Notes for Lecture 12

Scribed by Jonah Sherman, posted March 10, 2009

Summary

Today we prove the Goldreich-Levin theorem.

1 Goldreich-Levin Theorem

We use the notation
\[ \langle x, r \rangle := \sum_i x_i r_i \mod 2 \]  \hspace{1cm} (1)

**Theorem 1 (Goldreich and Levin)** Let \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) be a permutation computable in time \( r \). Suppose that \( A \) is an algorithm of complexity \( t \) such that
\[
P_{x,r}[A(f(x), r) = \langle x, r \rangle] \geq \frac{1}{2} + \epsilon \]  \hspace{1cm} (2)

Then there is an algorithm \( A' \) of complexity at most \( O((t + r)\epsilon^{-2}n^{O(1)}) \) such that
\[
P_{x}[A'(f(x)) = x] \geq \frac{\epsilon}{4} \]

Last time we proved the following partial result.

**Lemma 2 (Goldreich-Levin Algorithm – Weak Version)** Suppose we have access to a function \( H : \{0,1\}^n \rightarrow \{0,1\} \) such that, for some unknown \( x \), we have
\[
P_{r \in \{0,1\}^n}[H(r) = \langle x, r \rangle] \geq \frac{7}{8} \]  \hspace{1cm} (3)

where \( x \in \{0,1\}^n \) is an unknown string.

Then there is an algorithm GLW that runs in time \( O(n^2 \log n) \) and makes \( O(n \log n) \) oracle queries into \( H \) and, with probability at least \( 1 - \frac{1}{n} \), outputs \( x \).
This gave us a proof of a variant of the Goldreich-Levin Theorem in which the right-hand-side in (2) was $\frac{15}{16}$. We could tweak the proof Lemma 2 so that the right-hand-side of (4) is $\frac{3}{4} + \epsilon$, leading to proving a variant of the Goldreich-Levin Theorem in which the right-hand-side in (2) is also $\frac{3}{4} + \epsilon$.

We need, however, the full Goldreich-Levin Theorem in order to construct a pseudo-random generator, and so it seems that we have to prove a strengthening of Lemma 2 in which the right-hand-side in (4) is $\frac{1}{2} + \epsilon$.

Unfortunately such a stronger version of Lemma 2 is just false: for any two different $x, x' \in \{0, 1\}^n$ we can construct an $H$ such that

$$\mathbb{P}_{r \sim \{0, 1\}^n}[H(r) = \langle x, r \rangle] = \frac{3}{4}$$

and

$$\mathbb{P}_{r \sim \{0, 1\}^n}[H(r) = \langle x', r \rangle] = \frac{3}{4}$$

so no algorithm can be guaranteed to find $x$ given an arbitrary function $H$ such that $\mathbb{P}[H(r) = \langle x, r \rangle] = \frac{3}{4}$, because $x$ need not be uniquely defined by $H$.

We can, however, prove the following:

**Lemma 3 (Goldreich-Levin Algorithm)** Suppose we have access to a function $H : \{0, 1\}^n \to \{0, 1\}$ such that, for some unknown $x$, we have

$$\mathbb{P}_{r \sim \{0, 1\}^n}[H(r) = \langle x, r \rangle] \geq \frac{1}{2} + \epsilon$$

where $x \in \{0, 1\}^n$ is an unknown string, and $\epsilon > 0$ is given.

Then there is an algorithm $GL$ that runs in time $O(n^2 \epsilon^{-4} \log n)$, makes $O(n \epsilon^{-4} \log n)$ oracle queries into $H$, and outputs a set $L \subseteq \{0, 1\}^n$ such that $|L| = O(\epsilon^{-2})$ and with probability at least $1/2$, $x \in L$.

The Goldreich-Levin algorithm $GL$ has other interpretations (an algorithm that learns the Fourier coefficients of $H$, an algorithm that decodes the Hadamard code is sub-linear time) and various applications outside cryptography.

The Goldreich-Levin Theorem is an easy consequence of Lemma 3. Let $A'$ take input $y$ and then run the algorithm of Lemma 3 with $H(r) = A(y, r)$, yielding a list $L$. $A'$ then checks if $f(x) = y$ for any $x \in L$, and outputs it if one is found.

From the assumption that
it follows by Markov’s inequality (See Lemma 9 in the last lecture) that

\[ \mathbb{P}_x \left[ \mathbb{P}[A(f(x), r) = \langle x, r \rangle] \geq \frac{1}{2} + \epsilon \right] \geq \frac{\epsilon}{2} \]

Let us call an \( x \) such that \( \mathbb{P}[A(f(x), r) = \langle x, r \rangle] \geq \frac{1}{2} + \frac{\epsilon}{2} \) a good \( x \). If we pick \( x \) at random and give \( f(x) \) to the above algorithm, there is a probability at least \( \epsilon/2 \) that \( x \) is good and, if so, there is a probability at least \( 1/2 \) that \( x \) is in the list. Therefore, there is a probability at least \( \epsilon/4 \) that the algorithm inverts \( f() \), where the probability is over the choices of \( x \) and over the internal randomness of the algorithm.

2 The Goldreich-Levin Algorithm

In this section we prove Lemma 3.

We are given an oracle \( H() \) such that \( H(r) = \langle x, r \rangle \) for an \( 1/2 + \epsilon \) fraction of the \( r \). Our goal will be to use \( H() \) to simulate an oracle that has agreement \( 7/8 \) with \( \langle x, r \rangle \), so that we can use the algorithm of Lemma 2 the previous section to find \( x \).

We perform this “reduction” by “guessing” the value of \( \langle x, r \rangle \) at a few points.

