CS 880: Advanced Complexity Theory 2/29/2008

Lecture 15: Hardness Amplification within NP

Instructor: Dieter van Melkebeek Scribe: Priyananda Shenoy

In the last lecture, we introduced the general idea of boosting the hardness of a function by taking k independent copies of the function and aggregating them using another function h. We obtained the following result:

Lemma 1. *If f is balanced and ϵ-hard for circuits of size at most s, then $g = h \circ f^{\otimes k}$ is ϵ'-hard for circuits of size at most s' where $\epsilon' = \frac{1}{2} - \frac{1}{2}\mathrm{E}_R[|\widehat{h|_R}(\emptyset)|] - k\delta$, $s' = \Omega\left(\frac{\delta^2}{\log \frac{1}{\delta\epsilon}}\right) s - size(h)$, and R is a random restriction with parameter $\rho \geq \epsilon$.*

Our goal here is to find a suitable h, which boosts a "slightly average-case hard" function (i.e., with hardness $\epsilon = \Omega(1/\mathrm{poly}(n))$) to a function $g \in \mathrm{NP}$ which is close to $\frac{1}{2}$-hard. We need h to have the following properties:

- The expected bias of h must be small for ϵ' to be as close to $\frac{1}{2}$ as possible. This implies that h must be balanced or close to balanced. Indeed, if h is unbalanced and f is, then g can be predicted with nontrivial advantage.
- *size*(h) must not be too large.
- h must be in NP.
- h must be monotone.

Since the absolute value in the expected bias expression makes it hard to compute directly, we make use of the following bounds in our analysis:

$$\overbrace{\mathrm{E}_R\left[\left(\widehat{h|_R}(\emptyset)\right)^2\right]}^{(*)} \leq \mathrm{E}_R\left[\left|\widehat{h|_R}(\emptyset)\right|\right] \leq \sqrt{\mathrm{E}_R\left[\left(\widehat{h|_R}(\emptyset)\right)^2\right]}.$$

Note that the right-hand side is the square root of the left-hand side.

Using the analysis from Lecture 9, we obtain the following expression for $(*)$:

$$\begin{aligned}\mathrm{E}_R\left[\left(\widehat{h|_R}(\emptyset)\right)^2\right] &= \mathrm{E}_I\left[\sum_{S\subseteq I}\left(\widehat{h}(S)\right)^2\right]\\ &= \sum_{S\subseteq[n]} \Pr[S\subseteq I](\widehat{h}(S))^2\\ &= \sum_{S\subseteq[n]}(1-\rho)^{|S|}(\widehat{h}(S))^2.\end{aligned}$$

We get the last equality from the fact that for any element, the probability that it is in I is $(1-\rho)$. For a set S to be a subset of I, all its elements must be in I. Since the elements are independent, the probability of all the elements of S being in I is $(1-\rho)^{|S|}$.

From the result obtained in Lecture 6, Section 3, the noise sensitivity of h can be written as

$$NS_\epsilon(h) = \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]}(1-2\epsilon)^{|S|}(\widehat{h}(S))^2.$$

Using this we can relate $(*)$ and the noise sensitivity as

$$\mathrm{E}_R\left[\left(\widehat{h|_R}(\emptyset)\right)^2\right] = 1 - 2NS_{\frac{\rho}{2}}(h).$$

So for hardness amplification, we need a balanced (or almost balanced) monotone h with noise sensitivity as large as possible. Let us look at some properties of the noise sensitivity:

- For any function h, $NS_0(h) = 0$.
- For any balanced function h, $NS_{\frac{1}{2}}(h) = \frac{1}{2}$.
- For any function h, $NS_1(h) = \Pr[h(x) \neq h(-x)]$, which is 0 if h is even and 1 if h is odd.
- For an odd function h, $NS_{1-\epsilon}(h) = 1 - NS_\epsilon(h)$.
- For any nonconstant function h, $NS_\epsilon(h)$ strictly increases between $\epsilon = 0$ and $\epsilon = \frac{1}{2}$.

Using the last property, since $\rho \geq \epsilon$, $NS_{\frac{\rho}{2}}(h) \geq NS_{\frac{\epsilon}{2}}(h)$. Substituting this in the expression for ϵ' in Lemma 1, we get

$$\epsilon' \geq \frac{1}{2} - \frac{1}{2}\sqrt{1 - 2NS_{\frac{\epsilon}{2}}(h)} - k\delta.$$

We now examine some monotone functions in NP as candidates for h, and analyze their noise sensitivity.

