Locally Covert Learning

Justin Holmgren* Ruta Jawale†

Abstract

The goal of a covert learning algorithm is to learn a function f by querying it, while ensuring that an adversary, who sees all queries and their responses, is unable to (efficiently) learn any more about f than they could learn from random input-output pairs. We focus on a relaxation that we call *local covertness*, in which queries are distributed across k servers and we only limit what is learnable by $k-1$ colluding servers.

For any constant k, we give a locally covert algorithm for efficiently learning any Fourier-sparse function (technically, our notion of learning is improper, agnostic, and with respect to the uniform distribution). Our result holds unconditionally and for computationally unbounded adversaries. Prior to our work, such an algorithm was known only for the special case of $O(\log n)$-juntas, and only with $k = 2$ servers [IKOS19].

Our main technical observation is that the original Goldreich-Levin algorithm only utilizes i.i.d. pairs of correlated queries, where each half of every pair is uniformly random. We give a simple generalization of this algorithm in which pairs are replaced by k-tuples in which any $k-1$ components are jointly uniform. The cost of this generalization is that the number of queries needed grows exponentially with k.

1 Introduction

In CRYPTO 2019, [IKOS19] formulated a new learning task whose utility is illustrated by the following scenario. Company A wishes to outsource experiments to company B while deterring employees of B from selling the outcomes of these experiments to a competitor C. Motivated by this scenario and others like it, they asked the following question (parameterized by a function family $\mathcal{F}$):

> *Can one learn*[1] *a function $f \in \mathcal{F}$ with oracle queries such that an adversary, who sees all queries x along with the corresponding values $f(x)$, is unable to learn f?*

When might this be possible? The adversary necessarily learns some non-trivial information about f, namely pairs of the form $(x, f(x))$, called *examples* of f. We need it to be computationally intractable to use this information to "learn" f, e.g. output a circuit that agrees with f on all but a small fraction of inputs. There are simple function families for which learning from (uniformly) random examples is believed to be intractable. Such families include noisy linear functions over finite fields (see the learning parity with noise (LPN) [BKW03] and learning with errors (LWE) [Reg05] assumptions), $O(\log n)$-juntas [Blu03], and more generally[2] polynomial-size decision trees, DNFs, CNFs, and Fourier-sparse functions. In each of these cases, we emphasize that the complexity of learning f depends on the joint distribution of the examples, and is only conjectured to be difficult for examples that are independent and uniformly random.

On the other hand, the learner has the power to *choose* the values of x for which it sees $f(x)$. This power, known in the learning theory literature as the ability to make *membership queries*, enables learning all of the aforementioned function families in polynomial time. However, this power is a double-edged sword in our

*NTT Research. Email: justin.holmgren@ntt-research.com.

†University of Illinois, Urbana-Champaign. Email: jawale2@illinois.edu.

[1] We are deliberately vague for now about the precise meaning of "learn"; we defer to section 2 a detailed specification of the learning model used in our results.

[2] It is easy to prove that any k-junta has a decision tree of depth k and size 2^k, and also has at most 2^k non-zero Fourier coefficients.

setting; if not wielded carefully, it provides exactly the same benefits to the adversary! In fact, a learning algorithm must incorporate at least a one-way function in order to have any advantage over the adversary: if the learner's queries are not a (distributional) one-way function [IL89] of the learner's randomness, then the adversary could emulate the learner with arbitrary inverse polynomial accuracy. This also shows that we can only hope for security against a computationally bounded adversary.

Following Canetti and Karchmer [CK21], we focus on preserving any advantage of membership queries over random examples. They defined a learning algorithm to be *covert* if its transcript[3] with the membership query oracle is *simulatable* from random examples. As with all simulation definitions of security, there are variants (e.g. with computational, statistical, and perfect security) corresponding to analogous degrees of indistinguishability of the simulator's output from reality.

Previous Covert Learning Algorithms There are two main previous results on covert learning, which we now quickly summarize.

1. Prior to [CK21], Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS19] gave a simple algorithm for learning $O(\log n)$-juntas with a relaxed notion of covertness that we retroactively call 1-out-of-2 covertness. In this relaxation, the learning algorithm's membership queries are distributed across *two* oracles, and we only require that the transcript with any *one* of the oracles is simulatable given random examples.

   The [IKOS19] algorithm is non-trivial in the sense that $O(\log n)$-juntas are not known to be learnable in polynomial time from random examples. However, this non-triviality is quantitatively rather weak: r-juntas are learnable in time $O(n^r)$.

   The notion of "1-out-of-2 covertness" naturally generalizes to "t-out-of-k covertness" for any $t \leq k$. The single-oracle setting considered by Canetti and Karchmer is the special case obtained by setting $t = k = 1$. We call this case *globally* covert learning, and we call the $t < k$ case *locally* covert learning.

