CS 221: Computational Complexity **Prof. Salil Vadhan**

Lecture Notes 22

April 19, 2010 **Scribe: Tova Wiener**

1 Linearity Testing

Last time, we proved that $\mathbf{NP} \subseteq \mathbf{PCP}(\text{poly}(n), O(1))$. In this model, the verifier tosses a polynomial number of coins, which means the proof could be exponentially long, but still uses only a constant number of queries. A key step was having the verifier check whether the proof oracles f and g are close to being linear functions. Proof oracle was supposed to consist of $(\text{WH}(u), \text{WH}(uu^T))$, where $\text{WH}(u)$ is the truth-table of the linear function $f_u(x) = \sum_i u_i x_i \bmod 2$, and the verifier assumes that both are close to linear functions, and proceeds from there.

Definition 1 *$f_1, f_2 : \mathbb{Z}_2^n \to \mathbb{Z}_2$ are α-**close** if $\Pr_{x \in \mathbb{Z}_2^n}[f_1(x) = f_2(x)] \geq \alpha$.*

Definition 2 *f is **linear** if $\forall x, y \in \mathbb{Z}_2^n$ $f(x) + f(y) = f(x+y)$, where all arithmetic is in $\mathbb{Z}_2^n$.*

Linearity Test for oracle $f : \mathbb{Z}_2^n - > \mathbb{Z}_2$: Choose x, y randomly $(\overset{R}{\leftarrow})$ $\mathbb{Z}_2^n$. Accept if $f(x+y) = f(x) + f(y)$.
Even though linearity is a global property, we will show that this local checking is sufficient.

Theorem 3

- ***Completeness**: If f is linear, then $\Pr_{x,y}[f(x+y) = f(x) + f(y)] = 1$.*
- ***Soundness**: if $\Pr_{x,y}[f(x+y) = f(x) + f(y)] \geq 1 - \delta$ for $\delta < 1/14$, then f is $(1-2\delta)$-close to some linear $\tilde{f}$.*

Proof: The completeness follows immediately from the definition of linearity, so we proceed with soundness. Take $\tilde{f}(x) = maj_y\, f(x+y) - f(y)$. If f were truly linear, then this would always be $f(x)$. Here we take the majority vote. If there is a tie, we can break it arbitrarily, but we'll see below that there will never be a tie.

Claims:

1. Votes are overwhelming - always an overwhelming winner: $\forall x \Pr_y[f(x+y) - f(y) = \tilde{f}(x)] \geq 1 - 4\delta$. (Fraction of votes that the majority gets is approaching 1.)
2. $\tilde{f} \approx f$: $\Pr_x[\tilde{f}(x) = f(x)] \geq 1 - 2\delta$, for a slightly different constant.
3. $\tilde{f}$ is linear: $\forall x, y$ $\tilde{f}(x+y) = \tilde{f}(x) + \tilde{f}(y)$ (provided $\delta < 1/14$).

Proof of Claim 1: Fix x. Note that the test does not look at worst case x, but here we prove something about $\tilde{f}$ for the worst case x.

$$\begin{aligned}\Pr_y[f(x+y)-f(y)\neq\tilde{f}(x)] &\leq 2\cdot\Pr_{y,z}[f(x+y)-f(y)\neq f(x+z)-f(z)]\\ &\leq 2\cdot(\Pr[f(x+y)+f(z)\neq f(x+y+z)]+\Pr[f(x+z)+f(y)\neq f(x+y+z)])\\ &= 2\cdot(\delta+\delta)=4\delta\end{aligned}$$

The first line follows because $\tilde{f}$ is the majority vote over z of $f(x+z)-f(z)$, so at least half the time $\tilde{f}(x)=f(x+z)-f(z)$. Now, none of the evaluation points are themselves worst case. The second line follows because if the top equation fails, then it must be the case that one of the terms on the second line fails, by a union bound. We then have two single instances of the linearity test, so the overall bound is just δ for each term, which gives 2δ, so 4δ overall.

Proof of Claim 2:

$$\Pr_x[f(x)\neq\tilde{f}(x)]\leq 2\cdot\Pr_{xz}[f(x)\neq f(x+z)-f(z)]\leq 2\delta,$$

where the first inequality follows again because $\tilde{f}(x)$ is the majority of $f(x+z)-f(z)$ and the second because it is just an instance of the linearity test.