1 Noise sensitivity of monotone functions

1.1 Majority

As seen earlier, the *Majority* function is defined as

$$MAJ_n(x) = sign(\sum_{i=1}^{n} x_i).$$

Majority is balanced and monotone, so it is a feasible candidate for our purposes. But the following fact shows that it has low noise sensitivity and hence is not useful for hardness amplification.

Proposition 1. $NS_\epsilon(MAJ_n) = O(\sqrt{\epsilon})$.

Proof. (Sketch) We obtain y by flipping each of the n bits of x with probability ϵ. Let F be the set of bits which got flipped, therefore $|F|$ is roughly $(n \cdot \epsilon)$. The question is whether flipping the bits in F changed the majority, i.e., $sign(\sum_{i \notin F} x_i + \sum_{i \in F} x_i) \neq sign(\sum_{i \notin F} x_i - \sum_{i \in F} x_i)$. Since $\sum_{i \notin F} x_i + \sum_{i \in F} x_i = \sum_{i=1}^{n} x_i$, this is the same as asking $sign(\sum_{i=1}^{n} x_i) \neq sign(\sum_{i=1}^{n} x_i - 2\sum_{i \in F} x_i)$. The term $\sum_{i \in F} x_i$ is close to normally distributed, with mean 0 and standard deviation $\sqrt{n\epsilon}$. Hence with high probability its absolute value is $O(\sqrt{n\epsilon})$. In that case, switching from x to y means subtracting a term of size $O(\sqrt{n\epsilon})$ from x. The majority of y will be different only if the value of $\sum_{i=1}^{n} x_i$ was close enough to 0 that subtracting the new term results in a sign change. The weight of x_i's which are atmost $O(\sqrt{n\epsilon})$-far from balanced is $O(\sqrt{n\epsilon} \cdot \frac{1}{\sqrt{n}}) = O(\sqrt{\epsilon})$. □

Since we usually take ϵ to be $\frac{1}{n^{O(1)}}$, this isn't good enough to boost the hardness to a value close to $\frac{1}{2}$.

1.2 Recursive Majority

Although simple Majority doesn't have give us the necessary hardness boosting, *Recursive Majority* does work.

Definition 1. *The* Recursive Majority *function at the* $(d+1)^{th}$ *level is defined recursively as*

$$REC\text{-}MAJ_{3^{d+1}} = MAJ_3 \circ REC\text{-}MAJ_{3^d}^{\otimes 3},$$

where MAJ_3 *is the simple Majority function with 3 inputs.*

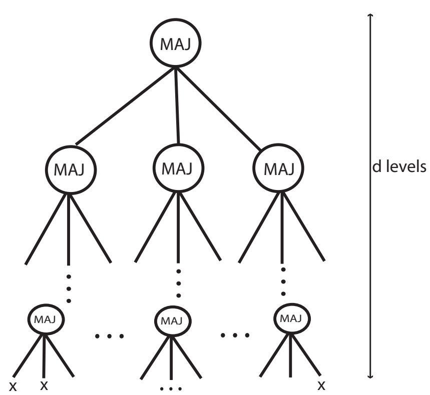

Fig 1: *Recursive Majority* function.

Recursive Majority is balanced and monotone. To analyze its noise sensitivity, we first analyze the noise sensitivity of MAJ_3. Since MAJ_3 has 3 inputs, each of which is flipped independently with probability ϵ, its noise sensitivity is a polynomial $p(\epsilon)$ of degree at most 3. We can determine the coefficients using the facts that $p(0) = 0$, $p(\frac{1}{2}) = \frac{1}{2}$, $p(1) = 1$, and by observing that $p(\epsilon) \sim \frac{3}{2}\epsilon$ for $\epsilon \to 0$. The latter follows from the following observation: For small ϵ, flips of more than one bit are of second order. There are three distinct ways in which exactly one flip can occur. Each of those happens with probability $\sim \epsilon$, and flips the value of MAJ_3 with (conditional) probability 1/2 (namely when the two other bits balance out). It follows that

$$p(\epsilon) = \epsilon^3 - \frac{3}{2}\epsilon^2 + \frac{3}{2}\epsilon$$

We now use this expression to get the noise sensitivity of the Recursive Majority function. The following fact suggests a recursive approach.

