CS 6815 Pseudorandomness and Combinatorial Constructions **Fall 2019**

Lecture 1: August 29

Lecturer: Eshan Chattopadhyay *Scribe: Benjamin Y Chan*

1.1 Introduction: Polynomial Identity Testing

We motivate randomized algorithms with an example problem.

Definition 1.1 (Polynomial Identity Testing) *Fix a finite field $\mathbb{F}$. Let $\mathcal{P}_{n,d}$ be all polynomials over $\mathbb{F}$ which are n-variate, degree $\leq d$. The Polynomial Identity Testing problem (PIT) is the following:*

$$\textit{Given } f, g \in \mathcal{P}_{n,d}, \textit{ is } f \equiv g?$$

1.1.1 Equivalence of Polynomials

What is equivalence? It's not a measure of whether two polynomials f and g evaluate the same function. (For example, over $\mathbb{F} = \mathbb{F}_2$, x and x^2 evaluate to the same function; but $x \not\equiv x^2$.)

1.1.2 Representation of Polynomials

This section explores how to represent polynomials and compute their equivalence.

1.1.2.1 As Coefficients

- a vector of coefficients
- at most $\binom{n+d}{\leq d} = \sum_{i=0}^{d} \binom{n+d}{i}$ = number of coefficients of n-variate degree $\leq d$ polynomial

Algorithm Sketch 1.2 (Equivalence of Coefficients) *If f and g are given as vectors of coefficients, equivalence reduces to vector comparison which is easy to compute. Iterate over the list of coefficients and check equality.*

1.1.2.2 As Sum of Products

Of course, there are more efficient representations. Consider the following:

- $f \in \mathcal{P}_{n,d}$ is given as a sum of products of linear terms; e.g. $f = (x_1 + x_6 + x_7)(x_2 + x_1 + x_{100}) + (x_2 + x_7 + x_11)(x_2 + x_3 + x_11 + x_23) + \ldots$.
- we can always compute and expand this expression to a vector of coefficients
- Problem: initial representation could be small, but the corresponding vector of coefficients might be very long and expensive to compute

Algorithm Sketch 1.3 *Compute vector of coefficients, and then run 'simple algorithm' above.*

1.1.2.3 As Arithmetic Circuits: General Representation of Polynomials

Definition 1.4 *An arithmetic circuit C is a directed acyclic graph, with*

- *n nodes with 0 in-degree called leaves,*
- *one node with 0 out-degree 'root-node'*
- *leaves are labeled with variables $(x_1, ..., x_n)$ or field elements*
- *All other nodes are labelled with + (plus) or $\times$ (product)*

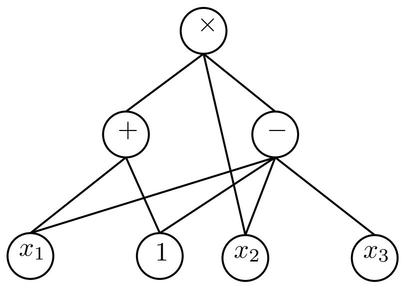

Figure 1.1: Example of an arithmetic circuit

1.1.3 Algorithms for PIT

Claim 1.5 (Deterministic Algorithm for PIT) *We have a deterministic algorithm for PIT that takes $n^{O(d)}$ time.*

For large d, the above is an exponential time algorithm. It is a challenging open question to design a deterministic algorithm for PIT that runs in time $poly(n, d)$.

Algorithm Sketch 1.6 (Randomized Algorithm for PIT) *:*

1. *pick random $x \in \mathbb{F}^n$*
2. *If $f(x) = g(x)$, output 'yes'*
3. *Else output 'no'*

Let $q = |\mathbb{F}|$. Assume q is large (i.e. $q = poly(n, d)$).

Claim 1.7 (Correctness) *If $f \not\equiv g$, $\Pr_{x \in \mathbb{F}^n}[f(x) = g(x)] \leq \frac{d}{q}$.*

The claim above follows directly from the following lemma, which proves that low degree polynomials have few roots.

Lemma 1.8 (DeMillo-Lipton-Schwartz-Zippel Lemma) *Let $f \in \mathcal{P}_{n,d}$ (over finite field $\mathbb{F}$), $f \not\equiv 0$. Then $\Pr_{x \in \mathbb{F}^n}[f(x) = 0] \leq \frac{d}{q}$.*

Proof: We prove by Induction on n. Case $n = 1$: follows from the fact that a univariate polynomial of degree $\leq d$ has at most d roots in $\mathbb{F}$.

