CS294: Probabilistically Checkable and Interactive Proofs February 21, 2017

Basic Probabilistic Checking 3

Instructor: Alessandro Chiesa & Igor Shinkar *Scribe: Izaak Meckler*

Today we prove the following result from [BFL91] and [BFLS91].

Theorem 1 $\text{NP} \subseteq \text{PCP}\left[O(\log n), \text{polylog}\,(n)\right]$.

The lecture is based on Moshkovitz's notes [DM10] on the same subject.

We begin with a road map to proving this result.

(1) Show that it is NP-hard to decide whether a set of degree-3 multivariate polynomials has a common zero, via a reduction from 3SAT.

(2) Show that we can convert an instance of that problem to an instance with a gap: that is, given a set of polynomials, create a new set of polynomials where at most a small fraction can simultaneously be made zero. Or, in other words, if there is no common zero, then most of the polynomials will be non-zero at any given point.

This gap comes at the price of worse arity. The new polynomials we create each depend on all the variables (while each polynomial in the original set only depended on three). This will have to be dealt with so that we can effectively check the values of individual polynomials at a point.

(3) Like last time, use this new set for our PCP. The proof contains a candidate common zero. We'll pick one polynomial at random and somehow verify that it is in fact zero at the specified point. If there was no common zero, because of the gap we created, we are likely to select a polynomial which is non-zero at that point.

So, finally we'll describe how we actually verify that our randomly selected polynomial is in fact zero at the specified point while using few queries and random coins. This involves a version of the sumcheck protocol.

1 Recap

Last time we proved that $\text{NP} \subseteq \text{PCP}\left[\text{poly}\,(n), O(1)\right]$ by exhibiting such a PCP for 3SAT. What was the proof system for 3SAT? The prover had to give the Hadamard encoding of (an outer product of) the variable assignment. Recall:

Definition 2 *For a vector $v \in \mathbb{F}_2^k$, the Hadamard encoding of v, Had_v, is the truth table for the linear function $\langle v, \cdot \rangle$. That is, Had_v is the 2^k bit string $\text{Had}_v = (\langle v, u \rangle)_{u \in 2^k}$.*

Had takes this k-dimensional space, puts k values on the basis vectors, and then extends linearly to the rest of the space. So if our k bits are $\alpha = (\alpha_1, \ldots, \alpha_k)$, then $\text{Had}_\alpha \colon \mathbb{F}_2^k \to \mathbb{F}_2$ is *the* linear function such that $\text{Had}_\alpha(e_i) = \alpha_i$.

Our last PCP was exponential size because a k-dimensional vector space V has size exponential in k, and the Hadamard encoding required us to write down a bit for every element of V.

So, intuitively, we need a better encoding in order to get a result like 3SAT $\in$ PCP $[O(\log n), \text{polylog}\,(n)]$. We want the encoding to still be redundant so we can make sure that the given word is close to the corresponding code word, but we want it also to be more succinct, to be less wasteful than Hadamard. And how we're going to do that is by using low-degree polynomials instead of linear functions (which are polynomials of total degree one).

2 Zero testing

Zero testing is the following problem: given a set of polynomials $p_1, \ldots, p_\ell \in \mathbb{F}[x_1, \ldots, x_n]$, determine if they have a common zero. That is, decide if there are $a_1, \ldots, a_n \in \mathbb{F}$ such that $p_i(a_1, \ldots, a_n) = 0$ for each $i \in [\ell]$.

Degree-3 common zero testing is the same problem but with the additional assumption that the input polynomials are degree 3. We will only need to consider degree 3 common zero testing.

Fact 3 *Degree 3 common zero testing is* NP*-complete over any finite field* $\mathbb{F}$.

Proof: Reduction from 3SAT. We'll describe how to map each literal e to a polynomial $p(e)$, each clause C to a polynomial $p(C)$.

1. $p(x_i) := 1 - x_i$
2. $p(\overline{x_i}) := x_i$
3. $p(e_1 \vee e_2 \vee e_3) := p(e_1)p(e_2)p(e_3)$

Now given a 3-CNF formula

$$\varphi = \bigwedge_i C_i$$

consider the set of degree-3 polynomials $S_\varphi := \{\, p(C_i) \,\}_i$.

