CSE 521: Design and Analysis of Algorithms I **Fall 2020**

Lecture 9: Schwartz-Zippel Lemma, Perfect Matching

Lecturer: Shayan Oveis Gharan *10/28/2020*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

In this lecture, we will discuss polynomial identity testing and its applications.

1.1 Polynomial Identity Testing and Schwartz-Zippel Lemma

Given two polynomials $p(x_1, \ldots, x_n)$ and $q(x_1, \ldots, x_n)$, we'd like to find out whether $p \equiv q$, i.e. whether they produce identical outputs given any input x, if

$$p(x_1, \ldots, x_n) - q(x_1, \ldots, x_n) = 0$$

is true for all $x \in \mathbb{R}^n$.

Definition 1.1. A **monomial** is a function defined as the product of powers of variables with nonnegative exponents. A constant coefficient may be present. The **degree** of a monomial is the sum of all the exponents involved.

Definition 1.2. A **polynomial** is a function defined as the sum of monomials. In a polynomial, each component monomial is also referred to as a **term**. The **degree** of a polynomial is the largest degree of any monomial with nonzero coefficient.

Example 1.3. Some examples of polynomials:

- $2x + 3xy^2$ is a polynomial of two variables with degree 3. It has two monomials x with coefficient 2 and xy^2 with coefficient 3.
- $0x^3 + 4x^2 + 3x - 1$ is a polynomial of a single variable with degree 2.
- The determinant of a matrix $A = [A_{i,j}]_{n \times n}$ is a polynomial of n^2 variables with degree n:

$$\det(A) = \sum_{\sigma:[n]\to[n]} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)},$$

  where σ is a permutation defined on $[n] = \{1, \ldots, n\}$ and $\operatorname{sgn}(\sigma)$ is either $+1$ or -1 depending on the nature of the permutation σ.

One naive way to test the identity of p and q is to simply make the list of all monomials for each polynomial and compare the resulting lists. Unfortunately, it is often impratical to do so. For instance, the determinant function consists of $n!$ terms, so listing them would cost us exponential time. So we assume that we are given **oracle access** to polynomials p, q, i.e., that we can query the oracle for a specific input x and it outputs $p(x), q(x)$. For instance, if we assign specific values to all terms in matrix $A_{i,j} = x_{i,j}$ for all i, j, we can compute the determinant in $O(n^3)$ time using the LU decomposition. The determinant example inspires the following formulation:

Definition 1.4 (Polynomial Identity Testing)**.** Given a polynomial p defined over a a set of variables $x_1, \ldots, x_n$, we'd like to determine whether $p \equiv 0$. We are only given **oracle access**: no individual monomial of p is known, but we may evaluate p at any specific input x.

Example 1.5. First consider a polynomial of a single variable of degree n

$$p(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n.$$

Is p identical to zero? It suffices to evaluate p at $(n+1)$ distinct values of x, e.g.

$$p(1), p(2), \ldots, p(n+1).$$

If any of them evaluates to nonzero, p is clearly not identical to zero. If, on the other hand, all of the $(n+1)$ values are zero, then p is indeed identical to zero. Why is that? By the Fundamental Theorem of Algebra, any nonzero polynomial of degree d has at most d real roots. If p were not identical to zero, then since it has degree n, it would have at most n real roots. Since $p(1) = p(2) = \cdots = p(n+1) = 0$, p has at least $(n+1)$ roots and thus p must be identically zero.

The multivariate case is not so simple, as multivariate polynomials may have infinitely many roots. For instance, the polynomial

$$x^2 - y$$

has uncountably many roots, namely any (x, y) satisfying $y = \sqrt{x}$. So even with an infinitely long list of roots for p, we cannot know for certain whether p is identically zero or not. All hope is not lost, however; it turns out that it's quite unlikely for any nonzero polynomial to evaluate to zero, provided that inputs are selected randomly:

Lemma 1.6 (The Schwartz-Zippel Lemma)**.** *Let $p(x_1, \ldots, x_n)$ be a nonzero polynomial of n variables with degree d. Let S be a finite subset of $\mathbb{R}$, with at least d elements in it. If we assign $x_1, \ldots, x_n$ values from S independently and uniformly at random, then*

$$\mathbb{P}[p(x_1, \ldots, x_n) = 0] \leq \frac{d}{|S|}.$$

This is an *amazing* result — all it takes is to pick a set S and try random inputs. If the polynomial p evaluates to zero, it is highly unlikely that p is nonzero: the probability that p evaluates to zero when it's not identically zero is quite small, especially when $|S| \gg d$. What's also amazing is that there is (yet) no deterministic counterpart to this randomized procedure. In fact, finding a deterministic algorithm for polynomial identity testing would lead to many interesting results, with impact akin to P=NP [KI04].