   The motivating scenario with which we began this introduction is nearly as relevant to locally covert learning as it is to globally covert learning. Only a small modification is required: the company A now outsources its experiments to *multiple* companies $B_1, \ldots, B_k$, and aims to deter employees from t of these companies from colluding and selling their (combined) experiment outcomes to a competitor C.

2. Canetti and Karchmer [CK21] devised a *globally* covert algorithm for learning (polynomial-size) decision trees. In fact, their algorithm achieves stronger guarantees known as *agnostic learning* — even if the target function f is only α-close to a decision tree, their algorithm can still produce a circuit that is $(\alpha + \epsilon)$-close to f for arbitrarily small ϵ. Compared to [IKOS19], Canetti and Karchmer learn a larger family of functions with stronger (agnostic) correctness guarantees, and they achieve global rather than local covertness, but they rely on the sub-exponential LPN assumption and only achieve the computational variant of covertness.

   Their main lemma regards a task that we call Goldreich-Levin learning (after the classic paper [GL89], which gave the first algorithm for the problem): using membership queries, produce a list of all *parity* functions whose correlation with f is above a given threshold $\tau > 0$. Goldreich-Levin learning is complemented by the Kushilevitz-Mansour algorithm [KM93], which shows how to learn a Fourier-sparse function f from only random examples if one is additionally given such a list of parity functions. Thus there is a quite general reduction from covert learning to covert Goldreich-Levin learning.

   Canetti and Karchmer's algorithm actually achieves only a weak variant of Goldreich-Levin learning; it does not output *all* parity functions that are correlated with f, only those that depend on $O(\log n)$ variables. Nevertheless, they show that this variant is sufficient for learning decision trees. We note that decision trees are also learnable with only random examples if the learner is allowed to run in quasi-polynomial time (and use quasi-polynomially many examples).

[3] The transcript lists the queries made to the oracle alongside the corresponding oracle replies.

1.1 Our Contributions

For any constant k, we construct a polynomial-time algorithm for Goldreich-Levin learning that is (perfectly) $(k-1)$-out-of-k covert. Combined with the Kushilevitz-Mansour algorithm, this immediately implies a polynomial-time algorithm for $(k-1)$-out-of-k covertly and agnostically learning Fourier-sparse functions under the uniform distribution. In contrast to previous works, this learning task is probably not achievable in quasi-polynomial time given only random examples. Assuming the sub-exponential hardness of LPN (with constant noise rate), it is even impossible to agnostically learn *parities*, which are maximally Fourier-sparse, in *sub-exponential* time given only random examples.

Unlike [CK21], our algorithm achieves the full functionality of the Goldreich-Levin algorithm (outputting *all* parities that correlate with the target function). In fact, for $k = 2$ our algorithm is nearly identical to the original algorithm! We merely specify which queries go to which oracle, add some dummy queries, and observe that the resulting algorithm is perfectly 1-out-of-2 covert. Our generalization to $k > 2$ is just a simple tweak to this algorithm, the crux of which is our Proposition 4.1.

We believe that our observation, though straightforward, is only obvious with hindsight. In particular, the original Fourier-analytic Goldreich-Levin algorithm that we crucially rely on appears to have been largely forgotten within the cryptography community. It has been supplanted in every cryptography curriculum known to the authors (and also in [CK21, GRSY21]) by a more elementary algorithm that is attributed (see e.g. [Bel99]) to Charles Rackoff. However, Rackoff's algorithm does *not* suffice for us. It is not locally covert with any reasonable parameters (i.e. t-out-of-k covert for $t = \Omega(k)$) for the simple reason that its queries do not have enough entropy. We believe that the modern language of Fourier analysis makes it easier to appreciate the elegance of the original proof. In the course of explaining our observations we give a succinct exposition of that proof (heavily inspired by O'Donnell [O'D14]), which we hope will help in that regard.

Interactive Proofs for Verifying Machine Learning [GRSY21]. We also mention the work of [GRSY21], which focused on a different but related problem that they call interactive proofs for PAC verification. In this problem, the goal is to verify that a given hypothesis $\tilde{h}$ is as accurate as it is purported to be, making use of both random examples and interaction with an untrusted prover. At first glance this appears easy even without a prover — one can easily see how well $\tilde{h}$ agrees with the target concept f by testing $\tilde{h}$ on random examples for f. The key requirement that makes their problem technically challenging is that they focus on agnostic learning, where $\tilde{h}$ is supposed to perform as well as the *best* function h^* in a function family $\mathcal{F}$, and the verifier does not know h^* or how well h^* agrees with f.