We claim that the polynomials S_φ have a common zero if and only if φ has a satisfying assignment (in fact, the satisfying assignment is the common zero).

Suppose $a = (a_1, \ldots, a_n)$ is a zero of every $p(C_i)$. Write $p(C_i) = p(e_{i,1})p(e_{i,2})p(e_{i,3})$. Since fields don't have zero divisors, for each i, we must have $p(e_{i,j})(a) = 0$ for some $j \in [3]$. Since $p(e_{i,j}) = x_{k(i,j)}$ or $1 - x_{k(i,j)}$, we thus have $a_{k(i,j)} \in \{0, 1\}$. Consider the assignment which takes each $x_{k(i,j)} := a_{k(i,j)}$ and assigns the other variables arbitrarily. It is clear that for each i, this assignment will cause the j-th variable of C_i to evaluate to true, and thus it constitutes a satisfying assignment for φ.

Conversely, suppose $a_1, \ldots, a_n$ is a satisfying assignment for φ. Then for each i, some literal e_i evaluates to 1 under the assignment. By the definition of p, this implies that $p(e_i) = 0$. This in turn means that the polynomial $p(C_i) = 0$ (since $p(e_i)$ is a factor of it). Thus $a_1, \ldots, a_n$ is a zero of every polynomial in S_φ. □

We are going to construct our PCP for this problem.

3 From zero testing to gap zero testing

The problem of finding a common zero of a set of polynomials $p_1, \ldots, p_\ell$ could have a small gap. That is, it is possible that $p_1, \ldots, p_\ell$ do not all have a common zero, but we can simultaneously satisfy $\ell - 1$ of them. This is a problem for constructing a PCP, for which we need it to be impossible to simultaneously satisfy more than a constant fraction of the constraints.

Thus, we want to make a gap for the zero testing problem. That is, given a set of ℓ polynomials S we want to create a new set S' such that

1. If S has a common zero then so does S'.
2. If S does not have a common zero then each assignment is a zero of at most half of S'.

Last time we did this by taking all possible linear combinations over $\mathbb{F}_2$ of our original set of polynomials: If the polynomials themselves had a common zero, then clearly any linear combination has that zero, and if there is no common zero, then at least half of the linear combinations (over $\mathbb{F}_2$) would be non-zero.

This will not work here. If we take S' to be all possible linear combinations, its size will be exponential in the size of S, and the verifier will have to use polynomially much randomness to pick random constraints to check. Notice though that if we picked, say, ℓ^2 random linear combinations then by a Chernoff bound the probability that any given assignment is a zero of more than $1/2 + \epsilon$ of them would be very small.

But, we would like to use only polylogarithmic randomness, so we're going to use the Reed–Solomon code to construct a small set of polynomials like this deterministically. We'll define a matrix $M \in \mathrm{Mat}(\mathbb{F}, 10m \times m)$ such that for all non-zero $v \in \mathbb{F}^m$, $\#(i : (Mv)_i \neq 0) \geq 9m$. That is, for non-zero v, 90% the coordinates of Mv will be non-zero.

Let $a_1, \ldots, a_{10m}$ be distinct non-zero elements of $\mathbb{F}$. For concreteness take $a_i := i$ (via the quotient map $\mathbb{Z} \to \mathbb{F}$). Now let

$$M := \begin{pmatrix} a_1 & a_1^2 & \cdots & a_1^m \\ a_2 & a_2^2 & \cdots & a_2^m \\ \vdots & \cdots & \vdots & \\ a_{10m} & a_{10m}^2 & \cdots & a_{10m}^m \end{pmatrix}$$

This matrix is called a Vandermonde matrix and is the generating matrix for the Reed–Solomon code.

