**Analysis of Boolean Functions** **(CMU 18-859S, Spring 2007)**

# Lecture 7: The Goldreich-Levin Algorithm

Feb. 6, 2007

*Lecturer: Ryan O'Donnell* *Scribe: Karl Wimmer*

In this lecture we make the jump from testing properties of functions to learning functions. The first algorithm that we will see is an algorithm of Goldreich and Levin, which was originally developed for a cryptographic application and later applied to learning. This algorithm works by finding a function's large Fourier coefficients.

## 1 Testing vs. Learning

We first set some goals for learning. First, how do we measure complexity? In testing, we were concerned with how many queries were required. In contrast, learning is an algorithmic problem, so we will be concerned with the running time of the learning algorithm. Second, what is our task? In testing, we had a class of functions $C$, and given $f$, our task was to determine whether or not $f \in C$. In learning, we are promised that $f \in C$, and we are required to identify the function in $C$ that $f$ is.

## 2 Learning Parities

Our first task will be learning parities.

**Proposition 2.1** *If $f : \{-1,1\}^n \to \{-1,1\}$ is (0-close to) a parity function, then we can identify $f$ in polynomial time using $n$ queries.*

**Proof:** We simply query $f$ on all strings $e_i$ for $1 \leq i \leq n$, where the string $e_i$ is $1$ everywhere, except $-1$ in entry $i$. If $f(e_i) = -1$, then $x_i$ is relevant in the parity function $f$. $\square$

**Remark 2.2** *The above algorithm does not run in linear time, because it takes linear time to write down each query.*

So we can learn functions that are parities. With a little more work, we can use local decoding to learn functions close to some parity. Before we do this, we recall the Hoeffding bound:

**Theorem 2.3** *(Hoeffding bound) If $\mathbf{X}$ is a random variable with values in $[-1,1]$, then the empirical average of $\mathbf{X}$ after $O(\log \frac{1}{\delta}/\epsilon^2)$ samples is within $\pm\epsilon$ of $\mathbf{E}[\mathbf{X}]$ with probability $1-\delta$.*

**Proposition 2.4** *Suppose $f : \{-1,1\}^n \to \{-1,1\}$ is $\epsilon$-close to $\chi_S$ for some $S$, where $\epsilon < \frac{1}{4} - c$ for $c > 0$. Then we can identify $f$ with probability $1-\delta$ using $O(n \log \frac{n}{\delta}$ queries.*

**Proof:** Use local decoding on all $e_i$. The probability that the correct answer is returned for one $e_i$ is at least $\frac{1}{2}+2c$. For each string, repeat the local decoding step $O(\log(\frac{n}{\delta}/c^2)$ times, and take the majority answer. By the Hoeffding bound, we get the wrong answer for $f(e_i)$ with probability at most $\frac{\delta}{n}$. By the union bound, we get the wrong answer for $f(e_i)$ for any $i$ with probability at most $\delta$, so we succeed with probability $1-\delta$ as claimed. □

**Remark 2.5** *Notice that the above algorithm cannot be deterministic. For any deterministic version of local decoding, an adversary could choose a function $f$ where the above algorithm would not work.*

To learn functions $\epsilon$-far from every parity function for $\epsilon \geq \frac{1}{4}$, we need to introduce the Goldreich-Levin algorithm.

## 3 The Goldreich-Levin Theorem

Since we are talking about functions that are not $\frac{1}{4}$-close to any parity function, we will talk about correlation rather than closeness. We will say that $f$ has correlation $\gamma$ with $\chi_S$ if $|\hat{f}(S)| \geq \gamma$.

Notice that in the case where $\gamma$ is small, there are potentially many $S$ such that $f$ has correlation $\gamma$ with $\chi_S$. The following easy bound follows from Parseval's identity:

**Proposition 3.1** $|\hat{f}(S)| \geq \gamma$ *for at most* $\frac{1}{\gamma^2}$ *sets* $S$.

We will commonly think of $\gamma$ as a constant. We can think of the task of outputting all $S$ such that $|\hat{f}(S)| \geq \gamma$ as list decoding.