Proof of Claim 3: Now nothing is random in the claim: we are making a statement about every x and y. Fix x, y, and show $\tilde{f}(x)+\tilde{f}(y)=\tilde{f}(x+y)$. We know by Claim 1:

$$\begin{aligned}\Pr_w[\tilde{f}(x)=f(x+w)-f(w)]&\geq 1-4\delta\\ \Pr_z[\tilde{f}(y)=f(y+z)-f(z)]&\geq 1-4\delta\\ \Pr_{w,z}[\tilde{f}(x+y)=f(x+y+w+z)-f(w+z)]&\geq 1-4\delta\end{aligned}$$

Now $f(x+w)+f(y+z)=f(x+y+w+z)$ holds except with probability δ over w,z. And that $f(w)+f(z)=f(w+z)$ holds except with probability δ, for a total of $12\delta+\delta+\delta=14\delta$, So if $\delta<1/14$, then we get that $P_{w,z}[\tilde{f}(x)+\tilde{f}(y)=\tilde{f}(x+y)]>0$. But this event has no randomness, so it just means that $\tilde{f}(x)+\tilde{f}(y)=\tilde{f}(x+y)$. ■

What we have done works for testing homomorphisms $f: G\to H$ for any abelian groups G, H. (When $|H|>2$, then the majority vote in the definition of $\tilde{f}$ should be replaced with a plurality vote — taking the most likely value.) There is a tighter analysis specific to $G=\mathbb{Z}_2^n$ and $H=\mathbb{Z}_2$ in one of the later chapters of Arora–Barak which uses Fourier Analysis. The tighter analysis is important for getting tight inapproximability results.

2 More Inapproximability

Recall for a maximization problem Π, $0<\rho\leq 1$:

$$\begin{aligned}(\textsc{Gap}_\rho\Pi)_Y&=\{(x,t):\text{Opt}_x\geq t\}\\ (\textsc{Gap}_\rho\Pi)_N&=\{(x,t):\text{Opt}_x<\rho t\}\end{aligned}$$

The general optimization problem is to find the largest number of satisfied clauses, largest independent set. We make this decisional by introducing the threshold, and asking if the optimal value is greater than t, or less than ρt. Minimization case is similar, except that $\rho > 1$, and the condition is $\mathrm{Opt}_x \leq t$, and $\mathrm{Opt}_x > \rho t$.

Examples:

- $\Pi = \textsc{Max}\text{-3SAT}$, $x = \{\phi_1, ..., \phi_m\}$, each ϕ_i = OR of 3 literals. Opt_x = max # clauses that can be satisfied.
- $\Pi = \textsc{Max}\text{-}q\text{CSP}$, same but each ϕ_i = function of q variables (not necessarily OR).
- $\textsc{Gap}_{c,s}$ instead of $\textsc{Gap}_\rho$: Here the promise problem has YES instances x where $\mathrm{Opt}_x \geq c \cdot m$ and NO instances where $\mathrm{Opt}_x \leq s \cdot m$. (We switch from strict inequality to non-strict inequality, but not particularly significant.)

We saw that the PCP Theorem is equivalent to the following:

Theorem 4 (PCP Theorem, restated) *There is a constant q such that $\textsc{Gap}_{1,1/2}\textsc{Max}\text{-}q\text{CSP}$ is **NP**-hard under polynomial-time mapping reductions.*

PCP Theorem gives the first inapproximability result, and we use it to get other inapproximability result.
Today we assume PCP Theorem and deduce more inapproximability results. But first we state a couple of simple corollaries of the PCP Theorem.

Corollary 5 *There is a constant q s.t. $\forall \rho > 1/2, \textsc{Gap}_\rho\textsc{Max}\text{-}q\text{CSP}$ is **NP**-hard under polynomial-time mapping reductions.*

We just switched from c, s version to a single parameter. This is immediate: if we could use a single parameter, we could just use that to distinguish between fully satisfiable and 1/2 satisfiable. Just let $t = m = \#clauses$.

Corollary 6 *There is a constant q such that if $\mathbf{P} \neq \mathbf{NP}$, there is no poly-time algorithm that given any satisfiable q-CSP instance finds an assignment satisfying more than half of the clauses.*

If we did have such an algorithm, then we could separate yes and no instances of $\textsc{Gap}_{1,1/2}\textsc{Max}\text{-}q\text{CSP}$.
Now, we use the PCP Theorem to deduce inapproximability for 3-SAT.

