Covert learning and its applications

Ari Karchmer
Boston University
arika@bu.edu

Based on https://ia.cr/2021/764 (Canetti, Karchmer TCC ’21)
https://ia.cr/2022/1039 (Karchmer SATML ’23)
Learning

With uniformly random examples

Nature, \( f: \{0,1\}^n \rightarrow \{0,1\} \)

- Nature chooses \( f \) from some class \( C \) (e.g. bounded depth decision trees)
- Nature samples uniformly random points \( x_1 \ldots x_m \)

\[ \text{Pr}_x \left[ f(x) \neq h(x) \right] \leq \epsilon \]

With probability 0.99 over choices of and set of examples

Poly-time

Uniform over \( \{0,1\}^n \)
Learning

With Queries

Nature, \( f: \{0,1\}^n \rightarrow \{0,1\} \)

- Nature chooses \( f \) from some class \( C \) (e.g. a bounded depth decision tree)

Uniform over \( \{0,1\}^n \)

\[ \Pr_x \left[ f(x) \neq h(x) \right] \leq \epsilon \]

With probability 0.99 over choices of
Covert learning

• You have query access to a dataset owned by another party
• Try to conduct some learning algorithm (or analysis, testing, etc.)
• Can you prevent anyone who views your queries -- and the answers -- from understanding what you have learned?
Covert learning

Main caveat: the party who owns data won’t help you

• Maybe they are the adversary, or maybe they don’t know how
• All you can do is choose your queries carefully (and have secret state)
Model extraction

In a model extraction attack, a client maliciously probes an interface to a machine learning model in order to extract the machine learning model itself.

- Model owner is “adversary’s adversary,” and won’t help
- You’re on your own to steal the model
- Model owner will try to figure out if/what you are learning

Model owner will try to figure out if/what you are learning:
- Your queries are suspicious
- I won’t respond and will ban you

Approximate replica of decision tree model: you/adv

Model owner
Model extraction

In a model extraction attack, a client maliciously probes an interface to a machine learning model in order to extract the machine learning model itself.

- Model owner is “adversary’s adversary,” and won’t help
- You’re on your own to steal the model
- Model owner will try to figure out if/what you are learning

Stealing my model would help you do:
- inversion attacks
- adversarial example attacks

Decision tree model

You/adv

Approximate replica

Model owner
Many defenses observe

• Determine if you are adversarial or benign
  • e.g. Extraction Monitor [KMAM18], PRADA [JSMA19], VarDetect [PGKS21]

Estimate information gained by client by training proxy model based on queries. Warn model owner when information gain passes a threshold.

Conduct normality test on Hamming distances between queries, and rejects deviating clients. Assumes honest clients have this “benign” property.

Uses Variational Autoencoder to map “problem domain” queries and “outlier” queries to distinct regions in latent space. Classifies clients as honest or adversarial.

“Security concerns give rise to a strong interest in using cryptographic techniques for protecting data in the cloud. The challenge is to balance security, performance, and functionality.”

-ESSA23 webpage

Covert learning will, in theory, defeat any poly-time defense like this [Kar23]
Quantitative Structure-Activity Research (QSAR)

- Can think of an experiment as a request to observe whether a reaction occurs between k of n molecules

- Want to build a model/sketch of how Nature determines these reactions

- E.g. drug discovery: outsource experiments to a lab
Quantitative Structure-Activity Research (QSAR)

- Biological data can’t run a protocol with you
- Outsourced experiments, the Lab always sees results first
- The lab could sell the results of the experiments or intellectual property of an inferred hypothesis to competitors

A genius hypothesis!
The perfect experiment design!
Covert learning goals

**Learning:** Alice learns a good model from her experiments

**Concept-hiding:** No information about the underlying molecular relationship is leaked

**Hypothesis-hiding:** No information about Alice's hypothesis or domain knowledge used to influence the hypothesis is leaked
Defining covert learning

• Experiments are basically queries to “concept” \( f: \{0,1\}^n \rightarrow \{0,1\} \)
• Poly-time learning is defined learning-theoretically, for class of concepts \( C \)

\[
\Pr_A \left[ \Pr_x \left[ f(x) \neq h(x) \right] \leq \varepsilon : h \leftarrow A^f \right] \geq 1 - \delta
\]

• Simulation-based security. Define \( \text{View}(A^f) \) as the random variable given by a transcript of the labelled queries performed by \( A \)

• Secure if there exists \( \text{SIM} \) (random examples) such that for every \( A, f \) in \( C \), \( \text{SIM} \) is indistinguishable from \( \text{View}(A^f) \)
Defining covert learning

Q: How do you hide the constant function? Noiseless linear function?