Covert learning turns out to be applicable to this problem, as explored in both [CK21] and [GRSY21]. The rough idea is that the verifier runs a covert learning algorithm $\mathcal{L}$, and whenever $\mathcal{L}$ makes a query q, the verifier requests $f(q)$ from the prover. To prevent the prover from lying with impunity, the verifier also requests $f(x)$ for different values of x for which the verifier already knows $f(x)$ — such x are readily available in the form of random examples for f. Covertness is used to ensure that the prover cannot distinguish these "dummy" queries (on which incorrect answers would be caught) from the real queries (where incorrect answers would be impactful). Since the prover is now forced to answer queries mostly correctly, the verifier is assured that the resulting output of $\mathcal{L}$ is good.

The connection between covertness and interactive PAC verifiability remains intact for locally covert learning. Specifically, a 1-out-of-k covert learning algorithm gives rise to a standard *multi-prover* interactive proof (MIP), where each prover's messages are a function only of the messages sent to that prover. Starting instead with a t-out-of-k covert learning algorithm for $t > 1$ gives soundness against a form of bounded prover collusion that to our knowledge has not been previously studied in the context of MIPs.

2 Locally Covert Learning

In this section, we spell out the details of our learning model. We first formalize what an adversary (a coalition of oracles) is able to see when one of our learning algorithms is executed.

Definition 2.1 (The Adversarial View). *For any k-oracle algorithm $\mathcal{L}$, an input x, and a function f, we define*

$$\mathsf{View}^f\big(\mathcal{L}, x\big) := (\mathcal{T}_1, \ldots, \mathcal{T}_k),$$

where $\mathcal{T}_1, \ldots, \mathcal{T}_k$ are correlated random variables that are sampled as follows. Let r be uniformly sampled randomness for $\mathcal{L}$. Suppose that on input x and randomness r, and with each oracle implementing f, the algorithm $\mathcal{L}$'s queries to its i^{th} oracle are $q_1^{(i)}, \ldots, q_{m_i}^{(i)}$. Then $\mathcal{T}_i$ (the transcript with the i^{th} oracle) is defined as

$$\mathcal{T}_i := \Big(\big(q_1^{(i)}, f(q_1^{(i)})\big), \ldots, \big(q_{m_i}^{(i)}, f(q_{m_i}^{(i)})\big)\Big).$$

For any subset $S = \{i_1, \ldots, i_t\} \subseteq [k]$ with $i_1 < \cdots < i_t$, we also define $\mathsf{View}_S^f(\mathcal{L}, x) := \big(\mathcal{T}_{i_1}, \ldots, \mathcal{T}_{i_t}\big)$.

We are now ready to define locally covert algorithms, following the standard real/ideal simulation paradigm that dates back to [GM82].

Definition 2.2 (Local Covertness). *We say that a k-oracle learning algorithm $\mathcal{L}$ is* (perfectly) t-out-of-k covert modulo $\ell(\cdot)$ *if there exists a probabilistic polynomial-time algorithm* Sim *such that for every subset $S \subseteq [k]$ with $|S| \leq t$, every function f, and every input x, it holds that*

$$\mathsf{View}_S^f(\mathcal{L}, x) \equiv \mathsf{Sim}^{\mathsf{Ex}(f)}\big(S, \ell(x)\big), \tag{1}$$

where $\equiv$ denotes equality of distributions and $\mathsf{Ex}(f)$ denotes a probabilistic oracle that when queried, samples x uniformly from the domain of f and returns $\big(x, f(x)\big)$.

If "$\equiv$" in Equation (1) is replaced by a form of computational (resp. statistical) indistinguishability, we say that $\mathcal{L}$ is computationally (resp. statistically) t-out-of-k covert.

In this definition, the function $\ell(\cdot)$ serves to enumerate all leakage about $\mathcal{L}$'s input that we consider benign. For example, the input x to a learning algorithm often includes an accuracy parameter ϵ and a confidence parameter δ. Relaxing the desired guarantees on accuracy or error probability means that the algorithm can make fewer queries. Fewer queries is usually a *good* thing, but it is also clearly visible to the adversary, so we declare it benign by defining $\ell(x)$ to include ϵ and δ.

On the other hand, for some learning algorithms the input may explicitly include more sensitive information, such as auxiliary information z about the target function f. When this information is omitted from $\ell(x)$, covertness modulo $\ell(\cdot)$ guarantees that the algorithm's queries do not convey information about z.

3 Fourier Analysis Preliminaries

It is mathematically convenient in this paper to view Boolean functions as functions from $\mathbb{F}_2^n$ to $\{-1, 1\}$ (as opposed to functions from $\{0,1\}^n$ to $\{0,1\}$). In this setting, a parity function is also known as a *character*.

Definition 3.1. *A* character *of $\mathbb{F}_2^n$ is a homomorphism from $\mathbb{F}_2^n$ to the multiplicative group $\{-1, 1\} \subseteq \mathbb{R}$. Characters are parameterized by a vector $\gamma \in \mathbb{F}_2^n$, with the character corresponding to γ mapping $x \mapsto (-1)^{\gamma \cdot x}$.*

The characters form a group under multiplication, which corresponds to addition of the vectors γ in $\mathbb{F}_2^n$. We will identify characters with their index, i.e. we will simply write $\gamma(x)$ rather than $(-1)^{\gamma \cdot x}$. To avoid resulting confusion, we will denote the set of characters by $\widehat{\mathbb{F}_2^n}$ instead of $\mathbb{F}_2^n$.

