Locally Covert Learning

Justin Holmgren
NTT Research

Ruta Jawale
UIUC
Learning Boolean Functions

1. Polynomial-time learner gets access to a function $f : \{0,1\}^n \rightarrow \{0,1\}$

2. Learner’s goal is to output a function $h$ that agrees with $f$ on most inputs.
The Learning Model, Part I

By “$\mathcal{H}$ is learnable”, we mean there exists a learning algorithm that is:

**Efficient:**
Given accuracy parameter $\alpha$ and confidence parameter $\delta$, learner runs in time $\text{poly}(n, 1/\alpha, \log(1/\delta))$.

**(Weakly) Agnostic:**
If target function $f$ is $\epsilon$-close to some $h^* \in \mathcal{H}$, the learner outputs a hypothesis $h$ that is $(O(\epsilon) + \alpha)$-close to $f$, with all but $\delta$ probability.

**Distribution-Specific:** “closeness” is measured wrt the uniform distribution

**Improper:** Learner can output any circuit, not necessarily in $\mathcal{H}$. 
The Learning Model, Part II

Types of Function Access

Passive Learning:
Learner gets pairs \((x, f(x))\)
for uniformly random \(x\)

Active Learning:
Learner gets oracle access to \(f\).
Passive vs. Active Learning

- quasi-poly time learnable passively
  - [Linial-Mansour-Nisan]

Learnable actively
- poly-size DNF (or CNF)
- poly-size DT
- parity
- polynomially Fourier concentrated

Learnable passively
- log(n)-junta
- not very much stuff...
where's the adversary?
Adversarial Learning

By now a burgeoning field. Includes, but not limited to:

1. **Covert Learning**: [Canetti-Karchmer ’21, IKOS ’19]
   curious eavesdropper tries to piggyback on the queries of an active learner.

2. **Verifiable Learning**: [Goldwasser-Rothblum-Shafer-Yehudayoff ’20]
   untrusted prover claims that a hypothesis $h$ approximates $f$ near-optimally (compared to some class of functions).
Covert Learning
[Canetti-Karchmer ’21, IKOS ’19]

usually i.i.d. uniform or pseudo-uniform queries

active learner

observer

≈

passive simulator

f
(1-of-2) Locally Covert Learning

[IKOS19]
Why Study Covert Learning?
Scenario 1: Delegating Scientific Discovery
[Canetti-Karchmer ’21]

**Plan:** Learn a function $f$ for which:
random examples are cheap / useless
queries are relatively useful but expensive,

For example, an organism’s genome $\rightarrow$ phenome map

**Problem:** Want to delegate to specialists, but …
what if they sell resulting data to your competitors?

**Solution:** use covert learning $\implies$ their data has no resale value
Scenario 2: Verifiable Learning

**Same Plan:** Delegate the query-learning of a function $f$; assume cheap but useless random examples for $f$.

**New Problem** [Goldwasser-Rothblum-Shafer-Yehudayoff '21]: How to ensure we receive a near-optimal circuit?

**One Approach:** Tell learner what queries to make (following a covert learning algorithm). Hide “test queries” (using random examples)

⇒ If test queries are correct, most others must be as well.

⇒ If learning algorithm is also “robust” then a few incorrect query answers can’t ruin the output.
Scenario 3: Model Extraction

[Canetti-Karchmer ’21]

Plan: Sell AI as a service (e.g. chat GPT)

• Generally trained on random data (more scalable)

Problem: Can competitor use queries to clone the model?

Defense?? Block users who make weird query patterns

😢 Can’t really work against a covert learner
Passive $\leq$ Covert $\leq$ Active

- Passive
- Covert (assuming LPN) [Canetti-Karchmer]
- Locally Covert [this work]
- Active

- $\log(n)$-junta [IKOS19]
- parity DT
- DNF (or CNF)
- polynomially Fourier concentrated

$\leq$ poly-size $\leq$
The Goldreich–Levin Theorem

Learning Version:
Given oracle access to $f$, one can efficiently find all parity functions $\gamma$ that are even weakly correlated with $f$.

Crypto Version:
Let $g$ be a OWF. Then $\langle x, r \rangle \pmod{2}$ is hard-core for $(g(x), r)$.

Proof assuming Learning Version:
1. $\langle x, \cdot \rangle \pmod{2}$ is a parity function.
2. If not hard-core, then an adversary $A(g(x), \cdot)$ weakly predicts $\langle x, r \rangle$.
3. GL $A(g(x), \cdot)$ outputs a list containing $x \mapsto$ contradicts that $g$ is a OWF.
Which “Goldreich-Levin Algorithm”?

Rackoff’s Algorithm

- Uses derandomization (querying all subset sums of $\approx \log(n)$ random vectors in $\mathbb{F}_2^n$)

⇒ Queries are not statistically uniform

The Original [Goldreich-Levin]

- Uses Fourier analysis
- Well-known in learning theory

This Work:

Original algorithm is basically already 1-out-of-2 covert.

