Class 15 Exercises

CS250/EE387, Winter 2022

In the lecture videos/notes, we saw *local list decoding,* and an algorithm to locally list-decode the Hadamard code. We saw one example (to learning Fourier-sparse functions) in the lecture videos, and today we'll see another application: *hardcore predicates from one-way-functions.*

Definition 1 (Hard-core bit)**.** *Let $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ be a function. We say that $b : \mathbb{F}_2^k \to \mathbb{F}_2$ is a* hard-core bit *for f if:*

- *It is computationally efficient to compute b.*
- *Given $f(x)$, it is hard to determine $b(x)$. Formally, for any randomized algorithm $\mathcal{A}$ that runs in time polynomial in k, and for any function $\varepsilon(k)$ that tends to zero polynomially fast in k,*

$$\Pr_{x \sim \mathbb{F}_2^k}[\mathcal{A}(f(x)) = b(x)] \leq \frac{1}{2} + \varepsilon(k).$$

(The probability is over both the choice of x and any randomness in $\mathcal{A}$).

Why do we care about hard-core bits? Briefly, these come up in designing *pseudorandom generators* (PRGs). Informally, a (cryptographic) PRG is a function $\varphi : \mathbb{F}_2^k \to \mathbb{F}_2^{k'}$ so that $k' > k$ that takes a random seed $x \in \mathbb{F}_2^k$ and outputs a longer pseudorandom string in $\mathbb{F}_2^{k'}$. Here, what "pseudorandom" means is that no polynomial-time algorithm can distinguish it from uniformly random with non-negligable advantage. A classical result is cryptography is that if one-way permutations[1] exist, then so do PRGs. The proof uses hardcore bits. Here's the basic idea. Let f be a one-way permutaiton. If we have a hardcore bit b for f, then consider the function ϕ given by $\phi(x) = f(x) \circ b(x)$, where $\circ$ denotes concatenation. The output of ϕ is one bit longer than the input (hooray, this is a win!). Further, if $x \sim \mathbb{F}_2^k$ is random, then anyone who could distinguish $f(x) \circ b(x)$ from a uniformly random string would also have some advantage at predicting $b(x)$ from $f(x)$, violating the hard-core property. (See this survey for more about PRGs: https://www.wisdom.weizmann.ac.il/~oded/prg-primer.html.)

But does such a function $b(x)$ exist for our one way permutation $f(x)$? Fortunately, it turns out that as long as there exists some one-way permutation to use in the application above, it's possible to modify that function so that it has a (simple) hard-core bit!

[1] Informally, a one-way permutation is a permutation $f : \{0,1\}^k \to \{0,1\}^k$ so that $f(x)$ is easy to compute, but $f^{-1}(y)$ is difficult to compute.

Theorem 1 (Goldreich-Levin Theorem)**.** *Suppose that $f : \mathbb{F}_2^k \to \mathbb{F}_2^k$ is a one-way permutation, meaning that for any randomized algorithm $\mathcal{A}$ that runs in time polynomial in k, and for any function $\varepsilon(k)$ that tends to zero polynomially fast in k, $\Pr_{x\sim\mathbb{F}_2^k}[\mathcal{A}(f(x)) = x] \leq \varepsilon(k)$.*

Consider the function $g : \mathbb{F}_2^{2k} \to \mathbb{F}_2^{2k}$ given by

$$g(x, r) = f(x) \circ r,$$

where $\circ$ denotes concatenation. Then $g(x, r)$ is a one-way permutation, and

$$b(x, r) = \langle x, r\rangle$$

is a hard-core bit for g.

(You check now that $g(x, r)$ is indeed a OWP if $f(x)$ is).

We'll prove this theorem by contradiction: suppose that b is *not* hard-core. Then there is some efficient algorithm $\mathcal{A}$ that can predict $b(x, r)$ given $f(x, r)$. We will use $\mathcal{A}$ as a black box to build an efficient algorithm $\mathcal{B}$ that inverts f. But since f was supposed to be a one-way permutation, this will be a contradiction!

1. Suppose that $\mathcal{A}$ is an efficient algorithm so that

$$\Pr_{x,r\sim\mathbb{F}_2^k}[\mathcal{A}(g(x, r)) = \langle x, r\rangle] = 1.$$

   Give an efficient algorithm $\mathcal{B}$ so that

$$\Pr_{x\sim\mathbb{F}_2^k}[\mathcal{B}(f(x)) = x] = 1.$$

   (Above and throughout, the probabilities are also over the randomness of $\mathcal{A}, \mathcal{B}$).

