Learning under Cryptographic Hardness

By: Anish Jayant

Cryptographic Assumption



For modern cryptography to be possible...

Definition (One-way function): A poly-time computable $f: \mathcal{X} \to \mathcal{Y}$ such that for all $\mathcal{A} \in PPT$,

$$\Pr[f(\mathcal{A}(f(x)) = f(x)] < \frac{1}{poly(|x|)}$$

Throughout the talk, we assume one-way functions exist $\Rightarrow P \neq NP$, $RP \neq NP$

Cryptographic Assumption

For modern cryptography to be possible...

Definition (One-way function): A poly-time computable $f: \mathcal{X} \to \mathcal{Y}$ such that for all $\mathcal{A} \in PPT$,

$$\Pr\left[f(\mathcal{A}(f(x))) = f(x)\right] < \frac{1}{poly(|x|)}$$

implies robustness to interactive attacker! (think about public-key...)



Cryptographic Assumption

For modern cryptography to be possible...

Definition (One-way function): A poly-time computable $f: \mathcal{X} \to \mathcal{Y}$ such that for all $\mathcal{A} \in PPT$,

$$\Pr\left[f(\mathcal{A}(f(x)) = f(x)\right] < \frac{1}{poly(|x|)}$$

Conjectured Hard Functions

Claim (Discrete Log. is One-Way): Given $f(x) = x^a \pmod{N}$, $\Pr[\mathcal{A}(f(x), x, N) = a]$ is small.

Claim (Factoring is One-Way): Given N = pq, $Pr[\mathcal{A}(N) \in \{p, q\}]$ is small.

Key Takeaways:

- All classes are *finite* thus mathematically possible to break (by brute force)
- Erratic: polytime algorithms can't distinguish a *hard-core bit* from pure randomness!

Attacking as Learning

• Property: Even with *best case* dataset, polytime attacker has no edge on f^{-1} : even worse on average!

Question: What does it mean to learn a set of behaviors?

Is there a set of behaviors that is hard to learn efficiently, but easy to learn otherwise One-Way Functions? Is there a relaxation of security that is hard in poly-time, but defeated with more compute

Goal Questions

By the end of the talk, we'd like some insight into

- What does it mean to learn a behavior, be learnable?
 - How does 'learning theory' differ from 'statistics'?
- What does it mean to (run-time) efficiently learn?
- Can one-way functions be efficiently learnable?
- (**) How do one-way results influence other structures, like Boolean circuits?

Let's look at a simple behavior that can be learned...

(English) Setup:

- User adds movies to watchlist if they are *above* some rating
- After observing *m* many decisions, learn a recommendation



Question: Given the decisions, how should we recommend?

A simple behavior...

(English) Setup:

- User adds movies to watchlist if they are above some rating
- After observing *m* many decisions, learn a recommendation



(Formal) Setup:

- Ratings $\mathcal{X} = [0, 10]$, Outputs $\mathcal{Y} = \{0, 1\}$
- For some c^* , $f(x) = 1(x \ge c^*)$

 $\mathcal{F} = \{ 1(x \ge c) | c \in [0, 10] \}$

Question: Given the decisions, how should we guess a $f \in \mathcal{F}$?

Question: Given the decisions, how should we a guess $f \in \mathcal{F}$?

(Formal) Setup:

- Ratings $\mathcal{X} = [0, 10]$, Outputs $\mathcal{Y} = \{0, 1\}$
- For some c^* , $f(x) = 1(x \ge c^*)$

 $\mathcal{F} = \{ 1(x \ge c) | c \in [0, 10] \}$

Answer: Find the worst watchable movie!

Algorithm $c' \leftarrow 10$ for $x \in S$:
if x watched and $x \leq c'$, set $c' \leftarrow x$ return c'

Question

- what is the runtime of *Algorithm*?
- what is the accuracy on S?
- what is the relationship between c' and c^* ?



Question: How does our guess perform?

