6.842 Randomness and Computation February 12, 2022

Lecture 4: Polynomial Identity Testing

Lecturer: Ronitt Rubinfeld *Scribe: Haimoshri*

1 Introduction

The following lecture was on polynomial identity testing and covered both univariate and multivariate polynomials and concluded with an application of it in bipartite matching.

For any two polynomials, we can ask whether they are equal. For example, are the two polynomials, $P(x) = (x+1)^2$ and $Q(x) = x^2+2x+1$, equal? It is easy to see they are. However, if we have polynomials of higher degree, it is harder to see that they are equivalent. For example, are $P(x) = (x+3)^{38}(x-4)^{83}$ and $Q(x) = (x-4)^{38}(x+3)^{83}$ equal? In the factor form, it is not that difficult to see that they are different (you can see that different signs at $x = 0$). But expanded out, they will be harder to compare. Generalising the problem, we can ask the following questions about polynomials.

Question 1. *Given two polynomials, P, Q each with degree $\leq d$, is $P \equiv Q$, that is, $P(x) = Q(x)$ for all x?*

Question 2. *Given polynomial, R, with degree $\leq d$, is $R \equiv 0$, that is, $R(x) = 0$ for all x?*

Note that, if we can answer question 2, we can also answer question 1. We can simply take polynomial $R = P - Q$ and ask whether $R \equiv 0$. The degree of R is bounded above by $max(\deg(P), \deg(Q))$. This is equivalent to asking if $P \equiv Q$. We will see methods to answer the above questions in the following sections.

2 Univariate Polynomial Testing

We will first work with univariate polynomials. For the algorithms discussed later in the section, we will need the following well-known theorem (stated without proof).

Theorem 1. *If $R \not\equiv 0$ has deg $\leq d$, then R has $\leq d$ roots.*

The above theorem is true over any field. In particular, we will use it for Z_q for prime q, i.e. arithmetic mod q.

2.1 Is the Polynomial Identically 0?

We will see two algorithms to test whether a polynomial is identically 0.

2.1.1 Deterministic Algorithm

There is a deterministic algorithm to decide if $R \equiv 0$. It is as follows:

1. Pick $d + 1$ distinct inputs $x_1, \ldots, x_{d+1}$.
2. If for all i, $R(x_i) = 0$, output $R \equiv 0$. Else (that is, if there exists an i for which $R(x_i) \neq 0$), output $R \not\equiv 0$.

The above algorithm has $O(d)$ evaluations of R.

2.1.2 Randomised Algorithm

There is a faster randomised algorithm as follows:

1. Pick $2d$ distinct inputs $x_1, \ldots, x_{2d}$.
2. Repeat the following k times:
   (a) Pick $i \in \{1, \ldots, 2d\}$.
   (b) If $R(x_i) \neq 0$, output $R \not\equiv 0$.
3. Output $R \equiv 0$.

If $R \equiv 0$, the above algorithm always outputs 0 as for all x_i, $R(x_i) = 0$. If $R \not\equiv 0$, as R has degree d, by Theorem 1, we have

$$Pr[R(x_i) = 0] \leq \frac{d}{2d} = \frac{1}{2}.$$

Thus, as Step 2 of the algorithm is repeated k times, for the algorithm to output 0 when $R \not\equiv 0$, we need to choose a root in all k iterations. Therefore, the probability that the algorithm correctly outputs $R \not\equiv 0$ is $\geq 1 - \frac{1}{2^k}$. To get a probability of error $\leq \delta$, we can pick $k = O(\log \frac{1}{\delta})$.

2.2 Human on the Moon

Suppose person E on earth has a $n+1$-bit string $w = w_n \ldots w_0$ and person M has another $n+1$-bit string $w^* = w_n^* \ldots w_0^*$. How do we evaluate if $w = w^*$? Communication is expensive between earth and moon and it would be very wasteful to send the whole strings (for e.g. if they differ in one bit).

Let us define the following degree n polynomials P and P^*:

$$P(x) = w_n x^n + \ldots + w_1 x + w_0,$$

$$P^*(x) = w_n^* x^n + \ldots + w_1^* x + w_0^*.$$

We have that $w = w*$ if and only if $P \equiv P^*$. So, the communication can happen as follows:

- Person E on earth randomly picks $r_1, \ldots, r_k \in \{1, \ldots, 2n\}$ and computes $r_i^{'} = r_i \pmod q$ and $p_i = P(r_i) \pmod q$ for al i, where $q \geq 2n$ is a prime with $O(\log n)$ bits.
- E sends $(r_i^{'}, p_i)$, for all i to person M on the moon.
- M can check whether P^* and P agree on r_i for all i.

Sending w would require $O(n)$ bits. As $r_i^{'}$'s and p_i's are numbers modulo q, they are at most $q-1$ and hence, have $O(\log n)$ bits, giving a total of $O(\log n)$ bits of communication.

