Lecture 5: Approximate Degree

Rajat Mittal

IIT Kanpur

I would like to thank Aditya Ranjan for making a preliminary version of these lecture notes.

1 Lower bounding randomized query complexity

We first recall a lower bound for the deterministic query complexity of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ using the *degree of* f.

Proof sketch:. Consider the decision tree for f having depth $\mathrm{D}(f)$. Corresponding to each computation path (i.e. a path from the root to some output leaf) in the tree, we can construct an indicator polynomial. That means the indicator polynomial is one whenever the input is part of the leaf and 0 otherwise.

Exercise 1. Can you construct such an indicator polynomial depending upon the path of the deterministic decision tree?

The indicator polynomial corresponding to any computation path is the product of either x_i or $(1-x_i)$, based on whether we set its value to be 1 or 0 respectively in the path (for all the variables x_i in the path). It can be seen that the degree of each of these indicator polynomials is equal to the depth of the leaf, and hence is at most $\mathrm{D}(f)$.

The (unique) polynomial corresponding to f is the sum of these indicator polynomials, all having degree at most $\mathrm{D}(f)$. Hence, the degree of their sum, $deg(f) \leq \mathrm{D}(f)$.

Exercise 2. Prove that the polynomial constructed as above represents the function.

□

Can degree be a lower bound on the randomized query complexity of a function? It turns out that this is not possible. Saks and Wigderson [2] showed that there are functions which separate the deterministic query complexity and the randomized query complexity. One such example is the *iterated* composition of Majority function.

Exercise 3. Show that the degree of iterated Majority function is full.

Using that result that degree multiplies under composition (you proved it earlier), the degree of composition of NAND function (however many times you compose) is also full. Saks and Wigderson [2]gave a randomized algorithm for this iterated composition, where the query complexity is not full. We have seen this algorithm in the previous lecture.

1.1 Modifying degree to get a lower bound

How should we extend such a proof for the randomized query complexity? Suppose R is a randomized decision tree (randomized query algorithm) for f.

Exercise 4. Can you say that the polynomial constructed in a similar manner for any decision tree in support of R will represent f?

How can we define a polynomial from the randomized algorithm? Let R be a randomized query algorithm for f. Remember that it can be seen as a probability distribution over some deterministic decision trees $D_1, D_2, \cdots, D_k$. Say the corresponding probabilities are $p_1, p_2, \cdots, p_k$.

From the proof sketch before, every decision tree D_i in support of R gives rise to a polynomial $P_i(x)$. What can we say about the polynomial $P = \sum_i p_i P_i$? Notice that P gives us the probability of acceptance for any input. The condition for R computing f can be rewritten as:

$$|f(x) - P(x)| \leq \frac{1}{3} \text{ for all inputs } x. \tag{1}$$

Exercise 5. Convince yourself that the above statement is true.

Let f be a Boolean function. Now, say we want our randomized algorithm R to compute f. We must have $\Pr(R(x) = f(x)) \geq \frac{2}{3}$ for every input x. Thus, for each input x:

$$f(x) = 0 \Longrightarrow \text{Probability of Acceptance} \leq \frac{1}{3} \qquad \text{(Acceptance is failure here)}$$

$$f(x) = 1 \Longrightarrow \text{Probability of Acceptance} \geq \frac{2}{3} \qquad \text{(Acceptance is success here)}$$

Thus, if we have a randomized algorithm R which computes f, then there is a corresponding polynomial P which satisfies Equation 1 (and also the constraint $0 \leq P(x) \leq 1 \, \forall \, x$, as P represents a probability). Notice that the degree of P is at most the maximum depth of D_i's and hence is bounded by the randomized query complexity of R.

We need to define a variant of degree, which will be a lower bound on the randomized query complexity.

Exercise 6. Can you already think of a variant?

This also means that for a given Boolean function f, let us look at *all* the polynomials P which satisfy (1). Then, the lowest degree among all these polynomials will give us a lower bound on the query complexity of R. Since this is true for any R which computes f, the lowest degree among all such polynomials is a lower bound on the randomized decision tree complexity

Note 1. Here we have removed the constraint $\forall x : 0 \leq P(x) \leq 1$. We already had the constraint $\forall x : -\frac{1}{3} \leq P(x) \leq \frac{4}{3}$ from (1) itself, and this additional constraint does not really change the lower bound for complexities much (just by constant). The idea is to first scale the polynomial, this will increase the error; then use repetition and majority to reduce error (see appendix of [4]).