Proposition 2. *If f is balanced then,*

$$NS_\epsilon(h \circ f^{\otimes k}) = NS_{NS_\epsilon(f)}(h).$$

Proof. The LHS can be written as

$$\Pr[h(x_1, x_2, \ldots, x_k) \neq h(y_1, y_2, \ldots, y_k)], \tag{1}$$

where the x_i's and y_i's are obtained as follows: For each $1 \leq i \leq k$ and $1 \leq j \leq n$ independently, pick $x_{i,j}$ uniformly at random from $\{-1, 1\}$ and set $y_{i,j} = x_{i,j}$ with probability ϵ and $y_{i,j} = -x_{i,j}$ otherwise; set $x_i = f(x_{i,1}, x_{i,2}, \cdots, x_{i,n})$, and $y_i = f(y_{i,1}, y_{i,2}, \cdots, y_{i,n})$.

Since each $x_{i,j}$ is an independent random bit and f is balanced, the x_i's are independent random bits. Moreover, for any fixed i, the probability that $x_i \neq y_i$ equals the probability that $f(x) \neq f(y)$, where x is uniform over $\{-1, 1\}^n$ and $y \sim_\epsilon x$. The latter probability equals $NS_\epsilon(f)$ by the definition of noise sensitivity. Thus, we can generate the distribution of the x_i's and y_i's by picking each x_i uniformly at random and picking $y_i \sim_{NS_\epsilon(f)} x_i$. Under that distribution, (1) is nothing else than the noise sensitivity of h at $NS_\epsilon(f)$, i.e., the RHS of the proposition. $\square$

Using this proposition, we obtain the following expression for the noise sensitivity of Recursive Majority:

$$NS_\epsilon(\text{REC-MAJ}_{3^d}) = p^{(\circ d)}(\epsilon),$$

where $p^{(\circ d)}$ denotes the dth iterate of p. Since $p(\epsilon) = NS_\epsilon(MAJ_3)$ is increasing on $[0, 1]$ and, MAJ_3 being odd, $p(1 - \epsilon) = NS_{1-\epsilon}(MAJ_3) = 1 - NS_\epsilon(MAJ_3)$ for $\epsilon \in [0, \frac{1}{2}]$, this means that $\frac{1}{2}$ is the only attractive fixed point of p. Thus, for values of $\epsilon \in (0, \frac{1}{2})$, $p^{(\circ d)}(\epsilon)$ increases monotonically to $\frac{1}{2}$ when d increases. To get a bound on how large d needs to be, we analyze how fast the convergence happens.

For small values of ϵ, we can neglect the higher order terms and approximate $p(\epsilon) \approx \frac{3}{2}\epsilon$, so we get $p^{(\circ d)}(\epsilon) \approx \left(\frac{3}{2}\right)^d \epsilon$. This approximation is accurate as long as $p^{(\circ d)}(\epsilon)$ remains close to 0, say as long as $(\frac{3}{2})^d \epsilon \leq \epsilon_0$ for some positive constant ϵ_0. For ϵ close to $\frac{1}{2}$, we can use a different approximation by tracking the distance from $\frac{1}{2}$, which we will denote by $\eta = \frac{1}{2} - \epsilon$. For small values of η, we can use the approximation $\frac{1}{2} - p(\frac{1}{2} - \eta) \approx \frac{3}{4}\eta$, so $\frac{1}{2} - p^{\otimes d}(\frac{1}{2} - \eta) \approx \left(\frac{3}{4}\right)^d \eta$. This approximation is accurate as soon as $\eta \leq \eta_0$, where η_0 is some positive constant. Since ϵ_0 and η_0 are constants, we can bridge the range between $\epsilon = \epsilon_0$ and $\epsilon = \frac{1}{2} - \eta_0$ using a constant number of iterations of p. As a result,

$$d = \widetilde{\Theta}\left(\log_{\frac{3}{2}}\left(\frac{1}{\epsilon}\right) + \log_{\frac{4}{3}}\left(\frac{1}{\eta}\right)\right)$$

suffices to make sure $p^{(\circ d)}(\epsilon) \geq \frac{1}{2} - \eta$, or equivalently, $NS_\epsilon(\text{REC-MAJ}_{3^d}) \geq \frac{1}{2} - \eta$.