Before jumping to the full proof of the Schwartz-Zippel Lemma, let's first prove a simpler instance.

1.2 Matrix Identity Testing

Suppose we are given three $n \times n$ matrices A, B, and C. We'd like to test whether $AB = C$. Yes, we could simply multiply A by B, but that would cost $O(n^3)$ time. It turns out we can do better, by turning to a randomized approach.

Let S be a finite subset of $\mathbb{R}$, and let's build a random vector $x \in \mathbb{R}^n$ by choosing each coordinate x_i independently and uniformly at random from S:

$$x_i \sim \text{Uniform}(S)$$

We test whether $ABx = Cx$; if $ABx = Cx$, then we conclude $AB = C$. This procedure costs at most $O(n^2)$, involving three matrix-vector multiplications. This is because $ABx = A(Bx)$. The cost is even lower when the matrices are sparse, in fact it is simply linear in the number of nonzero entries of A, B, C.

Now how likely is the false positive under this regime? That is, if $AB \neq C$, how likely is the outcome $ABx = Cx$? We will show that the false positive is highly unlikely:

Theorem 1.7. *If $AB \neq C$, then*

$$\mathbb{P}_{x_i \sim S}[ABx \neq Cx] \geq 1 - \frac{1}{|S|}.$$

This theorem can be directly proven by an application of Lemma 1.6. But, here we give a direct proof. It turns out that the proof below is in a sense similar to the proof of Lemma 1.6, but it is tuned to the case when p, q have degree 1.

Proof. First, let's write AB and C in terms of row vectors:

$$AB = \begin{bmatrix} — & a_1 & — \\ & \vdots & \\ — & a_n & — \end{bmatrix}, \quad C = \begin{bmatrix} — & c_1 & — \\ & \vdots & \\ — & c_n & — \end{bmatrix}.$$

Since $AB \neq C$, they should differ in at least one row: $a_i \neq c_i$ for some i. We will show that the inner products $\langle a_i, x\rangle$ and $\langle c_i, x\rangle$ are most likely different:

$$\mathbb{P}[\langle a_i, x\rangle \neq \langle c_i, x\rangle] \geq 1 - \frac{1}{|S|}$$

Notice that $\langle a_i, x\rangle$ and $\langle c_i, x\rangle$ are really 1-degree polynomials of variables $x_1, \ldots x_n$, so we could simply apply Schwartz-Zippel Lemma and be done with the proof. But for the sake of learning, let's produce a direct proof that does not depend on the lemma. In fact, the proof here will help us build a proof for the lemma as well.

To show $\mathbb{P}[\langle a_i, x\rangle \neq \langle c_i, x\rangle] \geq 1 - 1/|S|$, we employ a technique known as the **principle of deferred decision**: random choices are made only when they become relevant to the algorithm at hand. Since $a_i \neq c_i$, there exists a coordinate j such that $a_{i,j} \neq c_{i,j}$. Now, set $x_1, \ldots x_n$ except x_j arbitrarily. Since all x_i's are chosen independently of one another, the randomness of x_j is preserved when other x_i's get fixed. Now how likely is the event

$$\sum_{k=1}^{n} a_{ik}x_k - \sum_{k=1}^{n} c_{ik}x_k = 0? \tag{1.1}$$

Equation (1.1) can be re-written as

$$x_j(a_{ij} - c_{ij}) = -\sum_{k \neq j}(a_{ik} - c_{ik})x_k. \tag{1.2}$$

Since all other x_i's are fixed, and $a_{i,j} \neq c_{i,j}$, equation (1.2) holds for only one value of x_j. So *at most one value* from S will satisfy the equation, i.e.,

$$\therefore \quad \mathbb{P}_{x_j \sim S}[\langle a_i, x\rangle = \langle c_i, x\rangle] \leq \frac{1}{|S|}.$$

Since other x_i's don't affect the choice of x_j, the probability is not affected when we let other x_i's be random:

$$\mathbb{P}_{x \sim S}[\langle a_i, x\rangle = \langle c_i, x\rangle] \leq \frac{1}{|S|}.$$

□

Deferred decision is a great tool to use, but we ought to be careful: any analysis we make after fixing certain variables must hold regardless of their values (hence the world "arbitrarily"). The proof of the Schwartz-Zippel Lemma will show how not to use deferred decision.

1.3 Proof of Schwartz-Zippel Lemma

Proof of Lemma 1.6. We proceed by strong induction.

Base case: $n = 1$**.** The problem is reduced to the univariate case presented in Example 1.5.

Inductive step. Suppose that lemma holds for any polynomial with less than n variables; let's show that it would also hold if we have n variables.