1.2 Approximate degree of a Boolean function

We say that a polynomial P **approximates** f, if

$$|f(x) - P(x)| \leq \frac{1}{3} \text{ for all inputs } x. \tag{2}$$

Note 2. This is just one notion of approximation, there are many more such notions in mathematics. For example, another notion is that $P(x) = f(x)$ for a high fraction of inputs x. But this notion is not useful for us, as for e.g. if f is OR, then the constant (zero degree) polynomial 1 approximates OR (as it is equal to OR on almost all inputs).

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, its approximate degree, $\widetilde{\deg}_{1/3}(f)$ is the *minimum* possible degree of a polynomial p which approximates f.

$$\widetilde{\deg}_{1/3}(f) = \min_{\substack{\text{polynomial } p: \\ |p(x) - f(x)| \leq 1/3 \ \forall x}} \deg(p). \tag{3}$$

It is not difficult to see that $\widetilde{\deg}_{1/3}(f) \leq \deg(f)$, as the polynomial representation for f trivially approximates f. In fact, if $\epsilon \geq \delta$, then $\widetilde{\deg}_{\epsilon}(f) \leq \widetilde{\deg}_{\delta}(f)$. As mentioned before, the error reduction can be done by repeating and taking majority [4, See Appendix].

What can we say about the approximate degree of the composition of two Boolean functions. It is known (but not straightforward, why?) that the approximate degree of $f \cdot g$ is bounded by the multiplication of the approximate degree of f and approximate degree of g [3]. Though, whether there is an equality (asymptotically), is a big open question.

Lower bounds using approximate degree Remember that $R(f)$ refers to the smallest number of queries used by any randomized algorithm which computes f with bounded error.

From previous discussion, if there is a randomized algorithm for f which uses t queries, then $\exists$ an approximating polynomial for f with degree at most t. Now consider the optimal randomized algorithm which computes f, it uses $R(f)$ queries, so there is an approximating polynomial for f with degree $\leq \cdot R(f)$. The degree of this approximating polynomial is at least the approximate degree of f.

We get a lower bound on randomized query complexity using the approximate degree of f:

$$R(f) \geq \widetilde{\deg}_{1/3}(f). \tag{4}$$

We can write the same equation for quantum query complexity with a constant on the right hand side (what is that constant?). Notice how this is analogous to the statement $\mathrm{D}(f) \geq \deg(f)$.

1.3 Lower bound using approximate degree on quantum query complexity

Though we have not defined quantum query complexity formally, let us state a similar statement (without proof) for it as well (this will help us achieve lower bounds on the quantum query complexity):

Theorem 1. *The probability of* acceptance *of a quantum query algorithm on input $x = x_1x_2\dots x_n$ is a polynomial in $x_1, x_2, \dots, x_n$ having degree at most* twice *the number of queries.*

Note 3. Acceptance here refers to the event of the quantum algorithm giving 1 as the output (this is analogous to the definition of acceptance of a language in the Theory of Computation). There is no mention of any Boolean function f here, the theorem is independent of that.

Notice that this statement gives some sort of a lower bound on the number of queries (i.e. the quantum query complexity). We follow the same approach as in the case of randomized query complexity.

Let f be a Boolean function. Now, say we want our Quantum algorithm Q to compute f. We must have $\Pr(\mathrm{Q}(x) = f(x)) \geqslant \frac{2}{3}$ for every input x. Thus, we can say that for each input x:

$$f(x) = 0 \Longrightarrow \text{Probability of Acceptance} \leqslant \frac{1}{3} \qquad \text{(Acceptance is failure here)}$$

$$f(x) = 1 \Longrightarrow \text{Probability of Acceptance} \geqslant \frac{2}{3} \qquad \text{(Acceptance is success here)}$$

As specified in Theorem 1, let the probability of acceptance of Q on input x be a polynomial $P(x)$. The condition for Q computing f can be rewritten as:

$$|f(x) - P(x)| \leqslant \frac{1}{3} \text{ for all inputs } x \tag{5}$$

Thus, if we have a quantum algorithm Q which computes f, then there is a corresponding polynomial P which satisfies (5) (and also the constraint $0 \leq P(x) \leq 1 \,\forall\, x$, as P represents a probability) such that the number of queries taken by Q is at least $\frac{1}{2}\deg(P)$.