We first choose \( k \) random points \( r_1 \ldots r_k \in \{0,1\}^n \) where \( k = O(1/\epsilon^2) \). For the moment, let us suppose that we have “magically” obtained the values \( \langle x, r_1 \rangle, \ldots, \langle x, r_k \rangle \).

Then define \( H'(r) \) as the majority value of:

\[ H(r + r_j) - \langle x, r_j \rangle \quad j = 1, 2, \ldots, k \] (5)

For each \( j \), the above expression equals \( \langle x, r \rangle \) with probability at least \( \frac{1}{2} + \epsilon \) (over the choices of \( r_j \)) and by choosing \( k = O(1/\epsilon^2) \) we can ensure that

\[ \mathbb{P}_{r, r_1, \ldots, r_k} [H'(r) = \langle x, r \rangle] \geq \frac{31}{32}. \] (6)

from which it follows that

\[ \mathbb{P}_{r_1, \ldots, r_k} \left[ \mathbb{P}_r [H'(r) = \langle x, r \rangle] \geq \frac{7}{8} \right] \geq \frac{3}{4}. \] (7)

Consider the following algorithm.

function GL-FIRST-ATTEMPT
    pick \( r_1, \ldots, r_k \in \{0,1\}^n \) where \( k = O(1/\epsilon^2) \)
    for all \( b_1, \ldots, b_k \in \{0,1\} \) do
define $H'_{b_1...b_k}(r)$ as majority of: $H(r + r_j) - b_j$
apply Algorithm GLW to $H'_{b_1...b_t}$
add result to list
end for
return list
end function

The idea behind this program is that we do not in fact know the values $\langle x,r_j \rangle$, but we can “guess” them by considering all choices for the bits $b_j$. If $H(r)$ agrees with $\langle x,r \rangle$ for at least a $1/2 + \epsilon$ fraction of the $r$s, then there is a probability at least $3/4$ that in one of the iteration we invoke algorithm GLW with a simulated oracle that has agreement $7/8$ with $\langle x,r \rangle$. Therefore, the final list contains $x$ with probability at least $3/4 - 1/n > 1/2$.

The obvious problem with this algorithm is that its running time is exponential in $k = O(1/\epsilon^2)$ and the resulting list may also be exponentially larger than the $O(1/\epsilon^2)$ bound promised by the Lemma.

To overcome these problems, consider the following similar algorithm.

function GL
pick $r_1,\ldots,r_t \in \{0,1\}^n$ where $t = \log O(1/\epsilon^2)$

define $r_S := \sum_{j \in S} r_j$ for each non-empty $S \subseteq \{1,\ldots,t\}$

for all $b_1,\ldots,b_t \in \{0,1\}$ do
define $b_S := \sum_{j \in S} b_j$ for each non-empty $S \subseteq \{1,\ldots,t\}$

define $H'_{b_1...b_t}(r)$ as majority over non-empty $S \subseteq \{1,\ldots,t\}$ of $H(r + r_S) - b_S$
run Algorithm GLW with oracle $H'_{b_1...b_t}$
add result to list
end for
return list
end function

Let us now see why this algorithm works. First we define, for any nonempty $S \subseteq \{1,\ldots,t\}$, $r_S = \sum_{j \in S} r_j$. Then, since $r_1,\ldots,r_t \in \{0,1\}^n$ are random, it follows that for any $S \neq T$, $r_S$ and $r_T$ are independent and uniformly distributed. Now consider an $x$ such that $\langle x,r \rangle$ and $H(r)$ agree on a $\frac{1}{2} + \epsilon$ fraction of the values of $r$. Then for the choice of $\{b_j\}$ where $b_j = \langle x,r_j \rangle$ for all $j$, we have that

$$b_S = \langle x,r_S \rangle$$
for every non-empty $S$. In such a case, for every $S$ and every $r$, there is a probability at least $\frac{1}{2} + \epsilon$, over the choices of the $r_j$ that

$$H(r + r_S) - b_S = \langle x, r \rangle,$$

and these events are pair-wise independent. Note the following simple lemma.

**Lemma 4** Let $R_1, \ldots, R_k$ be a set of pairwise independent $0-1$ random variables, each of which is 1 with probability at least $\frac{1}{2} + \epsilon$. Then $\Pr[\sum_i R_i \geq k/2] \geq 1 - \frac{1}{4\epsilon^2 k}$.

**Proof:** Let $R = R_1 + \cdots + R_k$. The variance of a 0/1 random variable is at most 1/4, and, because of pairwise independence, $\text{Var}[R] = \text{Var}[R_1 + \cdots + R_k] = \sum_i \text{Var}[R_i] \leq k/4$.

We then have

$$\Pr[R \leq k/2] \leq \Pr[|R - \mathbb{E}[R]| \geq \epsilon k] \leq \frac{\text{Var}[R]}{\epsilon^2 k^2} \leq \frac{1}{4\epsilon^2 k}.$$

□

Lemma 4 allows us to upper-bound the probability that the majority operation used to compute $H'$ gives the wrong answer. Combining this with our earlier observation that the $\{r_S\}$ are pairwise independent, we see that choosing $t = \log(128/\epsilon^2)$ suffices to ensure that $H'_{b_1 \ldots b_t}(r)$ and $\langle x, r \rangle$ have agreement at least $7/8$ with probability at least $3/4$. Thus we can use Algorithm $A_5$ to obtain $x$ with high probability. Choosing $t$ as above ensures that the list generated is of length at most $2^t = 128/\epsilon^2$ and the running time is then $O(n^2 \epsilon^{-4} \log n)$ with $O(n \epsilon^{-4} \log n)$ oracle accesses, due to the $O(1/\epsilon^2)$ iterations of Algorithm GLW, that makes $O(n \log n)$ oracle accesses, and to the fact that one evaluation of $H'$ requires $O(1/\epsilon^2)$ evaluations of $H()$. 

5