Claim 4 *For all non-zero $v \in \mathbb{F}^m$, $\#(i : (Mv)_i \neq 0) \geq 9m$.*

Proof: Let $v = (c_1, \ldots, c_m)$ be a non-zero vector. Define the polynomial

$$f_v(x) := \sum_{k=1}^{m} c_k x^k$$

Then

$$(Mx)_i = \sum_{k=1}^{m} c_k a_i^k = f_v(a_i)$$

Since f_v has at most m roots and all the a_i are distinct, we have

$$\#\left(i : (Mv)_i = 0\right) = \#\left(i : f_v(a_i) = 0\right) \leq m$$

and so $\#\left(i : (Mv)_i \neq 0\right) \geq 9m$ as desired. □

We can now use this to reduce a zero testing instance to a zero testing instance with a large gap. Let $S = \{ p_1, \ldots, p_m \}$ be a zero testing instance and define

$$(q_1, \ldots, q_{10m}) := M(p_1, \ldots, p_m)$$

Let's check that this reduction satisfies the completeness and soundness conditions.

1. *Completeness:*

   If $p_1, \ldots, p_m$ have a common zero α, then

   $$\begin{aligned} M \cdot (p_1(\alpha), \ldots, p_m(\alpha)) &= M \cdot 0^m \\ &= 0^{10m} \\ &= (q_1(\alpha), \ldots, q_{10m}(\alpha)) \end{aligned}$$

   so α is a common zero of the q_i.

2. *Soundness:*

   If $p_1, \ldots, p_m$ do not have a common zero, then for any $\alpha \in \mathbb{F}^n$, there is some p_i with $p_i(\alpha) \neq 0$. Thus $v := (p_1(\alpha), \ldots, p_m(\alpha))$ is a non-zero vector, which means at most m of the $10m$ entries of $Mv = (q_1(\alpha), \ldots, q_{10m}(\alpha))$ are 0. That is, at most 1/10 of the q_j have α as a zero.

If you only learn one thing from this lecture, this is a good trick to know.

4 Locality and sumcheck

We got the gap we wanted, but now each polynomial depends on all the variables, so we lost the locality. We're going to use arithmetiziation to fix this. Look at a degree three polynomial $q(X)$. We have

$$q(X) = \sum_{1 \leq i \leq j \leq k \leq n} q_{ijk} x_i x_j x_k$$

If i, j, k are not in increasing order, define q_{ijk} to be 0.

Let $H \subseteq \mathbb{F}$ and take $m \in \mathbb{N}$ such that $|H|^m = n$. Let $\sigma \colon H^m \to [n]$ be an arbitrary bijection.

We're going to take

1. $|H| = O(\log n)$
2. $|\mathbb{F}| = \text{polylog}\,(n)$
3. $m = O\left(\frac{\log n}{\log \log n}\right)$

so that $|H|^m = n$ and $|\mathbb{F}|^m = \text{poly}\,(n)$.

We'll use H^m to encode the integers $[n]$. Define $Q\colon (H^m)^3 \to \mathbb{F}$ by

$$Q(u,v,w) := q_{\sigma(u),\sigma(v),\sigma(w)}$$

For an assignment to the variables $(\alpha_i)_{i\in[n]}$, let $A\colon H^m \to \mathbb{F}$ be defined by $A(u) := \alpha_{\sigma(u)}$.

Note that

$$\begin{aligned}\sum_{u,v,w\in H^m} Q(u,v,w)A(u)A(v)A(w) &= \sum_{u,v,w\in H^m} q_{\sigma(u),\sigma(v),\sigma(w)}\alpha_{\sigma(u)}\alpha_{\sigma(v)}\alpha_{\sigma(w)} \\ &= \sum_{1\le i\le j\le k\le n} q_{ijk}\alpha_i\alpha_j\alpha_k \\ &= q(\alpha)\end{aligned}$$

Why would we go from such a nice representation to such an ugly one? Well we're going to work with the sumcheck protocol, and sumcheck likes to work with such guys.

5 Recall low degree extensions

The low-degree extension of a function $f\colon H^m \to \mathbb{F}$ is the unique polynomial $\widehat{f}\colon \mathbb{F}^m \to \mathbb{F}$ of degree $|H|-1$ in each variable such that $f(z) = \widehat{f}(z)$ for all $z \in H^m$. Explicitly:

$$\widehat{f}(z_1,\ldots,z_m) := \sum_{a\in H^m} f(a_1,\ldots,a_m)\cdot L_a(z_1,\ldots,z_m)$$

where L_a is the polyonmial of degree $|H|-1$ in each variable such that, for all $z \in H^m$, $L_a(z) = 1$ if $z = a$ and $L_a(z) = 0$ if $z \neq a$. That is,

$$L_a(z_1,\ldots,z_m) := \prod_{i\in[m]}\prod_{b\in H\setminus\{\,a_i\,\}} \frac{z_i - b}{a_i - b}$$

We'll use these to embed a sum-check proof in the PCP we're developing.