Exercise 7. Show that

$$Q(f) \geq \frac{1}{2}\widetilde{\deg}_{1/3}(f). \tag{6}$$

2 Techniques for getting Approximate Degrees

We have converted the problem of computing lower bound on randomized query complexity of f to giving the lower bound on approximate degree of f. Below, we will discuss some ways to bound the approximate degree of a symmetric Boolean function. The standard trick in this case is known as *Minsky-Papert symmetrization.*

2.1 Symmetrization Trick by Minsky & Papert

One way of lower bounding the degree of a polynomial can be to argue about its number of roots. But here, our polynomials are multivariate polynomials, not univariate ones, so we cannot use the number of roots directly. We had seen before that a symmetric function/polynomial (in $x \in \{0,1\}^n$) can be thought of as a *univariate* polynomial (in $|x| \in \{0,1,\ldots,n\}$) having the *same* degree as the original polynomial.

Though f here is symmetric, our polynomial p approximating f need not be a symmetric polynomial. Thus, we need to use a *symmetrization step* to convert p into a symmetric polynomial.

Theorem 2. *Let p be a polynomial (and a function $p : \{0,1\}^n \to \mathbb{R}$) and let p_{symm} be the symmetric polynomial*

$$p_{\text{symm}}(x) = \frac{1}{n!}\sum_{\sigma \in S_n} p(\sigma(x)) \tag{7}$$

If p approximates a symmetric Boolean function f, p_{symm} approximates f too.

Proof. As f is symmetric, we can write f as

$$f(x) = \frac{1}{n!}\sum_{\sigma \in S_n} f(\sigma(x)) \tag{8}$$

Thus, we have

$$|p_{\text{symm}}(x) - f(x)| = \frac{1}{n!}\left|\sum_{\sigma \in S_n} p(\sigma(x)) - f(\sigma(x))\right| \leq \frac{1}{n!}\sum_{\sigma \in S_n} |p(\sigma(x)) - f(\sigma(x))| \tag{9}$$

Here, we have used the triangle inequality. The quantity inside the absolute value can be bounded by 1/3 (why?), so p_{symm} approximates f if p approximates f. $\square$

Note that p_{symm} has the same degree as that of p (this can be proven). Since p_{symm} is symmetric, it can further be converted into a univariate polynomial (in $|x|$) which has the same degree as p_{symm}, i.e. same degree as p.

Corollary 1. *If f is symmetric and $\widetilde{\deg}_{1/3}(f) = d$, then there exists a univariate polynomial $P(w)$ of degree d such that*

$$|P(w) - f(w)| \leq \frac{1}{3} \qquad \forall\ w \in \{0,1,\ldots,n\}. \tag{10}$$

Here, $f(w)$ is the value of f at inputs of Hamming weight w.

2.2 Approximate Degree of PARITY

Our task is to find the approximate degree of PARITY. In other words, we need to find the minimum degree among *all* polynomials which approximate PARITY. As usual, this lower bounding seems to be a difficult job.

Let us consider an optimal degree univariate polynomial P for PARITY. Note that,

$$\text{PARITY}(w) = \begin{cases} 0, & \text{if } w \text{ is even} \\ 1, & \text{if } w \text{ is odd} \end{cases}$$

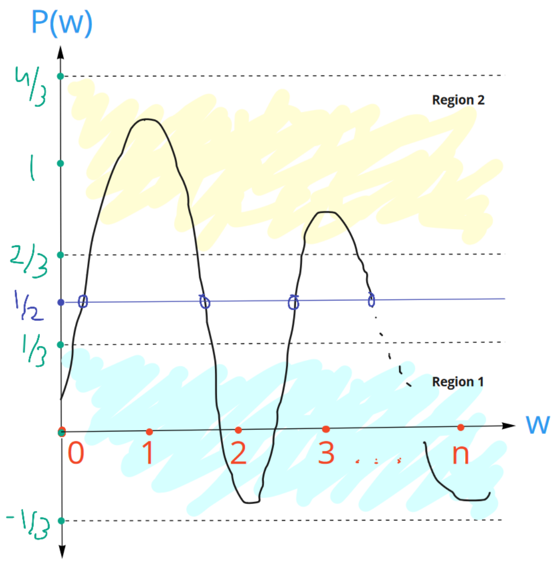

Fig. 1. Rough sketch of P for PARITY

This means that $-\frac{1}{3} \le P(w) \le \frac{1}{3}$ if w is even, and $\frac{2}{3} \le P(w) \le \frac{4}{3}$ if w is odd. So P roughly looks like this:

So $P(w)$ goes up and down from Region 1 to Region 2 on each successive step. The intuition is, since P keeps going up and down lot of times, it should have a high degree. To be precise, consider the line $y = \frac{1}{2}$ (any value between $\frac{1}{3}$ and $\frac{2}{3}$ works). The line intersects the polynomial many times.

