Notes for Lecture 14 v0.9

These notes are in a draft version. Please give me any comments you may have, because this will help me revise them.

Today we will prove the Goldreich-Levin theorem:

Theorem 1 *If $f : \{0,1\}^n \to \{0,1\}^n$ is a one-way permutation then $B(x,r) = x \cdot r = \sum_i x_i r_i \pmod 2$ is a hardcore predicate for $f'(x,r) = f(x), r$.*

In other words, given only $f(x)$ and r it is hard to compute $x \cdot r$.

The asymptotic version of the Theorem follows from the following finite version, that we state directly in the counterpositive direction.

Theorem 2 *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a bijection computable by a circuit of size t, and suppose that there is a circuit C of size S such that*

$$\mathbf{Pr}_{x,r}[C(f(x),r) = x \cdot r] \geq \frac{1}{2} + \epsilon$$

Then there is a circuit C' of size $O((S+t) \cdot poly(n, 1/\epsilon))$ such that

$$\mathbf{Pr}_x[C'(f(x)) = x] \geq \frac{\epsilon}{4}$$

In turn, Theorem 2 follows from the existence of the following algorithm.

Lemma 3 *There is a probabilistic algorithm that, given a parameter ϵ and oracle access to a function $g : \{0,1\}^n \to \{0,1\}$, runs in time $O(n^2 \epsilon^{-4} \log n)$, makes $O(n\epsilon^{-4} \log n)$ oracle accesses, and outputs a list of $O(1/\epsilon^2)$ elements of $\{0,1\}^n$.*

If x is such that $\mathbf{Pr}_r[B(r) = x \cdot r] \geq \frac{1}{2} + \epsilon$, then there is a probability at least $1/2$ that x is in the output list.

1 Proof of Theorem 2 Using Lemma 3

From the assumption that

$$\mathbf{Pr}_{x,r}[C(f(x),r) = x \cdot r] \geq \frac{1}{2} + \epsilon$$

it follows that

$$\mathbf{Pr}_x\left[\mathbf{Pr}_r[C(f(x),r) = x \cdot r] \geq \frac{1}{2} + \frac{\epsilon}{2}\right] \geq \frac{\epsilon}{2}$$

Let us call an x such that $\mathbf{Pr}_r[C(f(x),r) = x \cdot r] \geq 1/2 + \epsilon/2$ a *good* x, and, for a given x, let us denote by B_x the function defined as $B_x(r) = C(f(x),r)$.

For a good x, if we apply the algorithm of Lemma 3 to the function B_x with parameter $\epsilon/2$, we obtain a list that, with probability at least $1/2$, contains x.

Consider now the following algorithm: given a string y, define $B(r) := C(y, r)$ and run the algorithm of Lemma 3 with function $B()$ and parameter $\epsilon/2$. Once the algorithm outputs a list $x^1, x^2, \ldots, x^{O(1/\epsilon^2)}$, compute $f(x^i)$ for each i, and output the x^i such that $f(x^i) = y$, if any.

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. In particular, there is a fixed choice of the internal randomness of the algorithm that results in inverting $f()$ on an $\epsilon/4$ fraction of the inputs. Finally, we convert the algorithm into a circuit, and the resulting circuit is C'.

2 Proof of Lemma 3

2.1 A Special Case First

We start by considering the simpler case in which we are given oracle access to a function $B()$ such that $\mathbf{Pr}_r[B(r) = x \cdot r] \geq 7/8$ and we want to find x.

If we denote by e_i the vector that has a 1 in the i-th coordinate and 0s in other coordinate, we see that $x_i = x \cdot e_i$. Furthermore, for every r, we have $x_i = x \cdot (r \oplus e_i) \oplus x \cdot r$ because of the linearity of the $\cdot$ operator and of the fact that $r \oplus r$ is the all zero vector.

Consider now the process of picking a random r and computing $B(r \oplus e_i) \oplus B(r)$. Except with probability at most $1/8$, $B(r \oplus e_i) = x \cdot (r \oplus e_i)$ and, except with probability at most $1/8$, $B(r) = x \cdot r$. Therefore, with probability at least $3/4$,

$$B(r \oplus e_i) \oplus B(r) = x \cdot (r \oplus e_i) \oplus x \cdot r = x_i$$

This suggests the following algorithm:

Algorithm $A_{\frac{7}{8}}$**:**
  for $i := 1$ to n **do**
    pick $k = O(\log n)$ random elements $r^1, \ldots, r^k \in \{0,1\}^n$
    compute:
      $B(r^1 \oplus e_1) \oplus B(r^1)$
      $B(r^2 \oplus e_2) \oplus B(r^2)$
      $\vdots$
      $B(r^k \oplus e_n) \oplus g(r^n)$
    assign to x_i the value occurring in the majority of these computations
  return x

To analyze the algorithm, note that, for a particular value of i, we expect to get the right value of x_i in a fraction $3/4$ of the k trials, and the algorithm derives the correct value of

x_i provided that more than half of the k trials are correct. Then, by a Chernoff bound, the probability of estimating x_i incorrectly is $e^{-\Omega(k)}$. We can then choose $k = O(\log n)$ and make sure that the error probability is at most, say, $1/100n$, and hence we conclude that the output of the algorithm is correcgt with probability at least $99/100$.

We note that the running time of this program is $O(n^2 k) = O(n^2 \log n)$ and that it makes $O(nk) = O(n \log n)$ oracle accesses.

2.2 The General Case

Consider now the general case. We are given an oracle $B()$ such that $B(r) = x \cdot r$ for an $1/2 + \epsilon$ fraction of the r. Our goal will be to use $B()$ to simulate an oracle that has agreement $7/8$ with $x \cdot r$, so that we can use the algorithm of the previous section to find x. We perform this "reduction" by "guessing" the value of $x \cdot r$ at a few points.

