6.895 Randomness and Computation February 8, 2006

Lecture 1

Lecturer: Ronitt Rubinfeld *Scribe: David Sontag*

1 Polynomial Identity Testing

Given two polynomials P and Q, we want to know whether $P \equiv Q$. We begin with some definitions.

Definition 1. *The* total degree *of a multivariate polynomial P is the maximum degree of any term in P, where the degree of a particular term is the sum of the variable exponents.*

For example, the total degree of $x_1^2 x_2^3 x_3 + x_1^6 x_2 x_4$ is 8, as opposed to the maximum degree of any variable in the terms, which is 6. For the remainder of this lecture we use degree to mean total degree.

An easy way of checking whether two polynomials are equal is to expand them, then compare term by term. However, expanding the polynomials (e.g. $(x+y)^{29}z - (x-y)^{20}z^2$) could result in exponentially many terms. Instead, we give a randomized algorithm that can check for polynomial equality in polynomial time. The algorithm that we give works for any representation of the polynomial; we just assume that we have a black box which evaluates $P(x)$ and $Q(x)$. Note that $P \equiv Q$ iff $P - Q \equiv 0$, and so we instead give an algorithm to check if a polynomial is identically zero, i.e. $\forall x, P(x) \equiv 0$.

1.1 Univariate polynomials

We start by giving an algorithm for polynomial identity testing on univariate polynomials, and will later show how to extend this algorithm to multivariate polynomials. Let d be the degree of the univariate polynomial P. Assume that this polynomial is over a field F, e.g. $\mathbb{Z}_p$ (the field of integers modulo prime p). If $P \not\equiv 0$, then P has at most d roots. One approach would be to pick any $d+1$ points and evaluate P on them. If $P \not\equiv 0$, at least one of these $d+1$ points must be a non-root. As we show later, one must be careful with generalizing this approach to multivariate polynomials. Instead, we give a randomized algorithm, independently discovered in the late 1970's by DeMillo and Lipton, Schwarz, and Zippel. The general idea is to evaluate the polynomial at random points selected from a sufficiently large range. If any of these values are non-zero, then we report that the polynomial is not identically zero.

Definition 2. *The notation $x \in_R S$ means that x is sampled uniformly at random from the set S. Likewise, if $\mathcal{U}$ is a distribution, then $x \in_R \mathcal{U}$ means that x is sampled from the distribution $\mathcal{U}$.*

Let S be an arbitrary subset of F such that $|S| \geq 2d$. Since P has at most d roots in F, at most d of the $|S|$ points will evaluate to zero, and $|S| - d \geq d$ points will be non-roots (will not evaluate to zero). Thus, if $P \not\equiv 0$, then $x \in_R S$ will give a non-root with probability at least $\frac{d}{2d} = 1/2$. To make the probability of error small, repeat.

1.2 Multivariate polynomials

Now assume that the polynomial P is over n variables $x_1, \ldots, x_n$, and let d be its total degree. Testing polynomial identity for multivariate polynomials is harder because there are many more roots. For example, $P(x_1, x_2) = x_1 x_2$ has infinitely many roots in $\mathbb{R}$, and p roots in $\mathbb{Z}_p$. Also, as we mentioned earlier, expanding the polynomials into monomials could result in $\binom{n}{d}$ terms, so comparing them term by term would take time exponential in d.

Algorithm:

1. Let $S \subseteq F$, such that $|S| \geq 2d$ (S chosen arbitrarily.).
2. Choose $x_1, \ldots, x_n \in_R S$.
3. If $P(x_1, \ldots, x_n) = 0$, output "$\equiv 0$", else "$\not\equiv 0$".

Claim 3. *For $x_1, \ldots, x_n \in_R S$, $P \not\equiv 0 \Rightarrow \Pr(P(x_1, \ldots, x_n) = 0) \leq \frac{d}{|S|}$.*

While we won't prove the above claim, it can be shown via induction on n, the number of variables in the polynomial. We already did the base case (univariate polynomials). This claim tells us that not all of the elements of S^n can be roots of P.

1.3 Comments

1. The point $(x_1, \ldots x_n)$ must be sampled from the $n-$dimensional cube S^n (i.e. uniformly). Arbitrary distributions, even over large support, will not in general work because a multivariate polynomial can have lots of zeros.

2. This algorithm can be generalized to polynomials over trigonometric and geometric functions.

3. Attempts to de-randomize this algorithm have not yet resulted in a polynomial time algorithm. In fact, there is a result showing that if there exists a polynomial time (in n, the number of variables, and d, the degree) algorithm for polynomial identity testing, then either NEXP $\not\subset$ P/poly, or Permanent is not computable by polynomial-size arithmetic circuits [1]. Thus, randomness seems to be a necessary ingredient.

2 Applications of Polynomial Identity Testing

2.1 Man on the moon problem

Being the lazy MIT student you are, you wait until the last moment to hand in your homework, by which time your professor got bored of waiting and decided to go to the moon. You want to prove to your professor that your solution (string w) is identical to his solution (string z). Unfortunately, sending information to the moon is expensive, and you're an impoverished student. Thus, you would like to find a way to prove to your professor that you have the right solution (i.e. $w = z$) by sending fewer than w bits.