More formally, it can be seen that $\forall 1 \le k \le n$, $P(k-1)$ and $P(k)$ are in different regions (among region 1 & 2), i.e. one of $P(k-1)$ and $P(k)$ lies in $\left[-\frac{1}{3}, \frac{1}{3}\right]$, and the other one lies in $\left[\frac{2}{3}, \frac{4}{3}\right]$. Since polynomials are continuous, we can say that $\exists t \in (k-1, k)$ such that $P(t) = \frac{1}{2}$, and we can find such a $t \in (k-1, k)$ for every $1 \le k \le n$.

This means that our polynomial P has to intersect $y = \frac{1}{2}$ at least n times, i.e., $P(w) - \frac{1}{2}$ has at least n roots. Hence, $P(w)$ has at least n roots ($\deg(P) \ge n$).

The P we chose came from the optimal polynomial approximating PARITY, so $\widetilde{\deg}_{1/3}(\text{PARITY}) \ge n$. We already know that $\widetilde{\deg}_{1/3}(\text{PARITY}) \le \deg(\text{PARITY}) = n$, we get $\widetilde{\deg}_{1/3}(\text{PARITY}) = n$.

2.3 Approximate Degree of OR

We saw how doing the symmetrization trick for PARITY, we got a univariate polynomial having degree equal to the approximate degree of PARITY. This allowed us to lower bound the degree of this univariate polynomial.

Since OR is also a symmetric Boolean function, we will use the same strategy for OR.

As we did before, pick a univariate polynomial approximating OR having optimal degree, say P. Now,

$$\text{OR}(w) = \begin{cases} 0, & \text{if } w = 0 \\ 1, & \text{if } w = 1, 2, \ldots, n \end{cases}$$

This means that $P(w)$ remains in Region 1 ($\left[-\frac{1}{3}, \frac{1}{3}\right]$) at $w = 0$, and then shifts to Region 2 ($\left[\frac{2}{3}, \frac{4}{3}\right]$) at $w = 1, 2, \ldots n$. Thus, there is a sharp (i.e. non-zero) derivative of $P(w)$ in $(0, 1)$, and after that P remains bounded, at least on the integer points. We can then try the following inequality to lower bound the degree of P:

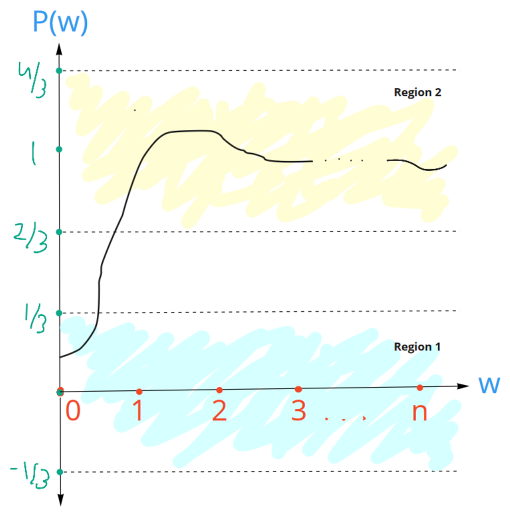

Fig. 2. Rough sketch of P for OR

Markov Brothers' Inequality

Theorem 3. *For a polynomial P, if $b_1 \leq P(x) \leq b_2$ in the interval $a_1 \leq x \leq a_2$, then*

$$|P'(x)| \leq \frac{d^2(b_2 - b_1)}{a_2 - a_1} \qquad \forall\ x \in [a_1, a_2] \tag{11}$$

where d is the degree of P.

Using mean value theorem, there is some $x \in (0, 1)$ such that $P'(x) = \frac{P(1)-P(0)}{1-0} \geq \frac{1}{3}$. One can then think of using Markov Brothers' Inequality for the interval $x \in [0, n]$ in which $P(x) \in [-\frac{1}{3}, \frac{4}{3}]$, which would indeed give us that $\deg(P) \geq c\sqrt{n}$ for some constant $c > 0$.