First, fix $x_1, \ldots, x_{n-1}$ arbitrarily. Then all values in $p(x_1, \ldots, x_n)$ are known except for x_n, so p becomes a univariate polynomial of x_n of degree k, for some $k \leq d$:

$$p(x_n) = a_k x_n^k + a_{k-1} x_n^{k-1} + \ldots + a_1 x_n^1 + a_0.$$

We've reduced the problem to the univariate case again, so the probability for p to be zero is small:

$$\mathbb{P}[p(x_n) = 0] \leq \frac{k}{|S|} \leq \frac{d}{|S|}. \tag{1.3}$$

So are we done? **No.** We still would need to argue that the probability in (1.3) would be unaffected by the choice of $x_1, \ldots, x_{n-1}$. Unfortunately, this is not the case. Say, an adversary could come and choose $x_1, \ldots, x_{n-1}$ such that the resulting polynomial of x_n is identically 0. In this case, $\mathbb{P}[p = 0] = 1$, and the induction hypothesis does not imply anything.

How can we salvage this argument? Intuitively, we should argue that the adverserial scenario discussed above will be "rare." To that end, we make use of the long division for polynomials [CLO08, p.64]:

> Let $p(x)$ be a polynomial with degree d and $d(x)$ be a polynomial with degree $k \leq d$. Then we can write $p(x)$ as follows:
>
> $$p(x) = d(x)q(x) + r(x)$$
>
> where the **quotient** $q(x)$ has degree at most $(d-k)$ and the **remainder** $r(x)$ has degree at most $k-1$. The polynomial $d(x)$ is the **divisor**.

Let k be the largest degree x_n in all monomials of p. So p can be "divided" by x_n^k as follows:

$$p(x_1, \ldots, x_n) = x_n^k q(x_1, \ldots, x_{n-1}) + r(x_1, \ldots, x_n),$$

where q is a polynomial of $x_1, \ldots, x_{n-1}$ of degree $(d-k)$ and the degree of x_n in r is at most degree $(k-1)$.

Now, we again use the principle of defferred decision. First, we assign values to $x_1, \ldots, x_{n-1}$ uniformly at random from S, and we save the randomness of x_n for later use. Using the inductive assumption, we have

$$\mathbb{P}_{x_1,\ldots,x_{n-1} \sim S}[q(x_1, \ldots, x_{n-1}) = 0] \leq \frac{d-k}{|S|}. \tag{1.4}$$

Observe that if $q \neq 0$, then $p(x_1, \ldots, x_n)$ is a univariate polynomial in x_n, and the coefficient of x_n^k is nonzero. So, conditioned on $q \neq 0$, $p(x_1, \ldots, _n)$ is a univariate polynomial which is not identically 0. Since the degree of this polynomial is k, for a random value of S, it is zero with probability at most $k/|S|$, i.e.,

$$\mathbb{P}_{x_n \sim S}[p = 0 | q \neq 0] \leq \frac{k}{|S|}. \tag{1.5}$$

We can now finish the proof using equations (1.4) and (1.5). by Bayes rule,

$$\begin{aligned}\mathbb{P}[p=0] &= \mathbb{P}[p=0|q=0]\cdot\mathbb{P}[q=0]+\mathbb{P}[p=0|q\neq 0]\,\mathbb{P}[q\neq 0]\\ &\leq \mathbb{P}[q=0]+\mathbb{P}[p=0|q\neq 0]\\ &\leq \frac{d-k}{|S|}+\frac{k}{|S|}=\frac{d}{|S|}.\end{aligned}$$

□

1.4 Bipartite Graph Matching

Polynomial identity testing can be use to determine the existence of a perfect matching within a given bipartite graph $G=(A,B,E)$.

Definition 1.8. A **bipartite graph** $G=(A,B,E)$ where $A=\{a_1,\ldots,a_n\}$ and $B=\{b_1,\ldots,b_n\}$ is a graph where every edge in E connects a vertex in A to a vertex in B.

Here we assume the two sides of the graph have the same number of vertices but this assumption is not necessarily the case in general.

Definition 1.9. A **perfect matching** of graph G is a subset M of edges in E such that every vertex of G is incident to exactly one edge of M.

In order for G to have a perfect matching we need $|A|=|B|$. So assume $|A|=|B|=n$. There is a deterministic algorithm that finds a perfect matching in $O(|E|\sqrt{n})$. Now let's build a randomized algorithm.