Theorem 3.2 (Fourier Expansion Theorem [O'D14, Theorem 1.1]). *Every function $f : \mathbb{F}_2^n \to \mathbb{R}$ can be uniquely expressed as*

$$f(x) = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma) \cdot \gamma(x)$$

where each $\hat{f}(\gamma)$ is a real number called the Fourier coefficient of f on γ. *This expression is called the* Fourier expansion *of f.*

Proposition 3.3 (Fourier Coefficient Formula [O'D14, Proposition 1.8]). *For $f : \mathbb{F}_2^n \to \mathbb{R}$ and $\gamma \in \widehat{\mathbb{F}_2^n}$, the Fourier coefficient $\hat{f}(\gamma)$ is given by the formula*

$$\widehat{f}(\gamma) = \mathop{\mathbb{E}}_{x \leftarrow \mathbb{F}_2^n}[f(x) \cdot \gamma(x)].$$

One can view $\hat{f}$ as a function from $\mathbb{F}_2^n$ to $\mathbb{R}$ (because characters γ correspond to elements of $\mathbb{F}_2^n$ as described in Definition 3.1). Then combining Theorem 3.2 and Proposition 3.3, we obtain the standard fact that two iterations of the Fourier transform applied to a function returns the same function scaled by a factor of 2^{-n}.

Theorem 3.4 (Fourier Duality). *For any $f : \mathbb{F}_2^n \to \mathbb{R}$, $\hat{\hat{f}} = 2^{-n} \cdot f$.*

Theorem 3.5 (Parseval's theorem). *For any function $f : \mathbb{F}_2^n \to \mathbb{R}$, it holds that*

$$\mathop{\mathbb{E}}_{x \leftarrow \mathbb{F}_2^n}[f(x)^2] = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \widehat{f}(\gamma)^2.$$

Recall that for a subset $A \subseteq X$, the *indicator function* of A in X, denoted by 1_A, maps $x \in X$ to 1 if $x \in A$ and to 0 otherwise.

Proposition 3.6 (Fourier Transform of Affine Subspaces [O'D14, Proposition 3.12]). *If $A = V + x$ is an affine subspace of $\mathbb{F}_2^n$ with codimension k, then*

$$\widehat{1_A}(\gamma) = \begin{cases} 2^{-k} \cdot \gamma(x) & \text{if } \gamma \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Here $1_A : \mathbb{F}_2^n \to \{0,1\}$ denotes the indicator function for A in $\mathbb{F}_2^n$, and $V^\perp$ denotes the set of $\gamma \in \widehat{\mathbb{F}_2^n}$ for which $\gamma \cdot v = 0$ (i.e. $\gamma(v) = 1$) for every $v \in V$.

In the statement of Proposition 3.6, note that although the decomposition $A = V + x$ is not unique (A can also be written $A = V + x'$ for any $x' \in x + V$), the definition of $\widehat{1_A}$ is independent of the choice of decomposition. This is because for any $x' \in x + V$ and any $\gamma \in V^\perp$, we have $\gamma(x) = \gamma(x' + (x - x')) = \gamma(x') \cdot \gamma(x - x') = \gamma(x')$.

Corollary 3.7. *If $A = V + \gamma^*$ is an affine subspace of $\widehat{\mathbb{F}_2^n}$ with dimension d, then for any $x \in \mathbb{F}_2^n$,*

$$\sum_{\gamma \in A} \gamma(x) = \begin{cases} 2^d \cdot \gamma^*(x) & \text{if } x \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, the function $f : \mathbb{F}_2^n \to \mathbb{R}$ defined by $\hat{f} = 1_A : \widehat{\mathbb{F}_2^n} \to \{0,1\}$ is

$$f(x) = \begin{cases} 2^d \cdot \gamma^*(x) & \text{if } x \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Follows from applying Fourier duality (Theorem 3.4) to Proposition 3.6. □

Corollary 3.8 (Generalization of Plancherel). *For any $f_1, \ldots, f_k : \mathbb{F}_2^n \to \mathbb{R}$, it holds that*

$$\sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \prod_{j=1}^{k} \hat{f}_j(\gamma) = \mathop{\mathbb{E}}_{\substack{x_1, \ldots, x_k \in \mathbb{F}_2^n \\ x_1 + \cdots + x_k = 0}} \left[\prod_{j=1}^{k} f_j(x_j) \right]. \tag{2}$$

