CS225: Pseudorandomness	Prof. Salil Vadhan

Lecture 19: Worst-case vs. Average-case Hardness

April 19, 2007

Based on scribe notes by Denis Chebikin and Sasha Schwartz.

In Lecture Notes 18, we saw how to construct pseudorandom generators from boolean functions that are very hard on average, where every nonuniform algorithm running in time t must err with probability greater than $1/2 - 1/t$ on a random input. Now we want to relax the assumption to refer to worst-case hardness. Here is the correct definition for probabilistic algorithms.

Definition 1 *A function $f_\ell : \{0,1\}^\ell \to \{0,1\}$ is* worst-case hard for (nonuniform) time $t(\ell)$ *if, for all (nonuniform) probabilistic algorithms A running in time $t(\ell)$ and all sufficiently large ℓ, there exists $x \in \{0,1\}^\ell$ such that $\Pr[A(x) \neq f_\ell(x)] > 1/3$, where the probability is over the coin tosses of A.*

Note that, for *deterministic* algorithms A, the conclusion simply says $A(x) \neq f(x)$. In the nonuniform case, restricting to deterministic algorithms is without loss of generality because we can always derandomize the algorithm using (additional) nonuniformity. Specifically, following the proof that $\mathbf{BPP} \subseteq \mathbf{P/poly}$, it can be shown that if f is worst-case hard for nonuniform deterministic algorithms running in time $t(\ell)$, then it is worst-case hard for nonuniform probabilistic algorithms running in time $\Omega(t(\ell)/\ell)$.

A natural goal is to be able to construct an average-case hard function from a worst-case hard function. More formally, given a function $f : \{0,1\}^\ell \to \{0,1\}$ vs. time $t = t(\ell)$, construct a function $\hat{f} : \{0,1\}^{O(\ell)} \to \{0,1\}$ such that $\hat{f}$ is average-case hard vs. time $t' = t^{\Omega(1)}$. Moreover, we would like $\hat{f}$ to be in $\mathbf{E}$ if f is in $\mathbf{E}$. (Whether we can obtain a similar result for $\mathbf{NP}$ is a major open problem, and indeed there are negative results ruling out natural approaches to doing so.)

Our approach to doing this will be via error-correcting codes. Specifically, we will show that if $\hat{f}$ is the encoding of f in an appropriate kind of error-correcting code, then worst-case hardness of f implies average-case hardness of $\hat{f}$.

Specifically, we view f as a message of length $L = 2^\ell$, and apply an error-correcting code $\mathrm{Enc} : \{0,1\}^L \to \Sigma^{\hat{L}}$ to obtain $\hat{f} = \mathrm{Enc}(f)$, which we view as a function $\hat{f} : \{0,1\}^{\hat{\ell}} \to \Sigma$, where $\hat{\ell} = \log \hat{L}$. Pictorially:

$$\boxed{\text{message } f : \{0,1\}^\ell \to \{0,1\}} \longrightarrow \boxed{\text{Enc}} \longrightarrow \boxed{\text{codeword } \hat{f} : \{0,1\}^{\hat{\ell}} \to \Sigma}.$$

(Ultimately, we would like $\Sigma = \{0,1\}$, but along the way we will discuss larger alphabets.)

Now we argue the average-case hardness of $\hat{f}$ as follows. Suppose, for contradiction, that $\hat{f}$ is not δ average-case hard. By definition, there exists an efficient algorithm A with $\Pr[A(x) = \hat{f}(x)] > 1-\delta$. We may assume that A is deterministic by fixing its coins. Then A may be viewed as a received word in $\Sigma^{\hat{L}}$, and our condition on A becomes $\Delta(A, \hat{f}) < \delta$. So if Dec is a δ-decoding algorithm for Enc, then $\mathrm{Dec}(A) = f$. By assumption A is efficient, so if Dec is efficient, then f may be efficiently computed everywhere. This would contradict our worst-case hardness assumption, assuming that

Dec(A) gives a time $t(\ell)$ algorithm for f. However, the standard notion of decoding requires reading all $2^{\hat{\ell}}$ values of the received word A and writing all 2^{ℓ} values of the message Dec(A), and thus Time(Dec(A)) $\gg 2^{\ell}$. Everything is easy for nonuniform time 2^{ℓ}, and even in the uniform case we are mostly interested in $t(\ell) \ll 2^{\ell}$. To solve this problem we introduce the notion of local decoding.

Definition 2 *A* local δ-decoding algorithm *for some* Enc : $\{0,1\}^L \to \Sigma^{\hat{L}}$ *is a probabilistic oracle algorithm* Dec *with the following property. Let* $f : \{0,1\}^{\ell} \to \{0,1\}$ *be any message with associated codeword* $\hat{f} = \mathrm{Enc}(f)$*, and let* $g : \{0,1\}^{\hat{\ell}} \to \Sigma$ *be such that* $\Delta(g, \hat{f}) < \delta$*. Then for all* $x \in \{0,1\}^{\ell}$ *we have* $\Pr[\mathrm{Dec}^g(x) = f(x)] \geq 2/3$*, where the probability is taken over the coins flips of* Dec*.*

In other words, given oracle access to g, we want to efficiently compute any bit of f with high probability. So both the input (namely, g) and the output (namely, f) are treated implicitly; the decoding algorithm does not need to read/write either in its entirety. This makes it possible to have sublinear-time (or even polylogarithmic-time) decoding. Also, we note that the bound of 2/3 in the definition can be amplified in the usual way. Having formalized a notion of local decoding, we can now make our earlier intuition precise.

Proposition 3 *Let* Enc *be an error-correcting code with local* δ*-decoding algorithm* Dec*, and let* f *be worst-case hard for nonuniform time* t*. Then* $\hat{f} = \mathrm{Enc}(f)$ *is* (t', δ) *average-case hard, where* $t' = t/\mathrm{Time}(\mathrm{Dec})$*.*

Proof: We do everything as explained before except with Dec^A in place of Dec(A), and now the running time is at most Time(Dec) $\cdot$ Time(A), since Dec can make at most Time(Dec) calls to A. ■

To simplify matters, we have only dealt with the nonuniform case. We used nonuniformity in the proof to fix the coin flips of A, making it deterministic. There are similar results for the uniform case (choosing the coin tosses of A randomly instead of nonuniformly, and replacing δ-average-case hard by, say, $(\delta/3)$-average-case hard in the conclusion), so nonuniformity is not being used in any essential way. Anyway, in light of the above proposition, our task is now to find an error-correcting code with a local decoding algorithm. Specifically, we would like the follows parameters.

1. We want $\hat{\ell} = O(\ell)$, or equivalently $\hat{L} = \mathrm{poly}(L)$.
2. We would like Enc to be computable in time $2^{O(\ell)} = \mathrm{poly}(L)$. This is because we want $f \in \mathbf{E}$ to imply $\hat{f} \in \mathbf{E}$.
3. Ideally we would like a local δ-decoding algorithm with $\delta = 1/2 - t^{-\Omega(1)}$ (for $\Sigma = \{0,1\}$), but this is impossible because one cannot decode from beyond half the minimum-distance. We will see how to deal with this next time.
4. Recalling that $\hat{f}$ will be average-case hard against time $t/\mathrm{Time}(\mathrm{Dec})$, we would want the running time of Dec to be at most t^{α} for a constant $\alpha < 1$.