Case $n > 1$: We can write f as $f(x_1, ..., x_n) = \sum_{i=0}^{d}(x^i \cdot f_i(x_2, ..., x_n))$. Let i be the maximum number s.t. f_i is not the 0 polynomial, which is well defined since f is not identically zero. Thus, degree $f_i \leq d - i$.

Choose $(x_2, ..., x_n)$ randomly. By induction hypothesis, $\Pr_{(x_2,...,x_n)\sim\mathbb{F}^{n-1}}[f_i(x_2, ..., x_n) = 0] \leq \frac{(d-i)}{q}$.

Next, consider the case that $f_i(x_2, ..., x_n) \neq 0$. Then f is a univariate polynomial of x_1 of degree $i \leq d$ (conditioned on $f_i \neq 0$). Then $\Pr_{x_1}[f(x_1, ..., x_n) = 0] \leq \frac{i}{q}$, using the base case.

This implies that:

$$\Pr_{x\sim\mathbb{F}^n}[f(x) = 0] \leq \frac{i}{q} + \frac{(d-i)}{q} = \frac{d}{q}$$

■

1.2 Complexity Classes

Fix a language L. L is in each of the following complexity classes if $\exists$ a poly-time algorithm A s.t.

Complexity Classes				
Event	RP	coRP	BPP	NP
$x \in L$	$\Pr_y[A(x,y) = 1] \geq \frac{1}{2}$	$\Pr_y[A(x,y) = 1] = 1$	$\Pr_y[A(x,y) = 1] \geq \frac{2}{3}$	$\Pr_y[A(x,y) = 1] > 0$
$x \notin L$	$\Pr_y[A(x,y) = 1] = 0$	$\Pr_y[A(x,y) = 1] \leq \frac{1}{2}$	$\Pr_y[A(x,y) = 0] \geq \frac{2}{3}$	$\Pr_y[A(x,y) = 1] = 0$

From this table, we immediately see that $RP \subseteq NP$, and $coRP \subseteq coNP$. The relationship between BPP and NP is unknown. It is in fact well believed that $P = BPP$.

1.3 Amplification: Error Reduction for Randomized Algorithms

First, we start with some tools that we need later to prove that we can reduce error in the general case.

1.3.1 Concentration Inequalities for Random Variables

1. **Markov's Inequality (First moment method)** Let X be a non-negative random variable over the non-negative real numbers. Then for any $t > 0$,

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$$

   Proof Sketch: $\mathbb{E}[X] = \sum i \cdot \Pr[x = i] \geq t \cdot \Pr[x \geq t]$. ■
2. **Chebyshev's Inequality (Second moment method)**

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}(X)}{t^2} \text{ for any } t > 0$$

   Recall that the variance is $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.
   Proof: Let $\mu = \mathbb{E}[X]$. Apply Markov's Inequality: $\Pr[(X - \mu)^2 \geq t^2] \leq \mathbb{E}[(x - \mu)^2]/t^2$ ■
3. **Chernoff Bounds**

Let $X_1, ..., X_n$ be i.i.d binary random variables. Let $X = \sum_{i=1}^{n} X_i$. Let $\mu = \mathbb{E}[X]$. Let $\mu_i = \mathbb{E}[X_i]$. Then the following inequality holds:

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2 \cdot exp(-\mu \cdot \frac{\delta^2}{3}) \text{ where } 0 < \delta < 1$$

Proof: To be proved in lecture 2. ■

1.3.2 Error Reduction

Recall $L \in BPP, \exists$ poly time algorithm A for L s.t. if $x \in L$, $\Pr_y[A(x,y) = 1] \geq \frac{2}{3}$, and if $x \notin L$, $\Pr_y[A(x,y) = 0] \geq \frac{2}{3}$. How do we reduce this error so that we can construct a randomized algorithm that will succeed with high probability?

Idea: Sample independent sources of randomness $y^1, y^2, ...y^n$ (n will be fixed later). Then run $A(x, y^1), A(x, y^2), ...A(x, y^n)$. Output the majority vote.

Suppose $x \in L$, define $Z_i =$ output of $A(x, y_i)$; $Z = \sum_i Z_i$. Then $\Pr[Z_i = 1] \geq \frac{2}{3}$; thus $\mathbb{E}[X_i] \geq \frac{2}{3}$ and $E[Z] \geq \frac{2}{3}n$.

The probability of error is then $\Pr[Z \leq \frac{n}{2}] \leq \Pr[|Z - \mathbb{E}[Z]| \geq \frac{n}{10}]$. Applying the Chernoff bound this is $\leq 2^{-\Omega(n)} < \epsilon$ (if we pick $n = O(log(1/\epsilon))$).

If the initial algorithm A uses r bits, we now use $r \cdot O(log(1/\epsilon))$ bits.