Proof. We have

$$\begin{aligned}
\sum_{\gamma\in\widehat{\mathbb{F}_2^n}}\prod_{j=1}^{k}\hat{f}_j(\gamma) &= \sum_{\gamma}\prod_{j=1}^{k}\mathop{\mathbb{E}}_{x\leftarrow\mathbb{F}_2^n}[f_j(x)\cdot\gamma(x)] && \text{(Proposition 3.3)}\\
&= \sum_{\gamma}\mathop{\mathbb{E}}_{x_1,\ldots,x_k\leftarrow\mathbb{F}_2^n}\Big[\prod_{j=1}^{k}f_j(x_j)\cdot\gamma(x_j)\Big] && \text{(independence of } x_1,\ldots,x_k)\\
&= \sum_{\gamma}\mathop{\mathbb{E}}_{x_1,\ldots,x_k\leftarrow\mathbb{F}_2^n}\Big[\Big(\prod_{j=1}^{k}f_j(x_j)\Big)\cdot\gamma\Big(\sum_{j=1}^{k}x_j\Big)\Big] && \text{(characters are homomorphisms)}\\
&= \mathop{\mathbb{E}}_{x_1,\ldots,x_k\leftarrow\mathbb{F}_2^n}\Big[f_1(x_1)\cdots f_k(x_k)\cdot\sum_{\gamma}\gamma\Big(\sum_{j=1}^{k}x_j\Big)\Big] && \text{(linearity of expectation)}
\end{aligned}$$

But this is equal to

$$\mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_k\in\mathbb{F}_2^n\\ x_1+\cdots+x_k=0}}\left[\prod_{j=1}^{k}f_j(x_j)\right]$$

because

$$\sum_{\gamma}\gamma\Big(\sum_{j=1}^{k}x_j\Big)=\begin{cases}2^n & \text{if } x_1+\cdots+x_k=0\\ 0 & \text{otherwise.}\end{cases}$$

□

4 Covertly Measuring Fourier Weight on Affine Spaces

The following algorithm is the main subroutine in our exposition of the Goldreich-Levin algorithm.

Proposition 4.1. *For all $k\in\mathbb{Z}^+$, there is a perfectly $(k-1)$-out-of-k covert algorithm (modulo the parameters n, ϵ and δ below) that takes as input:*

- *an affine subspace $A=V+\gamma^*$ of $\widehat{\mathbb{F}_2^n}$, where V is a vector subspace;*
- *an "accuracy" parameter $\epsilon>0$;*
- *a "confidence" parameter $\delta>0$; and*
- *oracle access to a function $f:\mathbb{F}_2^n\to[-1,1]$,*

runs in time $\mathsf{poly}\big(n,\frac{1}{\epsilon},\log(\frac{1}{\delta})\big)$ and, with all but δ probability, outputs a real number $\tilde{w}$ satisfying

$$\tilde{w}-\epsilon\le\sum_{\gamma\in A}\hat{f}(\gamma)^k\le\tilde{w}+\epsilon.$$

Moreover, the queries of this algorithm are non-adaptive and are computable in time $\mathsf{poly}\big(n,\frac{1}{\epsilon},\log(\frac{1}{\delta})\big)$ given V, ϵ, δ, and the algorithm's randomness (in particular there is no dependence on γ^*).

Proof. Let $A=V+\gamma^*$ be an affine subspace of $\widehat{\mathbb{F}_2^n}$ (here V is the vector space parallel to A, and $\gamma^*\in\widehat{\mathbb{F}_2^n}$ is the offset of A), and let d denote the dimension of A. We can write

$$\sum_{\gamma\in A}\hat{f}(\gamma)^k=\sum_{\gamma\in\widehat{\mathbb{F}_2^n}}\hat{f}(\gamma)^k\cdot 1_A(\gamma).$$

By Corollary 3.7, the indicator function $1_A : \widehat{\mathbb{F}_2^n} \to \{0,1\}$ is the Fourier transform of the function

$$g : \mathbb{F}_2^n \to \mathbb{R}$$
$$g(x) = \begin{cases} 2^d \cdot \gamma^*(x) & \text{if } x \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Rewriting 1_A as $\hat{g}$, we obtain

$$\sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma)^k \cdot \hat{g}(\gamma) = \mathop{\mathbb{E}}_{x_1,\ldots,x_k \leftarrow \mathbb{F}_2^n} \left[f(x_1) \cdots f(x_k) \cdot g(x_1 + \cdots + x_k) \right] \qquad \text{(Corollary 3.8)}$$
$$= \mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_k \\ x_1+\cdots+x_k \in V^\perp}} \left[f(x_1) \cdots f(x_k) \cdot \gamma^*(x_1 + \cdots + x_k) \right] \qquad (3)$$