   Solution

   The algorithm $\mathcal{B}$ is:

   - For $i = 1, \ldots, k$, set $x_i \leftarrow \mathcal{A}(f(x) \circ e_i)$.
   - Return $(x_1, \ldots, x_k)$.

   This works since $\mathcal{A}(f(x) \circ e_i) = \mathcal{A}(g(x, e_i)) = \langle x, e_i\rangle = x_i$.

2. Suppose that $\mathcal{A}$ is an efficient algorithm so that

$$\Pr_{x,r\sim\mathbb{F}_2^k}[\mathcal{A}(g(x, r)) = \langle x, r\rangle] \geq 3/4 + \varepsilon.$$

   Give an efficient algorithm $\mathcal{B}$ so that

$$\Pr_{x\sim\mathbb{F}_2^k}[\mathcal{B}(f(x)) = x] \geq \varepsilon'$$

   for some constant ε' (that can depend on ε).

Solution

Essentially, the algorithm $\mathcal{A}$ is giving us access to a noisy Hadamard codeword. That is, using $\mathcal{A}$ we can obtain z_r so that $z_r = \langle x, r\rangle = c_r$ for at least $3/4 + \varepsilon$ fraction of the positions r. Thus, we can use a unique decoding algorithm for the Hadamard code in order to recover x.

In more detail, let $T = O(\log(k)/\varepsilon^2)$. The algorithm $\mathcal{B}$ is:

- **Input**: $f(x)$
- For $i = 1, \ldots, k$:
  - For $t = 1, \ldots, T$:
    - Draw a random $r_i^{(t)} \sim \mathbb{F}_2^k$.
    - $x_i^{(t)} \leftarrow \mathcal{A}(f(x) \circ r_i^{(t)}) \oplus \mathcal{A}(f(x) \circ (r_i^{(t)} + e_i))$
  - Let x_i be the majority vote of $\{x_i^{(t)}\}_{t\in[T]}$.
- Return $(x_1, \ldots, x_k)$

To analyze this algorithm, notice that the probability that $\mathcal{A}(f(x) \circ r_i^{(t)}) = \left\langle x, r_i^{(t)} \right\rangle$ *and* $\mathcal{A}(f(x) \circ r_i^{(t)} + e_i) = \left\langle x, r_i^{(t)} + e_i \right\rangle$ is at least $1/2 + 2\varepsilon$, by the union bound. If that happens, the $x_i^{(t)} = x_i$. Thus, the probability that the majority-vote fails is the probability that the sum of T Bernoulli-$1/2 + 2\varepsilon$ random variables is less than $T/2$. By a Chernoff bound, this is at most $\exp(-\Omega(\varepsilon^2 T)) = 1/\mathrm{poly}(k)$, so we can choose the constants so that we can take a union bound over all values of $i \in [k]$, and this succeeds with very high probability.

3. Suppose that $\mathcal{A}$ is an efficient algorithm so that

$$\Pr_{x,r\sim\mathbb{F}_2^k}[\mathcal{A}(g(x, r)) = \langle x, r\rangle] \geq 1/2 + \varepsilon.$$

Give an efficient algorithm $\mathcal{B}$ so that

$$\Pr_{x\sim\mathbb{F}_2^k}[\mathcal{B}(f(x)) = x] \geq \varepsilon'$$

for some ε' that may depend on ε.

(Note: your answer may be of the form "use the algorithm for XXXX that we've seen in the mini-lectures." That's fine, but if you have time, try to write down what the algorithm would look like in this language. The main point is to make you go back over the algorithm and remember what it's doing.)

Solution

Now we can use the local-list-decoding algorithm for Hadamard codes! Let List-Decode be that algorithm (or rather, what we get when we run that algorithm k times, once for each of the message bits). Our new algorithm is:

- **Input:** $f(x)$.

- Run List-Decode on $z \in \mathbb{F}_2^{2^k}$ with query access given by

$$z_r = \mathcal{A}(f(x) \circ r).$$

  Get a list $\mathcal{S} = \{x^{(1)}, \ldots, x^{(L)}\} \subseteq \mathbb{F}_2^k$ of possible messages.
- For each $i = 1, \ldots, L$, compute $f(x^{(i)})$. If it is equal to $f(x)$, return $x^{(i)}$.