Theorem 7 *There is a constant $\epsilon > 0$ such that $\textsc{Gap}_{1,1-\epsilon}\textsc{Max}\text{-3SAT}$ is **NP**-hard under polynomial-time mapping reductions.*

Compared to the basic PCP Theorem, here we get inapproximability for $q = 3$ and for constraints of a specific type (ORs of literals), but we lose in the amount of approximation that is hard. From our proof, ϵ will be very small, but it still means we cannot distinguish perfectly satisfiable from 99.99% satisfiable.

Proof: We'll show that $\textsc{Gap}_{1,1/2}\textsc{Max-}q\text{CSP} \leq \textsc{Gap}_{1,1-\epsilon}\textsc{Max-}3\text{SAT}$.
Given a $\textsc{Max-}q$CSP instance $\phi = \{\phi_1, \ldots, \phi_m\}$, we'll apply the Cook-Levin reduction (separately) on each constraint $\phi_i(x_1, \ldots, x_n)$ to get a 3-CNF formula $\phi'_i = \wedge_j \phi'_{ij}$, where ϕ'_i depends on the q variables in ϕ_i plus some auxiliary variables $y_1^{(i)}, \ldots, y_t^{(i)}$. The number t of auxiliary variables and the number k of 3-clauses ϕ'_{ij} equal the number of gates in a circuit computing ϕ'_i, which is at most $2^q = O(1)$ since ϕ'_i depends on at most q variables.
If $\phi_i(x) = 1$, then there exists an assignment $y^{(i)}$ to the new variables such that all clauses of ϕ'_i satisfied. On the other hand, if $\phi_i(x) = 0$, then no matter how we assign $y^{(i)}$, at least one clause of ϕ'_i will not be satisfied.
So we map ϕ to $\phi' = \{\phi'_{ij}\}$. If ϕ satisfiable, ϕ' is satisfiable. If ϕ is at most 1/2-satisfiable, then ϕ' is at most $(1 - \frac{1}{2k})$-satisfiable, where again $k \leq 2^q$. For each one constraint ϕ_i not satisfied, you must violate at least a $1/k$ fraction of the new clauses ϕ'_{ij}. So we have a very small constant $\epsilon = 1/2k = \Theta(1/2^q)$ here. ■

2.1 Vertex Cover

Theorem 8 *There is a constant $\epsilon > 0$ such that $\textsc{Gap}_{2/3,2/3+\epsilon} - \textsc{Min-}\text{VC}$ is **NP**-hard under Karp reductions.*

Here $\textsc{Gap}_{a,b} - \textsc{Min-}\text{VC}$ for $a < b$ is the promise problem where YES instances are graphs with a vertex cover of size at most an and NO instances are graphs where every vertex cover has size at least bn, where n is the number of vertices in the graph.

Corollary 9 *There exists a constant $\epsilon > 0$ such that there is no $(1 + \epsilon)$-approximation algorithm for vertex cover unless $\mathbf{P} = \mathbf{NP}$.*

Proof: We'll show that the usual reduction from 3SAT to VC gives a reduction from $\textsc{Gap}_{1,1-\epsilon}\textsc{Max-}3\text{SAT}$ to $\textsc{Gap}_{2/3,(2+\epsilon)/3}\textsc{Min-}\text{VC}$. (So we can do reductions between approximation problems, but we need to be more careful and quantitative, and check how the objective function quantitatively translates from one to the other.)
$\phi \mapsto G$: Clause $x \vee y \vee \neg z$ goes to triangle of vertices all connected, which we think of as being labelled with the literals in the clause. Then we connect all oppositely labeled vertices: if there is anther triangle with $\neg x$, we connect x to $\neg x$.
ϕ satisfiable $\Rightarrow$ G has a VC of size at most $2m$, where m is the number of clauses. (There are $2m/3$ triangles in the graph.) The reason is that in each clause you need two vertices to cover three edges. In each clause, omit something true, then you will never omit both endpoints of one of the crossing edges.
ϕ not $(1 - \epsilon)$-satisfiable $\Rightarrow$ all vertex covers of G are of size at least $(2 + \epsilon)m$. Otherwise, it must be the case that at least $(1 - \epsilon)m$ triangles have a vertex not in the cover. Setting these corresponding literals to true satisfies $(1 - \epsilon)m$ clauses. ■