We would like to express the resulting hardness of the function $g = (\text{REC-MAJ}_{3^d} \circ f^{\otimes k})$ as a function of its input size $m = k \cdot n = 3^d \cdot n$. Using the bound on d obtained above, we get:

$$m = \left(\left(\frac{1}{\epsilon}\right)^{\log_{\frac{3}{2}} 3} \cdot \left(\frac{1}{\eta}\right)^{\log_{\frac{4}{3}} 3}\right)^{\widetilde{\Theta}(1)}.$$

As a result, for $\epsilon = \frac{1}{n^{\Theta(1)}}$, we can make g to be ϵ'-hard where $\epsilon' = \frac{1}{2} - \frac{1}{m^{\alpha}}$ for some constant $\alpha > 0$. In other words, we are able to boost the average-case hardness from inverse polynomial to some fixed polynomial level. Since Recursive Majority is monotone and in NP, this way we can achieve average-case hardness $H_g(m) \geq m^{\alpha}$ for some function $g \in$ NP assuming there exists a function $f \in$ NP such that every polynomial-size circuit has to err on at least an inverse polynomial fraction of the inputs in computing f.

1.3 Tribes function

As seen in Lectures 1 and 2, the Tribes function is monotone and almost balanced. The following fact, which we will prove in Homework 2, gives a bound on the noise sensitivity of the Tribes function.

Proposition 3. *For any fixed ϵ, $NS_{\epsilon}(TRIBES_k) = \frac{1}{2} - \frac{1}{k^{\Omega(1)}}$.*

As a result, using the Tribes function as the aggregator, we can boost the average-case hardness from some constant level to $\frac{1}{2} - \frac{1}{m^{\beta}}$ for some $\beta > 0$. We can use the REC-MAJ function described above to boost inversely polynomial hardness to the required constant level, and then apply the Tribes function to further boost the hardness. Further analysis shows that $\beta > \alpha$, i.e., the Tribes function can bring us further in terms of hardness amplification than the Recursive Majority function.

While this is a good improvement, we would like to have the hardness even closer to $\frac{1}{2}$. Ideally we would like to have hardness to be linear-exponentially close to $\frac{1}{2}$. We will first examine if this is possible at all using our approach based on balanced monotone functions.

2 Bounds on the noise sensitivity of monotone functions

We know from Lecture 5 that for a monotone function the individual influences are given by $I_i(h) = \widehat{h}(\{i\})$. For balanced functions we showed in Homework 1 that $I = \sum_{i=1}^{k} I_i \geq 1$, which implies that $\sum_{i=1}^{k} I_i^2 \geq \frac{1}{k}$. For such functions the noise sensitivity satisfies:

$$\begin{aligned}
NS_{\frac{\epsilon}{2}}(h) &= \frac{1}{2} - \frac{1}{2}\sum_{S \subseteq [k]} (1-\epsilon)^{|S|}(\widehat{h}(S))^2 \\
&\leq \frac{1}{2} - \frac{1}{2}(1-\epsilon)\sum_{i=1}^{k} I_i^2 \\
&\leq \frac{1}{2} - \Omega\left(\frac{1}{k}\right)
\end{aligned}$$

If k is polynomial in n, then the best we can achieve is polynomial closeness to $\frac{1}{2}$. To get exponentially closer, we need to make k exponential in n. We cannot directly do this because of the following problems:

1. The input length $m = k \cdot n$ no longer remains polynomial in n. This is no good because the hardness of the original function f holds with respect to circuits whose size is bounded by some function $s(n)$, whereas the hardness of the original function should hold for circuits of sufficiently large size $s'(m)$, and we always have $s'(m) < s(n)$.

2. We need to evaluate f k times, which cannot be done in polynomial time if k is super-polynomial in n.

We can resolve the first issue by using derandomization. Instead of generating $k \cdot n$ truly random bits, we can use a pseudorandom generator to produce $k \cdot n$ pseudorandom bits from few truly random bits. There exist such pseudorandom generators with seed length $O(n^{\frac{3}{2}})$, resulting in an input length for g of $m = O(n^{\frac{3}{2}})$.

We can resolve the second issue using nondeterminism and the Tribes function. Instead of evaluating f on all k copies, we can non nondeterministically guess which tribe is satisfied and evaluate only the copies of f that are involved in the tribe. This requires evaluating f only about $\log k$ times.

Using this approach we can boost hardness within NP from $\frac{1}{n^{O(1)}}$ to $\frac{1}{2} - \frac{1}{2^{\Omega(m^{2/3})}}$. Whether we can boost it to $\frac{1}{2} - \frac{1}{2^{\Omega(m)}}$ is still an open question.