A: You don’t, we seek to preserve “zero-knowledge” gained over random examples to the concept. Any function that can be learned with random examples is out.

- Backed by learning theory: learning DT/DNF is conjectured hard (from random examples)
“Natural covert learning”

- A query algorithm that uses pseudo-uniform queries

Experiments “look” uniformly random
Select covert learning positive results

<table>
<thead>
<tr>
<th>work</th>
<th>result</th>
<th>setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CK21]</td>
<td>Linear functions with little noise</td>
<td>“Single server,” computational security</td>
</tr>
<tr>
<td>Implicit in [IKOS19]</td>
<td>$O(\log n)$-juntas</td>
<td>“2-server,” perfect security</td>
</tr>
<tr>
<td>[JH23]</td>
<td>Fourier-sparse functions (agnostic)</td>
<td>“k-server,” perfect security</td>
</tr>
</tbody>
</table>

All these in “natural” setting of pseudo-uniform queries. Limited results outside this, great **Open Problems**!
## Select covert learning positive results

<table>
<thead>
<tr>
<th>work</th>
<th>result</th>
<th>setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CK21]</td>
<td>Linear functions with little noise</td>
<td>“Single server,” computational security</td>
</tr>
<tr>
<td>Implicit in [IKOS19]</td>
<td>O(log n)-juntas</td>
<td>“2-server,” perfect security</td>
</tr>
</tbody>
</table>

Very simple, will describe now

All these in “natural” setting of pseudo-uniform queries. Limited results outside this, great **Open Problems**!
Covert learning for noisy parities

**Low-noise LPN**

- **Low-noise LPN distribution** [Ale03]:
  - Sample $s \in \{0,1\}^n$, $e \in \{0,1\}$ from Bernoulli r.v. with mean $1/\sqrt{n}$
  - Sample $a \in \{0,1\}^n$ uniformly at random
  - Return $a, \langle a, s \rangle \oplus e$ ($s$ is persistent over each example)

- **Search LPN**: find $s$.

- **Decision LPN**: distinguish $a, \langle a, s \rangle \oplus e$ from uniformly random. Both are thought to be subexponentially-hard

- Is there a covert learning algorithm for learning $s$?

- Can learn it trivially with queries (exercise left to listeners)
Covert learning for noisy parities

“Masked queries”
Covert learning for noisy parities
Covert learning for noisy parities
Covert learning for noisy parities

For \( \mu \in [0, 1/2] \), and random variables \( E_1 \cdots E_m \) that are i.i.d. from Bernoulli r.v. with mean \( \mu \), we have that

\[
Pr[\bigoplus_{i=1}^m E_i = 0] = \frac{1}{2} + \frac{1}{2} (1 - 2\mu)^m
\]

Decoding procedure:

Noisy terms are biased to 0 w.p. \( \frac{1}{2} + \Omega(1) \)

Repetition and majority voting to decode bit-by-bit
“Multi-server” covert learning

*adversary sees just 1 of the 2 views*
Still obviously very relevant to model extraction and secure scientific discovery
k-Juntas

• An input variable $x_i$ of a function $f: \{0,1\}^n \rightarrow \{0,1\}$ is irrelevant if for any input $x$, $f(x) = f(y)$ where $y = x$ except at $x_i$

• A k-junta has at least $n-k$ irrelevant variables

• Learning $O(\log n)$-juntas with uniform examples is conjectured to require super-polynomial time (best algorithms only slightly better than brute force)

• Easy with queries (exercise)
2-oracle covert learning for juntas

- Implicit in [IKOS19] that studies distributed “cryptographic sensing”
- Motivated by bio data
2-server covert learning for juntas

• Use neighbor queries to find all variables that are relevant to the function
• Once we know these variables, use more random examples to build truth table of all $2^k$ settings of the variables

Works in polynomial time for $k = O(\log n)$
Open questions

• Make known covert learning algorithms practical! Our model may be much stronger than necessary for practical applications

• Study “covert property testing” learning is in general stronger than testing so we could possibly expect more here

• Is there a general compiler for any learning algorithm that makes it covert? My guess is no, give counter example?
• Grazie