First, define the **adjacency matrix** as follows:

$$M(i,j)=\begin{cases}1 & \text{if } a_i\sim b_j\\ 0 & \text{otherwise}\end{cases}$$

Construct a matrix M_x where its entries correspond to indeterminants,

$$M_x(i,j)=\begin{cases}x_{i,j} & \text{if } a_i\sim b_j\\ 0 & \text{o.w.}\end{cases}$$

Theorem 1.10. *Graph G has a perfect matching if and only if the determinant $\det(M_x)$ is not identical to zero.*

Proof. [⇒] Suppose G has a perfect matching. That is, there is a *bijection* σ that maps each $a_i\in A$ to a unique $b_j\in B$. (Since it is a matching, no two vertices in A will be mapped to the same vertex in B. Since the matching is perfect, no vertex in B will be left out.) Therefore, we can see σ as a permutation on the set of integers $[n]=\{1,2,\ldots,n\}$. It follows that

$$\prod_{i=1}^{n}M_x(i,\sigma(i))=\prod_{i=1}^{n}x_{i,\sigma(i)}$$

is a nonzero monomial of the polynomial $\det(M_x)$, recall the formula for the determinant:

$$\det(M_x)=\sum_{\sigma:[n]\to[n]}\operatorname{sgn}(\sigma)\prod_{i=1}^{n}M_x(i,\sigma(i)),$$

In particular, when for the particular permutation σ corresponding to a perfect matching in G in the above polynomial we get a monomial with a nonzero coefficient. This monomial is different from all other monomials of $\det(M_x)$, i.e., there is no cancellations. This means that $\det(M_x)$ is not a zero polynomial.

[$\Leftarrow$] Now, suppose $\det(M_x) \not\equiv 0$. That means that the polynomial has a nonzero monomial $\prod_{i=1}^{n} M_x(i, \sigma(i))$ corresponding to some permutation σ. But that permutation σ gives us a perfect matching in G as desired. □

The above theorem gives a simple and efficient algorithm to test if a given bipartite graph has a perfect matching. By Schwartz-Zippel lemma it is enough to assign values to $x_{i,j}$ from a set S of numbers of size $|S| \geq n^2$. Then, if G has a perfect matching, $\det(M_x) \neq 0$ with probability at least $1 - 1/n$.

The disadvantage of this algorithm is that it doesn't give us the perfect matching; it only tells us whether G has one or not. How do we find the perfect matching? For a bipartite graph G, we choose a big set $|S| \gg n$ and set $x_{ij} = 2^{w_{ij}}$ where w_{ij} is chosen independently and uniformly at random from S. Then, we can show that, with high probability, there is a unique minimum weight perfect. This means that we can write

$$\det(M_x) = 2^{w(F)}(\pm 1 + [\text{even number}]),$$

where $w(F)$ is the sum of the weight of edges of the minimum weight prefect matching F. Having this in hand, all we need to do is to test for every edge of G if that edge is a part of the minimum weight perfect matching. Note that $w(F)$ is uniquely defined in the above, given $\det(M_x)$; in particular, $w(F)$ is the the largest exponent of 2 that divides $\det(M_x)$. For every edge (a_i, b_j), we delete the edge and test if the weight the of the minimum weight perfect matching decreases to $w(F) - w_{i,j}$. If this happens, then $(a_i, b_j) \in F$ and otherwise it is not. This algorithm can be implemented in parallel in $O(\text{polylog}(n))$ time using polynomially many processors. See the following section for more details.

1.5 General Graph Matching

It turns out the idea in the previous section generalizes to find perfect matchings in general graph, although the proof is a lot more difficult. We begin by constructing **skew-symmetric matrix** as follows:

$$M_x(i,j) = \begin{cases} x_{ij} & \text{if } a_i \sim b_j \text{ and } i < j \\ -x_{ij} & \text{if } a_i \sim b_j \text{ and } i \geq j \\ 0 & \text{otherwise} \end{cases}$$

Theorem 1.11. *Graph G has a perfect matching if and only if the determinant $\det(A)$ is not identical to zero.*

We omit the proof. How difficult is to test $\det(M_x)$ against zero? We don't want to spend $O(n^3)$ time to compute the determinant. It turns out that there is a parallel algorithm that comptues the determinant using $poly(n)$ processors in $O(\log^2(n))$ time [Mul86].

References

[CLO08] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, 2008. URL: https://books.google.com/books?id=yCsDO425PC0C (cit. on p. 1-4).

[KI04] V. Kabanets and R. Impagliazzo. "Derandomizing polynomial identity tests means proving circuit lower bounds". In: *Computational Complexity* 13.1-2 (2004), pp. 1–46 (cit. on p. 1-2).

[Mul86] K. Mulmuley. "A fast parallel algorithm to compute the rank of a matrix over an arbitrary field". In: *STOC*. ACM. 1986, pp. 338–339 (cit. on p. 1-6).