Exercise 8. What is the mistake in this argument?

We cannot use Markov Brothers' Inequality directly here: even though $P(x)$ is bounded (by a constant) on the integer points in $[0, n]$, there is no guarantee that it will remain bounded *by a constant* in the whole interval $[0, n]$.

Note 4. You might think that P is bounded on any interval by its maximum value and hence we can use the inequality for the interval $[0, n]$. If this maximum value is not a constant (e.g. n instead of a constant), then we get an asymptotically smaller bound than our required lower bound of $\sqrt{n}$ for $\deg(P)$.

However, there is another result (which is derived from Markov Brothers' Inequality itself) that lower bounds the degree of a polynomial which is bounded on just integer points:

Degree Lower Bound for Polynomial bounded at integer points (Ehlich & Zeller)

Theorem 4. *If P is a polynomial such that $b_1 \leq P(i) \leq b_2\ \forall\ i \in \{0, 1, \ldots, n\}$, and $\exists 0 \leq x \leq n$ such that $|P'(x)| \geq c$, then*

$$\deg(P) \geq \sqrt{\frac{cn}{c + b_2 - b_1}} \tag{12}$$

Take $c = \frac{1}{3}, b_1 = -\frac{1}{3}, b_2 = \frac{4}{3}$ and apply this result directly to get $\deg(P) = \Omega(\sqrt{n})$. We proved that

$$\widetilde{\deg}_{1/3}(\mathrm{OR}) = \Omega(\sqrt{n}).$$

It is known that this bound is tight for approximate degree. For symmetric functions, approximate degree is known asymptotically [1].

Unfortunately, this bound *doesn't* turn out to be *tight* for R. In the next section, we will see another technique to get a lower bound on R. It will even work for non-symmetric f.

2.4 Dual witness technique

We have seen lower bounding techniques for symmetric functions. A prominent technique to give lower bound on approximate degree more generally is called *dual witness technique*.

A $\phi : \{0,1\}^n \to \mathbb{R}$ is called a dual witness for $f : \{0,1\}^n \to \mathbb{R}$ of degree d iff

- norm condition: $\|\phi\|_1 = \frac{1}{2^n}\sum_x |\phi(x)| = 1$,
- correlation condition: $\langle\phi|f\rangle > 1/3$,
- and pure high degree: ϕ has no monomial of degree less than equal to d

are satisfied. We will show that such a ϕ implies a d lower bound on the approximate degree of f.

Proof. The proof comes from analyzing $\langle\phi|f\rangle$ in another way. Suppose there is a polynomial p of degree d approximating f.

$$\langle\phi|f\rangle = \langle\phi|f-p\rangle \le \|\phi\|_1 \max_x |(f-p)(x)| \le 1/3.$$

Exercise 9. Make sure that you can prove each of the inequalities using the properties of p and ϕ.

This gives a contradiction with the correlation condition. So no polynomial of degree less than equal to d can approximate f. □

The above proof does not give much intuition about how would someone come up with such set of conditions. They arrive quite naturally by creating a linear program to optimize the error for all polynomials of degree less than equal to d and taking its *dual*.

We give another justification (not complete). Suppose you want to show that a vector v can't be close to any vector in a subspace S. If you can show that v is orthogonal to S (all vectors of S), then you are done. Though that need not be the case always. Instead, it is enough to give a unit vector v' which is orthogonal to S but close to v.

Dual witness ϕ is like v', except that v' is for L_2 distance. In case of dual witness, we need to optimize L_∞ and hence the dual norm (1-norm) shows up.

3 Assignment

Exercise 10. Read about duality theory of linear programming.

Exercise 11. Give an example of a function where R is not same as its approximate degree.

Exercise 12. Prove that if f is a non-constant symmetric Boolean function, then $\widetilde{\deg}_{1/3}(f) = \Omega(\sqrt{n})$.

Exercise 13. For the composition of approximate degree, what is the natural choice for the approximating polynomial of $f \circ g$? Why does it fail?

References

1. R. Paturi. On the degree of polynomials that approximate symmetric boolean functions. *STOC 1992*, 1992.
2. M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. *27th Annual Symposium on Foundations of Computer Science (sfcs 1986), Toronto, ON, Canada, 1986*, pages 29–38, 1986.
3. A. Sherstov. Making polynomials robust to noise. *STOC 2012*, 2012.
4. Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 551–560, 2014.