We know state the Goldreich-Levin theorem.

**Theorem 3.2** *(Goldreich-Levin) Given query access to* $f : \{-1,1\}^n \to [-1,1]$, *given* $\gamma$, $\delta > 0$, *there is a poly(*$n$*,*$\frac{1}{\gamma}\log\frac{1}{\delta}$*)-time algorithm outputs a list* $L = \{S_1, \ldots, S_m\}$ *such that (1) if* $\hat{f}(S) \geq \gamma$, *then* $S \in L$, *and (2) if* $S \in L$, *then* $\hat{f}(S) \geq \frac{\gamma}{2}$ *holds with probability* $1-\delta$.

The reason that this algorithm is useful is that a function is nearly determined by its large Fourier coefficients. As well, this theorem can be generalized to the case where the $f$ maps into $\mathbb{R}$. The running time of the algorithm is then dependent on the normal parameters as well as $\max_x(f(x)) - \min_x(f(x))$.

To prove this algorithm, we will introduce a powerful tool that we will use very often in the remainder of this course. This tool says that we can "efficiently estimate" any Fourier coefficient we wish.

**Lemma 3.3** *For any* $S \subseteq [n]$, *it is possible to estimate* $\hat{f}(S)$ *to within* $\pm\eta$ *with probability* $1-\delta$ *using* $O(\log(\frac{1}{\delta}/\eta^2)$ *queries, with an extra running time factor polynomial in* $n$.

**Proof:** By definition, $\hat{f}(S) = \mathbf{E}_{\boldsymbol{x}}[f(\boldsymbol{x})\chi_S(\boldsymbol{x})]$. We can sample from $f(\boldsymbol{x})\chi_S(\boldsymbol{x})$, and this random variable is in the range $[-1, 1]$. The result follows directly from an application of the Hoeffding bound. $\square$

**Corollary 3.4** *To prove the Goldreich-Levin theorem, it suffices to prove that (1) holds. To do this, we can estimate $\hat{f}(S)$ to within $\pm\frac{\gamma}{4}$ for every $S \in L$, then delete from $L$ all sets whose estimate has magnitude less than $\gamma/2$. The running time is proportional to the length of the list, and requires manipulating the confidence parameter $\delta$.*

## 4 The Goldreich-Levin Algorithm

Now we describe the algorithm promised by the Goldreich-Levin theorem. The idea is to think of $\hat{f}(S)^2 \in [0, 1]$ as the "weight" of the set $S$, and the weight of a collection of sets is simply the sum of weights of each set. By Parseval, the sum of the weights on the whole Boolean cube is 1.

Our goal is to find all sets $S$ with weight at least $\gamma^2$. To accomplish this, we will partition $2^{[n]}$ into chunks. We will repartition any chunk with weight $\gamma^2$, and discard any chunk with weight less than $\gamma^2$, as such a chunk can not contain a set we are looking for. Assuming we can do this, we only need to keep $\frac{1}{\gamma^2}$ chunks at any time, as the weight of the Boolean cube is 1.

**Definition 4.1** *Let $f : \{-1, 1\}^n \to \mathbb{R}$. Let $I \subseteq [n]$, and $\bar{I} = [n] \setminus I$. For any $S \subseteq I$, define $F_{S\subseteq I} : \{-1, 1\}^{|\bar{I}|} \to \mathbb{R}$ by $F_{S\subseteq I} = \hat{f}_{x\to\bar{I}}(S)$, where $f_{x\to\bar{I}}$ is the restricted version of $f$ resulting from fixing the coordinates in $\bar{I}$ to $x$.*