Our algorithm estimates Equation (3) by sampling $m = O(\log(1/\delta)/\epsilon^2)$ i.i.d. tuples $\left((x_1^{(i)}, \ldots, x_k^{(i)})\right)_{i \in [m]}$ sampled uniformly from $\left(\mathbb{F}_2^n\right)^k$ conditioned on $x_1^{(i)} + \cdots + x_k^{(i)} \in V^\perp$. Then for each $j \in [k]$, the algorithm queries its j^{th} oracle to obtain $f(x_j^{(i)})$, and computes a "sample" $f(x_1^{(i)}) \cdots f(x_k^{(i)}) \cdot \gamma^*(x_1^{(i)} + \cdots + x_k^{(i)})$. Finally, it outputs the average of these samples. This output satisfies the desired accuracy guarantee by Hoeffding's inequality. The "moreover" of the theorem statement follows from the fact that the values $(x_j^{(i)})$ were sampled from a distribution that depends only on V.

$(k-1)$-out-of-k covertness follows from the standard fact that for *any* v, when $(x_1^{(i)}, \ldots, x_k^{(i)})$ are sampled uniformly from $\mathbb{F}_2^n$ conditioned on $x_1^{(i)} + \cdots + x_k^{(i)} = v$, the distribution of $x_S^{(i)}$ is uniform on $(\mathbb{F}_2^n)^{|S|}$ for any $S \subseteq [k]$ with $|S| < k$. □

5 The Goldreich-Levin Theorem

The seminal theorem of Goldreich and Levin [GL89] plays an important role in cryptography, learning theory, and this paper. Loosely speaking, the theorem says that it is possible to efficiently find all heavy Fourier coefficients of a function f using membership queries to f. We prove a $(k-1)$-out-of-k covert variant of this theorem for any constant k (the running time and number of queries grow exponentially in k).

Theorem 5.1 (Locally Covert Goldreich-Levin)**.** *For every integer $k \geq 2$, there is an explicit algorithm that when given:*

- *an integer $n \in \mathbb{Z}^+$,*
- *a "threshold" parameter $0 < \tau \leq 1$,*
- *a "confidence" parameter $\delta > 0$, and*
- *access to k oracles all implementing the same function $f : \mathbb{F}_2^n \to [-1,1]$,*

the algorithm runs in time $\mathsf{poly}(n, \log(\frac{1}{\delta}), \frac{1}{\tau^k})$ *and outputs a set $S \subseteq \widehat{\mathbb{F}_2^n}$ (of size $O(1/\tau^k)$) such that with all but δ probability,*

$$S \text{ contains all } \gamma \in \widehat{\mathbb{F}_2^n} \text{ for which } |\hat{f}(\gamma)| \geq \tau. \qquad (4)$$

Moreover, this algorithm is perfectly $(k-1)$-out-of-k covert modulo (n, τ, δ).

Proof. We assume without loss of generality that k is even (so we are looking for γ satisfying $\hat{f}(\gamma)^k \geq \tau^k$). This is without loss of generality because for odd k, we can emulate a k-out-of-$(k+1)$ covert algorithm $\mathcal{L}$. Whenever $\mathcal{L}$ makes a query q to its i^{th} oracle, we pass the query to our $(i \bmod k)^{th}$ oracle. With this query

mapping, it is easy to see that the view of any $k-1$ of our oracles is simulatable from the view of k of $\mathcal{L}$'s oracles, which is in turn simulatable from random examples.

For a given f and τ, say that $\gamma \in \widehat{\mathbb{F}_2^n}$ is *heavy* if $|\hat{f}(\gamma)| \geq \tau$, and let H denote the set of heavy γ. Our algorithm maintains a set S of $O(1/\tau^k)$ disjoint affine subspaces that collectively cover H (i.e. $H \subseteq \cup_{A \in S} A$). The algorithm starts with the trivial covering $S = \{\widehat{\mathbb{F}_2^n}\}$. It then "refines" S until S contains only affine sets of dimension 0, i.e. singleton sets, at which point S (or more precisely $\cup_{A \in S} A$) is the desired output.

To refine S, the algorithm repeats the following two steps n times:

1. Replace each $A \in S$ by disjoint A_0 and A_1 such that $\dim A_0 = \dim A_1 = \dim A - 1$ and $A = A_0 \cup A_1$.
2. Use Proposition 4.1 to "filter" S, keeping all A for which $\sum_{\gamma \in A} \hat{f}(\gamma)^k \geq \tau^k$ (in particular this includes all A that contain a heavy γ) and removing all A for which $\sum_{\gamma \in A} \hat{f}(\gamma)^k \leq \tau^k/2$. This involves running the algorithm of Proposition 4.1 $|S|$ times, where recall $|S| \leq O(1/\tau^k)$. For covertness, we we also perform up to $O(1/\tau^k)$ "dummy" executions of the algorithm so that we do not leak information through $|S|$.

