6.5240 Sublinear Time Algorithms November 30, 2022

Lecture 22

Lecturer: Ronitt Rubinfeld *Scribe: Joseph Heerens*

1 Background from Last Lecture

Last lecture, it was shown that there is an algorithm such that if a function f is ϵ-close to some linear function g, then we may "self-correct" f to g in $O(\frac{1}{\beta})$ calls to f such that we may output the function g with probability at least $1-\beta$. Therefore, we may test if a function is linear through $O(\frac{1}{\beta})$ calls to f such that if f is linear, we always pass and if f is ϵ-far, then the self tester fails with probability at least $1-\beta$. Further, it is also known that the self-corrector does not know the value of the function.

Consider the class of Probablistic Checkable Proofs (PCP). This class, denoted $PCP(r,q)$, is the statement that using r random bits and q queries to the proof, there exists some proof such that any verifier will accept an instance of the question if it is true, and there is no proof that will be accepted by a verifier with probability at least $\frac{1}{4}$ if we do not have an instance of the question. The goal using this information is to show that $NP \subseteq PCP(O(n^3), O(1))$. This is relatively surprising given that only a constant number of queries and polynomial random bits are needed to have an instance of a problem such as 3-SAT, which we will look at today.

Denote the inner product $x \cdot y = \sum_i x_i \cdot y_i$ and then the outer product of two n-bit vectors x and y as $x \circ y = (x_1y_1, x_1y_2, ..., x_ny_n)$. Then it was shown that if $\overline{a} \neq \overline{b}$, as two n-bit vectors, then it is true that $P[\overline{a} \cdot \overline{r} \neq \overline{b} \cdot \overline{r}] \geq \frac{1}{2}$, for any $\overline{r} = \{0,1\}^n$. Further, if A, B are two $n \times n$ matrices with $A \cdot B \neq C$, then $P[A \cdot B \cdot \overline{r} \neq C \cdot \overline{r}] \geq \frac{1}{2}$, for any n dimensional vector $\overline{r}$.

2 3SAT Satisfied

We wish to utilize the above facts shown in the previous lecture to prove this. First, we want to write down all possible values of the product. This is done via arithmetization, where some boolean formula F is written as $A(F)$, which is an algebraic expression in Z_2. A clause should evaluate to 1 if true and 0 if false, with x_i being x_i and $\overline{x_i} = 1 - x_i$. From boolean algebra, $\alpha \wedge \beta = \alpha \cdot \beta$, also $\alpha \vee \beta = 1-(1-\alpha)(1-\beta)$, and $\alpha \vee \beta \vee \gamma = 1 - (1-\alpha)(1-\beta)(1-\gamma)$. Then a only satisfies F if and only if $A(F)(a) = 1$.

Now let input F be represented as $\wedge_i C_i$ which are $C_i = (y_{i_1} \vee y_{i_2} \vee y_{i_3})$, so that we may arithmetize. So now we take $\mathcal{C}(x) = (\hat{C}_1(x), \hat{C}_2(x), ..., \hat{C}_k(x))$. which has each of the $\hat{C}_i$ is the complement of the arithmetization of C_i, and evaluate to 0 if an assignment validates it. Thus, if some x satisfies F, then $\mathcal{C}(x) = (0, 0, ..., 0)$. This creates a scenario where each of the $A(\hat{C}_i(x))$ will be a polynomial of degree at most 3, and further the verifier will know each coefficient of the polynomial. Our argument must show that we may convince the verifier that all of these polynomials are 0 without sending the assignment, and while only looking at a constant number of locations in the proof.

The solution presented will try to use the theorem presented earlier and take $\mathcal{C}(x) \cdot r$ and then set b equal to the zero vector. If $\mathcal{C}(x)$ is 0, then we should always have $\mathcal{C}(x) \cdot r = 0 \cdot r$, otherwise we may apply the theorem from before which would state that the probability that the two vectors are not equal is at least $\frac{1}{2}$, particularly equal to $\frac{1}{2}$ in modulo 2.