**Proposition 4.2** $\widehat{F_{S\subseteq I}}(T) = \hat{f}(S \cup T)$ *for any $T \subseteq \bar{I}$.*

**Proof:** By definition,

$$\widehat{F_{S\subseteq I}}(T) = \mathop{\mathbf{E}}_{\boldsymbol{x}}[F_{S\subseteq I}(\boldsymbol{x})\boldsymbol{x}_T] = \mathop{\mathbf{E}}_{\boldsymbol{x}}[\hat{f}_{\boldsymbol{x}\to\bar{I}}(S)\boldsymbol{x}_T] = \mathop{\mathbf{E}}_{\boldsymbol{x}}[\mathop{\mathbf{E}}_{\boldsymbol{w}}[f_{\boldsymbol{x}\to\bar{I}}(\boldsymbol{w})\boldsymbol{w}_S]\boldsymbol{x}_T] = \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{w}}[f(\boldsymbol{w},\boldsymbol{x})\boldsymbol{w}_S\boldsymbol{x}_T] = \hat{f}(S\cup T)$$

where $f(w, x)$ denotes $f$ with $x \to \bar{I}$ and $w \to I$. $\square$

**Corollary 4.3** $\mathbf{E}_{\boldsymbol{x}}[F_{S\subseteq I}(\boldsymbol{x})^2] = \sum_{T\subseteq\bar{I}} \widehat{F_{S\subseteq I}}(T)^2 = \sum_{T\subseteq\bar{I}} \hat{f}(S \cup T)^2 = \sum_{U\subseteq[n]:U\cap I=S} \hat{f}(U)^2.$

We will use the above corollary to construct our partitions to estimate weights on collections on sets. For $I = \{1, 2, \ldots, k\}$ and $S \subseteq I$, it is convenient to write $U$ such that $U \cap I = S$ as an "indicator string." For example, with $k = 4$, a possible indicator string is $(1, 0, 1, 1, *, *, \ldots, *)$.

**Definition 4.4** *The indicator string $\mathcal{S} = (a_1, \ldots, a_k, *, \ldots, *)$ is $\{U \subseteq [n] : \forall i = 1, \ldots, k, i \in U \text{ iff } a_i = 1\}$.*

**Definition 4.5** *The weight of an indicator string $\mathcal{S}$ is $\sum_{U \in \mathcal{S}} \hat{f}(U)^2$. We will denote this quantity by $W(\mathcal{S})$.*

We will make use of the previous corollary to get our second important tool: we can efficiently estimate $W(\mathcal{S})$. We do this by efficiently estimating $\mathbf{E}_{\boldsymbol{x}}[F_{S\subseteq I}(\boldsymbol{x})^2]$.

**Proposition 4.6** $\mathbf{E}_{\boldsymbol{x}}[F_{S\subseteq I}(\boldsymbol{x})^2]$ *can be efficiently estimated.*

**Proof:**

$$\mathop{\mathbf{E}}_{\boldsymbol{x}}[F_{S\subseteq I}(\boldsymbol{x})^2] = \mathop{\mathbf{E}}_{\boldsymbol{x}}[\hat{f}_{\boldsymbol{x}\to\bar{I}}(S)^2] = \mathop{\mathbf{E}}_{\boldsymbol{x}}[(\mathop{\mathbf{E}}_{\boldsymbol{w}}[f_{\boldsymbol{x}\to\bar{I}}(\boldsymbol{w})\boldsymbol{w}_S])^2].$$

Now instead of taking a square of a expectation, we pick two independent copies of $w$ and write a product of expectations to yield:

$$\mathop{\mathbf{E}}_{\boldsymbol{x}}[(\mathop{\mathbf{E}}_{\boldsymbol{w}}[f_{\boldsymbol{x}\to\bar{I}}(\boldsymbol{w})\boldsymbol{w}_S])^2] = \mathop{\mathbf{E}}_{\boldsymbol{x}}[\mathop{\mathbf{E}}_{\boldsymbol{w},\boldsymbol{w}'}[f_{\boldsymbol{x}\to\bar{I}}(\boldsymbol{w})\boldsymbol{w}_S f_{\boldsymbol{x}\to\bar{I}}(\boldsymbol{w}')\boldsymbol{w}'_S] = \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{w},\boldsymbol{w}'}[f(\boldsymbol{w},\boldsymbol{x})\boldsymbol{w}_S f(\boldsymbol{w}',\boldsymbol{x})\boldsymbol{w}'_S].$$