We first choose t random points $r^1 \dots r^t \in \{0,1\}^n$ where $t = O(1/\epsilon^2)$. For the moment, let us suppose that we have "magically" obtained the values $x \cdot r^1, \dots, x \cdot r^k$. Then define $B'(r)$ as the majority value of:

$$x \cdot r^j \oplus B(r \oplus r^j) \qquad j = 1, 2, \dots, t \tag{1}$$

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

$$\mathbf{Pr}_{r, r^1, \dots, r^t} \left[B'(r) = x \cdot r \right] \geq \frac{31}{32}. \tag{2}$$

from which it follows that

$$\mathbf{Pr}_{r^1, \dots, r^k} \left[\mathbf{Pr}_r \left[B'(r) = x \cdot r \right] \geq \frac{7}{8} \right] \geq \frac{3}{4}. \tag{3}$$

Consider the following algorithm.

Algorithm GL-First-Attempt:
  pick $r^1, \dots, r^t \in \{0,1\}^k$ where $t = O(1/\epsilon^2)$
  for all $b_1, \dots, b_t \in \{0,1\}$
    define $B'_{b_1 \dots b_t}(r)$ as majority of: $b_j \oplus B(r \oplus r^j)$
    apply Algorithm $A_{\frac{7}{8}}$ to $B'_{b_1 \dots b_t}$
    add result to list

The idea behind this program is that we do not in fact know the values $x \cdot r^j$, but we can "guess" them by considering all choices for the bits b_j. If $B(r)$ agrees with $x \cdot r$ for at least a $1/2 + \epsilon$ fraction of the rs, then there is a probability at least $3/4$ that in one of the iteration we invoke algorithm $A_{\frac{7}{8}}$ with a simulated oracle that has agreement $7/8$ with $x \cdot r$. Therefore, the final list contains x with probability at least $3/4 - 1/100 > 1/2$.

The obvious problem with this algorithm is that its running time is exponential in $t = O(1/\epsilon^2)$ 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.

Algorithm GL:

pick $r^1, \ldots, r^l \in \{0,1\}^k$ where $l = \log O(1/\epsilon^2)$

define $r_S := \bigoplus_{j \in S} r^j$ for each non-empty $S \subseteq \{1, \ldots, l\}$

for all $b_1, \ldots, b_l \in \{0,1\}$

  define $b_S := \bigoplus_{j \in S} b_j$ for each non-empty $S \subseteq \{1, \ldots, l\}$

  define $B'_{b_1 \ldots b_l}(r)$ as majority over non-empty $S \subseteq \{1, \ldots, l\}$ of $b_S \oplus B(r \oplus r_S)$

  run Algorithm $A_{\frac{7}{8}}$ with oracle $B'_{b_1 \ldots b_l}$

  add result to list

Let us now see why this algorithm works. First we define, for any nonempty $S \subseteq \{1, \ldots, l\}$, $r_S = \bigoplus_{j \in S} r^j$. Then, since $r^1, \ldots, r^l \in \{0,1\}^k$ 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 $x \cdot r$ and $B(r)$ agree on a $\frac{1}{2} + \epsilon$ fraction of the values of r. Then for the choice of $\{b_j\}$ where $b_j = x \cdot r^j$ for all j, we have that

$$b_S = x \cdot r_S$$

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

$$b_S \oplus B(r \oplus r_S) = x \cdot r \ ,$$

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

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

PROOF: Let $R = R_1 + \cdots + R_t$. The variance of a $0/1$ random variable is at most $1/4$, and, because of pairwise independence, $\mathbf{Var}[R] = \mathbf{Var}[R_1 + \ldots + R_t] = \sum_i \mathbf{Var}[R_i] \leq t/4$.

We then have

$$\mathbf{Pr}[R \leq t/2] \leq \mathbf{Pr}[|R - \mathbf{E}[R]| \geq \epsilon t] \leq \frac{\mathbf{Var}[R]}{\epsilon^2 t^2} \leq \frac{1}{4\epsilon^2 t}$$

□

Lemma 4 allows us to upper-bound the probability that the majority operation used to compute B' gives the wrong answer. Combining this with our earlier observation that the $\{r_S\}$ are pairwise independent, we see that choosing $l = 2\log 1/\epsilon + O(1)$ suffices to ensure that $B'_{b_1 \ldots b_l}(r)$ and $x \cdot r$ have agreement at least $7/8$ with probability at least $3/4$. Thus we can use Algorithm $A_{\frac{7}{8}}$ to obtain x with high probability. Choosing l as above ensures that the list generated is of length at most $2^l = O(1/\epsilon^2)$ and the running time is then $O(\epsilon^{-4} \cdot n^2 \log n)$ with $O(\epsilon^{-4} \cdot n \log n)$ oracle accesses, due to the $O(1/\epsilon^2)$ iterations of Algorithm $A_{\frac{7}{8}}$, that makes $O(n \log n)$ oracle accesses, and to the fact that one evaluation of $B'()$ requires $O(1/\epsilon^2)$ evaluations of $B()$.

3 References

The results of this lecture are from [GL89]. Goldreich and Levin initially presented a different proof. They credit the proof with pairwise independence to Rackoff. Algorithm $A_{\frac{7}{8}}$ is due to Blum, Luby and Rubinfeld [BLR93]. The use of Algorithm GL-First-Attempt as a motivating example might be new to these notes. (Or, actually, to the Fall 2001 notes for this class.)

References

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. Preliminary version in *Proc. of STOC'90*. 5

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 25–32, 1989. 5