Small modification gives $(k - 1)$-out-of-$k$ covertness.
(Locally) Covert Goldreich-Levin Algorithms

**Previous Lemma:** [Canetti-Karchmer] (following [Rackoff])

Assuming LPN is subexponentially hard, there is a **computationally** covert algorithm for **low-degree** Goldreich-Levin learning

- All log(n)-variable γ s.t. |\(\hat{f}(\gamma)\)| ≥ \(\tau\)
  (except with \(\delta\) probability)

**Our Main Theorem** (following [Goldreich-Levin]):

For any constant \(k\), there is a **perfectly** \((k - 1)\)-out-of-\(k\) covert algorithm for Goldreich-Levin learning

- All γ s.t. |\(\hat{f}(\gamma)\)| ≥ \(\tau\)
  (except with \(\delta\) probability)
Fourier Analysis Essentials

For \( \gamma \in \mathbb{F}_2^n \), let \( \hat{f}(\gamma) \in [-1,1] \) denote the correlation of \( f \) with the parity function \( \langle \gamma, \cdot \rangle \). Call \( \hat{f}(\gamma)^2 \) the weight of \( \gamma \).

Fact: \( \sum_{\gamma \in \mathbb{F}_2^n} \hat{f}(\gamma)^2 = 1 \). Not many heavy parities.

Lemma: With queries to \( f \), one can efficiently estimate weight \( (p) := \sum_{s \in \mathbb{F}_2^{n-k}} \hat{f}(p \circ s)^2 \) for any “prefix” \( p \in \mathbb{F}_2^k \), will prove this later

and 1-of-2 covertly

Not many heavy prefixes.
Weighing Parity Prefixes $\implies$ Goldreich-Levin

**Basic Idea:** Maintain a list of candidate prefixes of heavy parities $\gamma$ (those with $\hat{f}(\gamma)^2 \geq \tau$), starting with 1-bit prefixes $\{0,1\}$.

1. Weigh each prefix in the list and throw away light prefixes (those with weight $< \tau$)
   \[ \Rightarrow \text{At most } 1/\tau \text{ prefixes.} \]

2. Replace remaining prefixes $p$ by $p \circ 0$ and $p \circ 1$
   \[ \Rightarrow \text{At most } 2/\tau \text{ prefixes} \]

3. Repeat until prefixes are $n$-bit strings.
Lemma:

With queries to $f$, one can efficiently estimate

$$\text{weight}(p) := \sum_{s \in \mathbb{F}_2^{n-k}} \hat{f}(p \circ s)^2 \text{ for any "prefix" } p \in \mathbb{F}_2^k.$$ 

More Generally:

With queries to $f$, one can efficiently estimate

$$\text{weight}(A) := \sum_{\gamma \in A} \hat{f}(\gamma)^2 \text{ for any affine subspace } A \subseteq \mathbb{F}_2^n.$$
How To Weigh Affine Spaces

Lemma:
For any affine subspace $A \subseteq \mathbb{F}_2^n$, $A = \gamma^* + V$, one can efficiently estimate weight$(A) := \sum_{\gamma \in A} \hat{f}(\gamma)^2$

Formula:

$$\sum_{\gamma \in A} \hat{f}(\gamma)^2 = \mathbb{E}_{x_1 + x_2 \in V^\perp} \left[ f(x_1) \cdot f(x_2) \cdot \gamma^*(x_1 + x_2) \right]$$

Expectation can be directly empirically estimated
(\(k - 1\))-of-\(k\) Covertness

Previous formula naturally generalizes:
If \(A = \gamma^* + V\) is an affine subspace of \(\mathbb{F}_2^n\), then

\[
\sum_{\gamma \in A} \hat{f}(\gamma)^k = \mathbb{E}_{x_1 + \cdots + x_k \in V^\perp} \left[ f(x_1) \cdots f(x_k) \cdot \gamma^*(x_1 + \cdots + x_k) \right]
\]

• To get \((k - 1)\)-of-\(k\) covert GL, apply same strategy, using \(k\)-weight instead of 2-weight
Goldreich-Levin with $k$-weights

($k$ even WLOG)

**Strategy:** Maintain a list of candidate $l$-bit prefixes of heavy parities $\gamma$ (those with $\hat{f}(\gamma)^2 \geq \tau$), starting with $\{0,1\}$.

1. Weigh each prefix in the list and throw away light prefixes (those with $k$-weight $< \tau^{k/2}$)
   
   $\Rightarrow$ At most $1/\tau^{k/2}$ prefixes because 2-weight $> k$-weight

2. Replace remaining prefixes $p$ by $p \circ 0$ and $p \circ 1$
   
   $\Rightarrow$ At most $2/\tau^{k/2}$ prefixes

3. Repeat until prefixes are $n$-bit strings.

$\gamma \overset{\hat{f}}{\rightarrow} f(\gamma)^2 \geq \tau$

$k < \tau^{k/2}$

$1/\tau^{k/2}$ prefixes

$2/\tau^{k/2}$ prefixes

$\tau$ can be $1 / \text{poly}(n)$, so running time $n^{O(k)}$
Q3: Covert algorithms in other learning models?

Q4: Is $t$-out-of-$k$ covert learning easier for $t < k - 1$ than $t = k - 1$?

Q2: Is there a gap for natural classes?

Covert (assuming LPN) [Canetti-Karchmer]
Thanks & Happy Birthday!

ia.cr/2023/392