6 Putting it all together

Now we can begin to define the full PCP $[O(\log n), \text{polylog}\,(n)]$ proof system for degree 3 common zero testing.

Say the input is a set of polynomials $p_1,\ldots,p_\ell$. We'll use our gap creation procedure on these to produce $q_1,\ldots,q_{10\ell}$. The symbols that the proof is written in will be elements of $\mathbb{F}$. The proof will then consist of

Type of data	**Value provided by the honest prover**
The evaluation table of of a function $\mathbb{F}^m \to \mathbb{F}$	The evaluation table of $\widehat{A}$, the LDE of a correct variable assignment A.
For each $j \in [n]$, the evaluation tabe of a function Φ_j, $(\mathbb{F}^m)^3 \to \mathbb{F}$	We expect that $$\Phi_j(u,v,w) = \widehat{Q_j}(u,v,w)\widehat{A}(u)\widehat{A}(v)\widehat{A}(w).$$
For each $i \in [3m]$ and for every possible choice of $r_1, \ldots, r_{3m} \in \mathbb{F}$, a degree $\lvert H \rvert$ univariate polynomial $G^i_{r_1,\ldots,r_{i-1}}$ (given as a vector of coefficient).	We expect $G^i_{r_1,\ldots,r_{i-1}}(z)$ to be $$\sum_{h_{i+1},\ldots,h_{3m} \in H} \Phi_j(r_1, \ldots, r_{i-1}, z, h_{i+1}, \ldots, h_{3m})$$
Some kind of proof that that the evaluation tables provided correspond to polynomials of the expected degrees.	We will cover this next lecture.

The verifier will do the following:

(1) Check (via some mechanism which we will talk about next time) that the evaluation tables provided correspond to polynomials of the expected degrees.

(2) Pick a random $j \in [n]$.

(3) Pick random points $u, v, w \in \mathbb{F}^m$ and check that

$$\Phi_j(u,v,w) = \widehat{Q_j}(u,v,w)\widehat{A}(u)\widehat{A}(v)\widehat{A}(w)$$

This requires querying the proof about $\Phi_j(u,v,w), \widehat{A}(u), \widehat{A}(v)$, and $\widehat{A}(w)$. $\widehat{Q_j}(u,v,w)$ can be computed directly from the input.

(4) Now we do a sumcheck protocol to check whether $\sum_{u,v,w \in H} \Phi_j(u,v,w) = 0$. Pick random points $r_1, \ldots, r_{3m} \in \mathbb{F}$ and check that

(a)
$$\sum_{h \in H} G^1(h) = 0$$

(b) For $1 \le i \le 3m-1$,
$$G^i_{r_1,\ldots,r_{i-1}}(r_i) = \sum_{h \in H} G^{i+1}_{r_1,\ldots,r_i}(h)$$

(c)
$$G^{3m}_{r_1,\ldots,r_{3m}} = \Phi_j(r_1, \ldots, r_{3m})$$

6.1 Analyzing resource usage

6.1.1 Randomness

Let's count the amount of randomness our verifier uses. Going step by step, it uses

1. The randomness r_{LDT} used by the yet-to-be-described degree testing procedure.
2. To pick $j \in [n]$ we use $\log n$ bits.
3. To pick 3 random points of $\mathbb{F}^m$, we use $3m \log |\mathbb{F}| = O(\log n)$ bits.
4. To pick $r_1, \dots, r_{3m}$, we again use $3m \log |\mathbb{F}| = O(\log n)$ bits.

In total we use $O(\log n) + r_{\text{LDT}}$, which we will see next time is $O(\log n)$.

6.1.2 Queries

Next let's count how many queries are made, keeping in mind that a single query reads an element of $\mathbb{F}$.