We remark that with a careful choice of decomposition in step 1, this algorithm can be made nonadaptive. Specifically, fix a basis $e_1, \ldots, e_n$ for $\mathbb{F}_2^n$, and in the i^{th} iteration choose $A_b := \{\gamma \in A : \gamma(e_i) = (-1)^b\}$. Then in the i^{th} iteration at step 2, each affine space $A \in S$ will be parallel to the fixed vector space $V_i = \{\gamma \in \widehat{\mathbb{F}_2^n} : \gamma(e_1) = \cdots = \gamma(e_i) = 1\}$. The "moreover" of Proposition 4.1 then implies that the queries can all be made nonadaptively.

To complete the description and analysis of the algorithm, it remains to show that this filtration process guarantees $|S| \leq O(1/\tau^k)$. Specifically we claim $|S| \leq 2/\tau^k$, for otherwise we would have

$$\begin{aligned}
1 &< |S| \cdot \frac{\tau^k}{2} \\
&\leq \sum_{A \in S} \sum_{\gamma \in A} \hat{f}(\gamma)^k \\
&\leq \sum_{A \in S} \sum_{\gamma \in A} \hat{f}(\gamma)^2 && \text{(each } \hat{f}(\gamma) \text{ lies in } [-1,1]) \\
&\leq \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma)^2 && \text{(the spaces } A \in S \text{ are disjoint)} \\
&\leq 1 && \text{(Parseval).}
\end{aligned}$$

□

A Agnostic Learning from Heavy Fourier Coefficients

In agnostic learning, the goal is to learn a hypothesis $h \in \mathcal{H}$ that is nearly as good as any other hypothesis $\mathcal{H}$. More precisely, we aim to (approximately) minimize a *loss function* $L(h, f)$.

We use the squared ℓ_2 loss function, i.e. $L(h,f) = \|h - f\|^2 := \mathbb{E}_{x \leftarrow \mathbb{F}_2^n}\left[(h(x) - f(x))^2\right]$. This loss function possesses a number of appealing mathematical properties. When $h, f : \mathbb{F}_2^n \to \{0,1\}$ are both Boolean functions, $L(h,f)$ is just the fraction of inputs on which h and f differ. On the other hand if $\tilde{h}$ is a *real-valued* function with $L(\tilde{h}, f) = \epsilon$, then one can efficiently obtain a *Boolean* function h with $L(h,f) \leq 4\epsilon$ by setting $h(x) = 0$ if $\tilde{h}(x) \leq 1/2$, and $h(x) = 1$ otherwise.

This loss function has two appealing properties that connect it to Boolean functions. When $h, f : \mathbb{F}_2^n \to \{-1,1\}$ are both Boolean functions, $L(h,f)$ is exactly four times the fraction of inputs on which h and f differ. On the other hand if $\tilde{h}$ is a *real-valued* function with $L(\tilde{h}, f) = \alpha$, then one can efficiently obtain a *Boolean* function h with $L(h,f) \leq 4\alpha$ by setting $h(x) = \mathrm{sign}(\tilde{h}(x))$, i.e.

$$h(x) = \begin{cases} 1 & \text{if } \tilde{h}(x) \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

For completeness we include a proof of the following proposition, which was *already known* (see e.g. the survey of Mansour [Man94]).

Proposition A.1. *For every integer $k \geq 2$, there is an explicit algorithm that when given:*

- *an integer $n \in \mathbb{Z}^+$,*
- *a "sparsity" parameter $t \in \mathbb{Z}^+$,*
- *an "accuracy" parameter $\epsilon > 0$,*
- *a "confidence" parameter $\delta > 0$, and*
- *access to k oracles all implementing the same function $f : \{-1,1\}^n \to [-1,1]$,*

runs in time $\mathsf{poly}\left(n, \log(\frac{1}{\delta}), \left(\frac{t}{\epsilon}\right)^k\right)$ and outputs a t-sparse polynomial $\tilde{p} : \mathbb{R}^n \to \mathbb{R}$ (as a list of monomials and coefficients) such that with all but δ probability, $L(\tilde{p}, f) \leq L(p, f) + \epsilon$ for every *t-sparse polynomial p.*

Moreover, this algorithm is perfectly $(k-1)$-out-of-k covert modulo (n, t, ϵ, δ).

Proof. The algorithm executes the following steps:

1. Use the (locally covert) Goldreich-Levin algorithm of Theorem 5.1 to find a set $\tilde{S} \subseteq \widehat{\mathbb{F}_2^n}$ such that, for some parameter τ to be specified later, $\tilde{S}$ contains all γ for which $|\hat{f}(\gamma)| \geq \tau$ (unless a certain bad event B_1 occurs, which happens with probability at most $\delta/2$). Without loss of generality assume $|\tilde{S}| \geq t$ (arbitrary elements can be added to $\tilde{S}$ if necessary to ensure this).