2.1 High Level Proof

What does $\mathcal{C}(a) \cdot r$ look like? Well we have that $\sum_i r_i \hat{C}_i(a) = \Gamma + \sum_i a_i \alpha_i + \sum_{ij} a_i a_j \beta_{ij} + \sum_{ijk} a_i a_j a_k \delta_{ijk}$ all in mod 2, and the verifier knows the values of each α, β, γ, and Γ. This is because verifier may compute these in the allotted space. However, the verifier does not know that assignment values.

As a high level thought, we will write down all degree $1, 2$, and 3 assignments of the function. Proof π contains all linear assignments of $\overline{a}$. We write $\alpha : F_2^2 \to F_2$ with $\alpha(x) = \sum a_i x_i = a^T x$ write table of α for all x. These are all linear functions evaluated at a. We also write all degree 2 functions of $\overline{a}$. This is $\beta : F_2^{n^2} \to F_2$. So $\beta(y) = \sum a_i a_j y_{ij} = (a \circ a)^T y$. Repeating this for degree 3 gives $\gamma(z) = \sum a_i a_j z_{ijk} = (a \circ a \circ a)^T z$, and so to describe the truth table of this is going to be in space 2^{n^3} so that is the size of this proof. The proof will contain the description of each of these functions for any given possible values of x, y, and z. Also note that each of these functions are linear in x, y, z, and so we may apply the linear function self-correcting properties from before.

Now the verifier must validate that ensure that each of the truth tables are consistent with one another and linear. It then must see whether or not an assignment will correspond to 0 for some assignment. It is first checked whether the function is in good form.

2.2 Outline of Proof

We know from the algorithm presented before, we may check if each of the tables for α, β, and γ may be checked for 1/8-approximation to linear functions using constant query time to get a constant in the probability that it is checked. Therefore, this utilizes $O(1)$ queries. Denote $sc - \tilde{A}$, $sc - \tilde{B}$, and $sc - \tilde{C}$ as the self-checked tables for each of the degrees of functions, such that the algorithm is run and they are properly corrected to linear functions.

To check that these each correspond, only pass if $sc - \tilde{B} = sc - \tilde{A} \circ sc - \tilde{A}$ and $sc - \tilde{C} = sc - \tilde{B} \circ sc - \tilde{A}$. This will verify that the assignments align with the desired coefficients. We will show this works.

First, randomly pick x_1, x_2, x, y, and then

$$sc - \tilde{A}(x_1) \cdot sc - \tilde{A}(x_2) = \sum a_i x_{1i} \circ \sum a_i x_{2i} = \sum_{i,j} a_i x_{1i} a_j x_{2j} = \sum_{i,j} b_{i,j} x_{1i} x_{2j} = sc - \tilde{B}(x_1 \circ x_2).$$

Additionally,

$$sc - \tilde{A}(x) \cdot sc - \tilde{B}(y) = \sum a_i x_i \circ \sum b_{ij} y_{ij} = \sum_{i,j,k} a_i x_i b_{jk} y_{jk} = \sum_{i,j,k} c_{i,j,k} x_i y_{jk} = sc - \tilde{C}(x \circ y).$$

Then, if $a \circ a \neq b$, it can be seen that $P[(a \circ a)r \neq br] = \frac{1}{2}$ for any r via our fact. Therefore, the probability that we will fail testing by simply checking $(a \circ a)r \neq br$ for some r will be greater than $\frac{1}{4}$ which is the desired.

Now going back to the proof, we seek the verifier validate linearity and consistency. It is known that linearity may follow here if we choose a β small enough that a union bound will see the table unlikely to ever see an error. Further, it is already shown that with high enough probability, we can verify that the different tables upon being self-corrected will be consistent.

All that is left to do is the satisfiablility check. Imagine taking an $r \in F_2^n$. Then we may compute each of the coefficients that are written in the table, and self correct these using the function from earlier.

Then, plugging the r in should always give us 0 if the function is satisfiable and otherwise at some point would give us 1, each with probability $1/2$. Therefore, after some constant number of queries, it may be seen that we fail anything that is not satisfiable with probability at least $3/4$ and will always accept a proof that should pass it, thus showing that 3SAT is in $PCP(O(n^3), O(1))$.

An interesting aside is that this problem happens to be in $PCP(O(\log n), O(1))$, but the proof is too difficult for this class.