You can view w and z as univariate polynomials P_w and P_z, where the i'th bit of the string is the coefficient of x^i in the polynomial (of degree $N+1$, where N is the string length). The professor will choose a value $x \in_R S$, where $S \subseteq F$ and $|S| \geq 2N$, and request the student to send him $P_w(x)$. If $w \neq z$, then $P_w(x) \neq P_z(x)$ with probability at least 1/2. Assuming the field F is chosen reasonably (e.g. $\mathbb{Z}_p$ for p only slightly bigger than $|S|$), this will have communication cost only $O(log(N))$.

Q: Why not just choose random indices of w to transmit? A: If w and z differ in just 1 bit, then with O(log(N)) transmissions there's only a log(N)/N chance of realizing that $w \neq z$ (compared to 1/2 with this algorithm)!

2.2 Bipartite matching

Definition 4. *A* bipartite graph *is a graph $G = (V, E)$ s.t. $\exists L, R$ where $L \cup R = V$, s.t. $\forall v_i, v_j \in L, (v_i, v_j) \notin E$, and $\forall v_i, v_j \in R, (v_i, v_j) \notin E$. A* matching *is a subset of the edges $M \subseteq E$ such that no two edges share a vertex. A* perfect matching *is a matching where each vertex is incident to some edge.*

We want to ask the question, given a bipartite graph G, does a perfect matching exist? This could be answered deterministically using network flow, but here we give a randomized algorithm, based on polynomial identity testing. The algorithm begins by defining the *Edmonds matrix*[1] $A_G = [a_{ij}]$, where $a_{ij} = X_{ij}$ if $(i, j) \in E$, and $a_{ij} = 0$ otherwise. Note that this is a matrix of *variables*, not numbers. The determinant of this matrix is a polynomial.

Claim 5. *G has a perfect matching iff $det(A_G) \neq 0$.*

[1] In class, we called it the *Tutte matrix*. It would also make sense to call it the *Frobenius and Koenig matrix.*

Proof

$$det(A_G) = \sum_{\sigma,\text{ all permutations}} sign(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)} \quad (1)$$

Each permutation σ corresponds to a possible matching. The product $\prod_{i=1}^{n} a_{i,\sigma(i)}$ will be non-zero iff $\forall i = 1 \dots n$, each edge $(v_i, v_{\sigma(i)}) \in E$, in which case σ corresponds to a perfect matching in G. The polynomial $det(A_G)$ is non-zero iff any term in the determinant is non-zero; no cancellations can occur because every term differs in at least one variable. ■

The determinant $det(A_G)$ is a polynomial of degree at most n, where $n = |V|$. To test if it is identically zero, just use an appropriately large field and apply the Schwarz algorithm.

Claim 6. *$det(A_G)$ is a multivariate polynomial on n^2 variables with total degree $\leq n$ and $n!$ terms.*

Using the polynomial identity testing algorithm described earlier in lecture, we pick integer random values for the X_{ij}'s in the range $1 \dots 2n$, and then compute the determinant of the resulting integer matrix. This gives a $O(n^\alpha)$ time algorithm, where α is the exponent of the best known matrix multiplication algorithm (currently 2.376). See [2] for the use of this technique to actually find a matching.

3 Existence of a 2-coloring

In this section we introduce Erdös *probabilistic method.* The probabilistic method is a technique to prove the existence of an object by showing that the probability that it exists is greater than zero. We begin by a simple example, demonstrating how it can be applied to the problem of showing the existence of a 2-coloring in sets.

Given $S_1, \dots S_m \subseteq S$, $\forall i |S_i| = l$, is it possible to color each element of S such that none of the S_i are monochromatic (e.g. all red or all blue)? In general, you can't know without explicitly checking all of the colorings. However, we can show that the answer is always "yes" when $m < 2^{l-1}$.

Theorem 7. *If $m < 2^{l-1}$, there always exists a 2-coloring.*

Proof Randomly assign a color (blue or red) to each element of S.

$$\begin{aligned}\forall i, \Pr(S_i \text{ is monochromatic}) &= \Pr(S_i \text{ is all blue}) + \Pr(S_i \text{ is all red}) \\ &= \frac{1}{2^l} + \frac{1}{2^l} = \frac{1}{2^{l-1}}.\end{aligned}$$

$$\begin{aligned}\Pr(\text{any } S_i \text{ is monochromatic}) &\leq \sum_i \Pr(S_i \text{ is monochromatic}) \\ &= m\frac{1}{2^l} < 1.\end{aligned}$$

Here we made use of the Union-Bound theorem, which says that $\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$ for any two events A and B. Since the probability that any of the S_i is monochromatic is less than 1, the probability that none of the S_i are monochromatic is greater than 0, showing the existence of such a coloring. ■

References

[1] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 355–364, New York, NY, USA, 2003. ACM Press.

[2] Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *FOCS*, pages 248–255, 2004.