Where the first equality used the independence of $w$ and $w'$. Since the random variable inside the expectation can be sampled from and has the range $[-1,1]$, we can use the Hoeffding bound to estimate this expectation. $\square$ Equipped with this tool, we can state the Goldreich-Levin algorithm:

**Algorithm 4.1:** GOLDREICHLEVIN($f$)

```
$L \leftarrow (*, *, \ldots, *)$
for $k = 1$ to $n$
    for each $\mathcal{S} \in L, \mathcal{S} = (a_1, \ldots, a_{k-1}, *, \ldots, *)$
        let $\mathcal{S}_{a_k} = (a_1, \ldots, a_{k-1}, a_k, *, \ldots, *)$ for $a_k = 0, 1$
        estimate $W(\mathcal{S}_{a_k})$ to within $\pm\gamma^2/4$ with probability at least $1 - \delta$
        remove $\mathcal{S}$ from $L$
        add $\mathcal{S}_{a_k}$ to $L$ if the estimate of $W(\mathcal{S}_{a_k})$ is at least $\frac{\gamma^2}{2}$ for $a_k = 0, 1$
return $(L)$
```

We will now analyze the algorithm. As a first assumption, we will assume that all estimations are accurate. We will later see how to remove this assumption.

**Invariant 4.7** *After 1 iteration of the algorithm, $W(\mathcal{S}) \geq \frac{\gamma^2}{4}$ for all $\mathcal{S} \in L$.*

**Proof:** All estimates are assumed to be correct, and for all $\mathcal{S} \in L$, $\mathcal{S}$ was placed in $L$ because its estimated weight was at least $\frac{\gamma^2}{2}$, and the estimate is correct to within an additive $\frac{\gamma^2}{4}$. $\square$

**Invariant 4.8** *At any time, $|L| \geq \frac{4}{\gamma^2}$. This follows from our previous observation combined with Parseval's identity.*

Since $|L| \geq \frac{4}{\gamma^2}$, the algorithm performs at most 2 estimations per set in $L$ at any iteration, and there are $n$ iterations, the algorithm performs a total of at most $\frac{8n}{\gamma^2}$ estimations.

**Invariant 4.9** *For any $S$ such that $\hat{f}(S) \leq \gamma^2$, there exists $\mathcal{S} \in L$ such that $S \in \mathcal{S}$. This follows from the correctness of our estimations.*

From the above invariants, we can conclude that our algorithm is correct.

To remove our assumption, given $\delta > 0$, we will define $\delta' = \frac{\delta}{8n/\gamma^2}$, and perform each estimation with confidence $1 - \delta'$. By the union bound, if the algorithm performs $\frac{8n}{\gamma^2}$ estimations, they are all correct with probability at least $1 - \delta$, so the algorithm is correct with probability at least $1 - \delta$.

The total running time is dominated by the estimations. There are at most $\frac{8n}{\gamma^2}$ estimations, and each takes $O(\log(\frac{1}{\delta}/\gamma^2)$ samples to estimate, so the overall running time is poly($n$,$\frac{1}{\gamma}$) $\log(\frac{1}{\delta})$.

Notice that, since we are estimating the weight of each set before it is added to $L$, we get property (2) from the Goldreich-Levin theorem for free from this algorithm.

## 5 Applications to cryptography

We will discuss some of the original applications of the Goldreich-Levin theorem to cryptography. We will require a few definitions.

**Definition 5.1** *$f : \{0,1\}^n \to \{0,1\}$ is a $\gamma(n)$-one way permutation if (1) $f$ is a permutation, (2) $f$ is deterministic poly-time computable, and (3) for any probabilistic poly-time algorithm $A$, $\mathbf{Pr}_{\boldsymbol{x},A'\text{s randomness}}[A(f(\boldsymbol{x})) = \boldsymbol{x}] < \gamma(n)$.*

**Remark 5.2** *In most applications, it is desired to have a family of one way permutations, one defined for each input length.*

In the definition, we can replace "permutation" with other terms, such as the more general "function" or the more specific "trapdoor permutation," where a trapdoor permutation is a permutation where there exists a short piece of information that allows for easy inversion of $f$.