3 Multivariate Polynomial Testing

We will now move onto multivariate polynomials.

3.1 Is the Polynomial Identically 0?

We want to determine if $R(x_1, \ldots, x_n) \equiv 0$. We need to define the total degree of a multivariate polynomial.

Definition 1 (Total Degree)**.** *The **total degree** of a multivariate polynomial is the maximum of the sum of degrees of x_i's in any term of the polynomial.*

Example 1. *Consider the polynomial $2xy + 3z^3 + 4xyz^2$. The first term $2xy$ has degree 2, the second term $3z^2$ has degree 3 and the last term has degree 4. Thus, the total degree of the polynomial is 4.*

For multivariate polynomial testing, we encounter a few difficulties. For one, the terms in a polynomial of total degree d can be $\binom{n}{d}$. Also, unlike the univariate case, $R \not\equiv 0$ can have infinitely many roots as shown in the following example.

Example 2. (a) *$R_1(x, y) = xy$ has infinitely many roots of the form $(x, y) = (0, y)$ where y can be any real number. But $R_1(1, 1) = 1$, which means $R_1 \not\equiv 0$.*

(b) *$R_2(x, y) = x - y$ has infinitely many roots of the form $(x, y) = (x, x)$ where x can be any real number. But $R_2(2, 1) = 1$, which means $R_2 \not\equiv 0$.*

However, there is a theorem that allows us to conduct randomised testing of multivariate polynomials.

Theorem 2 (Schwartz - Zippel - DeMillo Lipton)**.** *For R of total degree d such that $R \not\equiv 0$, and given S containing $2d$ elements, pick $x_i \in_R S$ for all i. Then, $Pr[R(x_1, \ldots, x_n) = 0] \leq \frac{d}{|S|}$.*

The above theorem can be proved by induction on d, which we shall not cover here. In the following section, we will see applications of polynomial testing to other problems.

3.2 Bipartite Perfect Matching

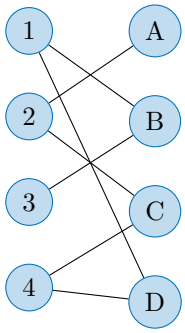
There is a surprising application of multivariate polynomial testing in bipartite matching. Recall that a matching, M, is a subset of edges of a graph such that no two edges in M share an endpoint. A perfect matching is a matching where all vertices of the graph have some edge in M incident to it. Bipartite matching can be solved in polynomial time by using flow algorithms. However, today we will try another approach using polynomial testing. Before we show that approach, we will define the following.

Definition 2 (Tutte Matrix)**.** *The **tutte matrix** A_G of bipartite graph $G = (V, E)$ is defined as follows:*

$$a_{uv} = \begin{cases} X_{uv} & \text{if } (u, v) \in E, \\ 0 & \text{otherwise}, \end{cases}$$

where a_{uv} is the element in uth row and vth column of A_G and the X_{uv}'s are variables.

To understand, we have included an example graph G in Figure 1. A perfect matching in G would be the set $\{(1, D), (2, A), (3, B), (4, C)\}$.

$G =$ 

$$A_G = \begin{bmatrix} 0 & X_{1B} & 0 & X_{1D} \\ X_{2A} & 0 & X_{2C} & 0 \\ 0 & X_{3B} & 0 & 0 \\ 0 & 0 & X_{4C} & X_{4D} \end{bmatrix}$$

Figure 1: An example bipartite graph G with its corresponding tutte matrix A_G.

Lemma 1. *G has perfect matching if and only if $Det[A_G] \neq 0$.*

Proof. From Leibniz's formula, we know that,

$$Det[A_G] = \sum sign(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

where σ is a permutation of $1, ..., n$.

The permutations σ correspond to perfect matchings in G where $(i, \sigma(i))$ is an edge in the matching. $\prod_{i=1}^{n} a_{i,\sigma(i)}$ will be 0 even if only one of $(i, \sigma(i)) \notin E$. This means that $\prod_{i=1}^{n} a_{i,\sigma(i)} \neq 0$ if and only if σ is a perfect matching. So, $Det[A_G] \not\equiv 0$ if and only if there exists some σ which is a perfect matching. □

If we want to test whether a bipartite graph G has a perfect matching, it is enough to test whether $Det[A_G] \not\equiv 0$. To analyse the runtime of testing the above, we see that $Det[A_G]$ is a polynomial and thus, this is reduced to a problem of polynomial testing. $Det[A_G]$ is a polynomial of degree n that has n^2 variables (one for each edge) and $n!$ terms. The number of terms is very large, but we do not need to evaluate them or their coefficients. The sequential running time is $O(n^\alpha)$. However, this works well for parallel algorithms.