means it must have derivative at least 1 at a point near $n/4r$. Paturi's theorem therefore tells us that $\deg(P) = \Omega(n/r)$. Combining with the previous bound on $\deg(P)$, we get $\deg(P) = \Omega(\sqrt{r} + n/r)$. This is minimized at $r = n^{2/3}$, so regardless of the value of r we must have $\deg(P) = \Omega(n^{1/3})$.

The case where $P(n/2, 1, n/2, 2) > 1/2$ is similar, except that we define r as the smallest integer such that $P(n/2, 1, n/2, r) \notin [-1, 2]$, and then examine the degree of $P(n/2, 1, n/2, x)$ and of $P(n - rx, 1, rx, r)$. This gives the same $\Omega(n^{1/3})$ lower bound. □

Note that the collision lower bound cannot be shown using the positive adversary method, due to the property testing barrier. There is a known proof of the collision lower bound using the negative-weight adversary method [BR13].

Finally, let's consider the element distinctness problem ED. This function takes inputs strings in $[m]^n$ (with $m \geq n$), outputs 0 if the characters of the input string are all distinct, and outputs 1 otherwise. The certificate complexity barrier says that the positive-weight adversary cannot prove a lower bound better than $\Omega(\sqrt{n})$. We will now show that $\mathrm{Q}(\mathrm{ED}_n) = \Omega(n^{2/3})$, at least when m is large enough.

Theorem 5.23. $\mathrm{Q}(\mathrm{ED}_n) = \Omega(n^{2/3})$.

Proof. Suppose there was a fast quantum algorithm for computing ED_n. Consider the following algorithm for computing collision on strings of length $m = n^2/100$: first, sample n random positions; second, run the algorithm for element distinction on those n bits. Note that if the input to collision contains $m/2$ alphabet symbols appearing twice each, then with high probability (probability at least 0.9) the n random positions will not all be distinct, by the same calculation as the birthday paradox. Hence, with probability bounded away from $1/2$, the ED algorithm applied to those n random bits will give the correct output to the collision problem on $n^2/100$ bits. Since the latter requires $\Omega((n^2)^{1/3})$ quantum queries, it must be the case that ED requires $\Omega(n^{2/3})$ quantum queries. □

The above reduction from collision to element distinctness also works for polynomials: we have $\widetilde{\deg}(\mathrm{ED}_n) = \Omega(n^{2/3})$. The trick is similar, except that instead of taking a *random* subset of n elements out of $n^2/100$, we take an *average* over all such subsets (i.e. we construct a polynomial p_S which checks whether the elements in positions indexed by $S \subseteq [m]$ are distinct, and then average p_S over all sets S of size n).

Finally, while the above arguments only showed the element distinctness lower bound when the alphabet size m is at least $n^2/100$, it turns out that the lower bound still holds when $m = n$. Indeed, Ambainis [Amb05] showed the following general theorem.

Theorem 5.24. *Let f be a (possibly partial) function with $\mathrm{Dom}(f) \subseteq [m]^n$ and with Boolean outputs, where $m \geq n$. Suppose that f is symmetric under both permuting the characters of the input string (an S_n symmetry) and also under permuting the names of the alphabet symbols (an S_m symmetry). Let f' be the restriction of f to the promise $P = \{x \in \mathrm{Dom}(f) : x \in [n]^n\}$. Then $\widetilde{\deg}(f) = \widetilde{\deg}(f')$.*

Since element distinctness is symmetric under both permuting the input string and also renaming the alphabet, it follows that the approximate degree of ED with alphabet of size $m = n^2/100$ is the same as the approximate degree of ED with alphabet of size n, so both are at least $\Omega(n^{2/3})$. Finally,

we remark that this is tight, as there is a quantum algorithm using $O(n^{2/3})$ queries which computes element distinctness (and hence also a quantum algorithm using $O(n^{1/3})$ queries for collision).

References

[Amb05] Andris Ambainis. "Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range". In: *Theory of Computing* 1.1 (2005) (pp. 9, 11).

[AS04] Scott Aaronson and Yaoyun Shi. "Quantum Lower Bounds for the Collision and the Element Distinctness Problems". In: *Journal of the ACM* 51.4 (2004). DOI: 10.1145/1008731.1008735. URL: http://doi.acm.org/10.1145/1008731.1008735 (p. 9).

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald De Wolf. "Quantum lower bounds by polynomials". In: *Journal of the ACM (JACM)* 48.4 (2001). DOI: 10.1145/502090.502097. eprint: quant-ph/9802049 (p. 7).

[BR13] Aleksandrs Belovs and Ansis Rosmanis. "Adversary lower bounds for the collision and the set equality problems". In: *arXiv preprint arXiv:1310.5185* (2013) (p. 11).

[EZ64] Hartmut Ehlich and Karl Zeller. "Schwankung von Polynomen zwischen Gitterpunkten". In: *Mathematische Zeitschrift* 86.1 (1964) (p. 6).

[Kut05] Samuel Kutin. "Quantum lower bound for the collision problem with small range". In: *Theory of Computing* 1.1 (2005) (p. 9).

[Pat92] Ramamohan Paturi. "On the degree of polynomials that approximate symmetric Boolean functions". In: *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. 1992 (p. 7).

[RC66] Theodore J Rivlin and Elliott Ward Cheney. "A comparison of uniform approximations on an interval and a finite subset thereof". In: *SIAM Journal on numerical Analysis* 3.2 (1966) (p. 6).