**Example 5.3** *(RSA cryptosystem) Pick $N$ to be the product of two large randomly chosen primes $p$,$q$. Pick a random $e$ from $\mathbb{Z}_N^*$, the group of integers relatively prime to $N$ under mod-$N$ multiplication. Then $x \to x^e$ is a trapdoor permutation of $\mathbb{Z}_N^*$.*

**Remark 5.4** *Although elements of $\mathbb{Z}_N^*$ are not strings, we can massage them to be strings without too much difficulty.*

**Remark 5.5** *The factorization of $N$ is a trapdoor for the above example.*

**Definition 5.6** *A poly-time computable function $B : \{0,1\}^n \to \{0,1\}$ is a $\gamma(n)$-hardcore predicate for $f$ if for all probabilistic polynomial time algorithms $A$, $\mathbf{Pr}_{\boldsymbol{x},A'\text{s randomness}}[A(f(\boldsymbol{x})) = B(\boldsymbol{x})] < \frac{1}{2} + \gamma(n)$.*

**Remark 5.7** *A one way permutation combined with a hardcore predicate for it can be used to construct a pseudorandom generator with stretch $n \to n+1$ by stretching $x$ to $(f(x), B(x))$. This idea can be repeated to obtain arbitrary stretch.*

We can now state the Goldreich-Levin theorem in this context.

**Theorem 5.8** *Let $f$ be a $\gamma(n)$-one way permutation (or function, or trapdoor permutation, etc.) for $\gamma(n) \geq \frac{1}{poly(n)}$. Then (1) $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ defined by $g(x,r) = (f(x), r)$ is also a $\gamma(n)$-one way permutation, and (2) $B(x,r) = x \cdot r(\mod 2)$ is a $3\gamma(n)$-hardcore predicate for $g$.*

**Proof:** The proof of (1) is easy. Any advantage in inverting $g$ directly implies an advantage in inverting $f$.

We will prove (2) by contraposition. We will do this under the assumption that $A$ is deterministic. poly-time algorithm with an advantage for computing $B$. We can use $A$ to get an algorithm with an advantage for inverting $f$. By assumption, $\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{r}}[A(f(\boldsymbol{x})) = \boldsymbol{x} \cdot \boldsymbol{r}] \geq \frac{1}{2} + 3\gamma(n)$. By averaging, for at least a $2\gamma$ fraction of the $x$'s, $\mathbf{Pr}_{\boldsymbol{r}}[A(f(x)) = x \cdot \boldsymbol{r}] \geq \frac{1}{2} + \gamma(n)$, so $A(f(x), \cdot)$ (is a function of $r$ that) is $2\gamma$-correlated with the parity function associated with $x$. We can invoke the Goldreich-Levin algorithm on $A(f(x), \cdot)$. We get a list of possible parities, one of which is $x$. Because the size of our list is small when the algorithm succeeds, we can determine $x$ by applying $f$ to everything in the list. So with high probability, this algorithm finds preimages of $f(x)$ for at least a $\gamma$ fraction of the $x$'s, so $f$ is not a $\gamma(n)$-one way permutation.

If we want to rid ourselves of the assmuption that $A$ is deterministic, let $\mathcal{A}$ be the function of $r$ equal to $\mathbf{E}_{A'\text{s randomness}}[A(f(x), r)]$, where the expectation is over $A$'s randomness. Now $\mathcal{A}$ is a function whose range is $[-1, 1]$, so we can try to apply Goldreich-Levin to it. However, in applying Goldreich-Levin, we need to have access to $\mathcal{A}$. We can circumvent this difficulty by sampling from $A$, and our proof stays the same modulo taking expectation over the randomness of $A$. $\Box$