2. Use random examples to obtain estimates $\tilde{f}(\gamma)$ of $\hat{f}(\gamma)$ such that

$$\left|\tilde{f}(\gamma) - \hat{f}(\gamma)\right| \leq \sqrt{\tau} \text{ for each } \gamma \in \tilde{S} \tag{5}$$

   (unless a certain bad event B_2 occurs, which happens with probability at most $\delta/2$).

3. Sort the elements of $\tilde{S}$ as $\tilde{\gamma}_1, \ldots, \tilde{\gamma}_{|\tilde{S}|}$ such that $|\tilde{f}(\tilde{\gamma}_1)| \geq |\tilde{f}(\tilde{\gamma}_2)| \geq \cdots \geq \left|\tilde{f}(\tilde{\gamma}_{|\tilde{S}|})\right|$, and define $\tilde{S}_t = \{\tilde{\gamma}_1, \ldots, \tilde{\gamma}_t\}$.

4. Output $\tilde{p} = \sum_{\gamma \in \tilde{S}_t} \tilde{f}(\gamma) \cdot \gamma$.

Toward analyzing the correctness of this algorithm, enumerate the elements of $\widehat{\mathbb{F}_2^n}$ as $\gamma_1, \gamma_2, \ldots, \gamma_{2^n}$ such that $|\hat{f}(\gamma_1)| \geq \cdots \geq \left|\hat{f}(\gamma_{2^n})\right|$, let S_t denote the set $\{\gamma_1, \ldots, \gamma_t\}$, and define the function $p = \sum_{\gamma \in S_t} \hat{f}(\gamma) \cdot \gamma$. Note that this p is the t-sparse function that is closest to f, so we wish to compare $L(\tilde{p}, f)$ to $L(p, f)$.

$L(p, f)$ has a convenient formula in the Fourier domain:

$$L(p, f) = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \left(\hat{p}(\gamma) - \hat{f}(\gamma)\right)^2 = \sum_{\gamma \notin S_t} \hat{f}(\gamma)^2. \tag{6}$$

Similarly

$$L(\tilde{p}, f) = \sum_{\gamma \notin \tilde{S}_t} \hat{f}(\gamma)^2 + \sum_{\gamma \in \tilde{S}_t} \left(\hat{f}(\gamma) - \tilde{f}(\gamma)\right)^2, \tag{7}$$

so we have

$$L(\tilde{p}, f) - L(p, f) = \sum_{\gamma \notin \tilde{S}_t} \hat{f}(\gamma)^2 - \sum_{\gamma \notin S_t} \hat{f}(\gamma)^2 + \underbrace{\sum_{\gamma \in \tilde{S}_t} \left(\hat{f}(\gamma) - \tilde{f}(\gamma)\right)^2}_{\leq t\tau \text{ by } (5)}. \tag{8}$$

It remains to bound

$$\sum_{\gamma \in \tilde{S}_t} \hat{f}(\gamma)^2 - \sum_{\gamma \in S_t} \hat{f}(\gamma)^2 = \sum_{i=1}^{t} \left(\hat{f}(\gamma_i)^2 - \hat{f}(\tilde{\gamma}_i)^2 \right). \tag{9}$$

Recall that $\hat{f}(\gamma_i)$ can be obtained by sorting the 2^n Fourier coefficients of f and picking the i^{th} largest. In comparison, $\hat{f}(\tilde{\gamma}_i)$ is obtained by first *perturbing* each Fourier coefficient by at most τ, then sorting and taking the i^{th} largest, then *unperturbing* by at most τ. Sorting of real numbers is 1-Lipschitz with respect to the ℓ_∞ metric (this follows from the analogous fact for $\min$ and $\max$), which implies that $|\hat{f}(\gamma_i) - \hat{f}(\tilde{\gamma}_i)| \leq 2\tau$ for every i. Since each $|\hat{f}(\gamma)|$ is at most 1, (9) is at most $4t\tau$, which finally implies that (8) is at most $5t\tau$.

Setting $\tau = \epsilon/5t$ then achieves the desired accuracy. □

References

[Bel99] Mihir Bellare. The goldreich-levin theorem, October 1999. Lecture notes, available at https://cseweb.ucsd.edu/~mihir/papers/gl.pdf.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[Blu03] Avrim Blum. Learning a function of r relevant variables. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pages 731–733. Springer, 2003.

[CK21] Ran Canetti and Ari Karchmer. Covert learning: How to learn with an untrusted intermediary. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 1–31. Springer, 2021.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 25–32. ACM, 1989.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982.

[GRSY21] Shafi Goldwasser, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 41:1–41:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[IKOS19] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptographic sensing. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 583–604. Springer, 2019.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *FOCS*, pages 230–235. IEEE Computer Society, 1989.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993.

[Man94] Yishay Mansour. Learning boolean functions via the fourier transform. *Theoretical advances in neural computation and learning*, pages 391–424, 1994.

[O'D14] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014. Available online at https://arxiv.org/abs/2105.10386.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.