1. The number of queries q_{LDT} used by the still-yet-to-be-described degree testing procedure.
2. 0 queries.
3. Here we make 4 queries.
4. Finally, here we make $O(3m\,|H|) = \text{polylog}\,(n)$ queries.

So in total we use $\text{polylog}\,(n) + q_{\text{LDT}}$, which we will see next time is $\text{polylog}\,(n)$.

6.2 Completeness and soundness

6.2.1 Completeness

Completeness is straightforward. Suppose $q_1, \dots, q_n$ do have a common zero given by an assignment A and the prover writes down everything as expected. Then the verifier goes through steps 1, 2, 3, and also passes through step 4 since

$$\begin{aligned}\sum_{h_1 \in H} G^1(h_1) &= \sum_{h_1, h_2, \dots, h_{3m} \in H} \Phi_j(h_1, \dots, h_{3m}) \\ &= q_j(A) \\ &= 0\end{aligned}$$

and

$$\begin{aligned}G^i_{r_1,\dots,r_{i-1}}(r_i) &= \sum_{h_{i+1},\dots,h_{3m} \in H} \Phi_j(r_1, \dots, r_{i-1}, r_i, h_{i+1}, \dots, h_{3m}) \\ &= \sum_{h_{i+1},\dots,h_{3m} \in H} \Phi_j(r_1, \dots, r_{i-1}, r_i, h_{i+1}, \dots, h_{3m}) \\ &= \sum_{h \in H} \sum_{h_{i+2},\dots,h_{3m} \in H} \Phi_j(r_1, \dots, r_i, h, h_{i+2}, \dots, h_{3m}) \\ &= \sum_{h \in H} G^{i+1}_{r_1,\dots,r_i}(h)\end{aligned}$$

and also

$$G^{3m}_{r_1,\ldots,r_{3m}} = \Phi_j(r_1, \ldots, r_{3m}).$$

So the verifier will accept.

6.2.2 Soundness

Let's check soundness under the assumption that the malicious prover really writes down evaluation tables of polynomials of the expected degrees. We will address the case that the functions are not low-degree polynomials in the next class.

Claim 5 *Our protocol has soundness error less than* $\frac{1}{2}$.

Proof: Suppose $q_1, \ldots, q_n$ do not have a common zero. We may assume that the proof is formatted correctly (since otherwise the verifier will just reject).

In step (1), the probability that we pick a j such that q_j is zero on the provided assignment A is $\frac{1}{10}$, so we may give up that much soundness and assume that the assignment encoded by A does not satisfy q_j. This implies that

$$\sum_{u,v,w \in H^m} \widehat{Q_j}(u,v,w)\widehat{A}(u)\widehat{A}(v)\widehat{A}(w) \neq 0.$$

If $\Phi_j(u,v,w) \not\equiv \widehat{Q_j}(u,v,w)\widehat{A}(u)\widehat{A}(v)\widehat{A}(w)$ as polynomials, then by Schwartz–Zippel the probability that step (2) accepts is at most $\frac{6m|H|}{|\mathbb{F}|}$. Otherwise, we have $\sum_{u,v,w \in H^m} \Phi_j(u,v,w) \neq 0$, and the check protocol will accept with probability at most $\frac{3m|H|}{|\mathbb{F}|}$. Therefore, the probability that the verifier accepts the proof is at most

$$\begin{aligned} \frac{1}{10} + O\left(\frac{m\,|H|}{|\mathbb{F}|}\right) &= \frac{1}{10} + o(1) \\ &\leq \frac{1}{2} \end{aligned}$$

as required. □

References

[BFL91] László Babai, Lance Fortnow, and Carsten Lund, *Non-deterministic exponential time has two-prover interactive protocols*, Computational Complexity **1** (1991), 3–40, Preliminary version appeared in FOCS '90.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy, *Checking computations in polylogarithmic time*, Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, STOC '91, 1991, pp. 21–32.

[DM10] Michael Forbes Dana Moshkovitz, *PCP and hardness of approximation course notes, lecture 3*, `http://www.cs.utexas.edu/~danama/courses/pcp-mit/3-sum-check.pdf`, 2010.