 $c' \leftarrow 10$ for $x \in S$: if $x \ge c^*$ and $x \le c'$, set $c' \leftarrow x$ **return** c'

Troublesome region: $[c^*, c']$...

Idea: If m = |S| grows, $[c^*, c']$ becomes less significant...

 $\Pr[x \in [c^*, c']] \to 0$

Ideologically, this means we learn!



<u>Algorithm</u>

 $c' \leftarrow 10$ for $x \in S$: if $x \ge c^*$ and $x \le c'$, set $c' \leftarrow x$ return c'

Conjecture: If m = |S| grows, $[c^*, c']$ becomes less significant... $\Pr[x \in [c^*, c']] \to 0$

Let *d* be such that $\Pr[x \in [c^*, d]] \leq \epsilon$

Claim 1: If we receive any sample in $[c^*, d]$, then $\Pr[x \in [c^*, c']] \leq \epsilon$



<u>Algorithm</u>

 $c' \leftarrow 10$ for $x \in S$: if $x \ge c^*$ and $x \le c'$, set $c' \leftarrow x$ return c'

Conjecture: If m = |S| grows, $[c^*, c']$ becomes less significant... $\Pr[x \in [c^*, c']] \to 0$ Let d be such that $\Pr[x \in [c^*, d]] \le \epsilon$

Claim 1: If we receive any sample in $[c^*, d]$, then $\Pr[x \in [c^*, c']] \leq \epsilon$

...then $c' \leq d$, so $\Pr[x \in [c^*, c']] \leq \Pr[x \in [c^*, d]] \leq \epsilon$



<u>Algorithm</u>

 $c' \leftarrow 10$ for $x \in S$: if $x \ge c^*$ and $x \le c'$, set $c' \leftarrow x$ return c'

Conjecture: If m = |S| grows, $[c^*, c']$ becomes less significant... $\Pr[x \in [c^*, c']] \to 0$ Let d be such that $\Pr[x \in [c^*, d]] \le \epsilon$

Claim 1: If we receive any sample in $[c^*, d]$, then $\Pr[x \in [c^*, c']] \leq \epsilon$

Claim 2: All samples miss with probability $(1 - \epsilon)^m \le e^{-\epsilon m}$

... since samples are i.i.d., the misses compound *exponentially*!



<u>Algorithm</u>

 $c' \leftarrow 10$ for $x \in S$: if $x \ge c^*$ and $x \le c'$, set $c' \leftarrow x$ return c'

Conjecture: If m = |S| grows, $[c^*, c']$ becomes less significant... $\Pr[x \in [c^*, c']] \to 0$ Let d be such that $\Pr[x \in [c^*, d]] \le \epsilon$

Claim 1: If we receive *any* sample in $[c^*, d]$, then $\Pr[x \in [c^*, c']] \leq \epsilon$

Claim 2: All samples miss with probability $(1 - \epsilon)^m \le e^{-\epsilon m}$

Claim 3: If $m \ge \frac{\log(\frac{1}{\delta})}{\epsilon}$, then we get a bad sample with prob. $\le \delta$

... just by solving $e^{-\epsilon m} \leq \delta$



<u>Algorithm</u>

 $c' \leftarrow 10$ for $x \in S$: if $x \ge c^*$ and $x \le c'$, set $c' \leftarrow x$ return c'

Conjecture: If m = |S| grows, $[c^*, c']$ becomes less significant... $\Pr[x \in [c^*, c']] \to 0$

Let *d* be such that $\Pr[x \in [c^*, d]] \leq \epsilon$

Claim 1: If we receive any sample in $[c^*, d]$, then $\Pr[x \in [c^*, c']] \leq \epsilon$

Claim 2: All samples miss with probability $(1 - \epsilon)^m \le e^{-\epsilon m}$

Claim 3: If $m \ge \frac{\log(\frac{1}{\delta})}{\epsilon}$, then we get a bad sample with prob. $\le \delta$ **Conclusion:** With $m \ge \log\left(\frac{1}{\delta}\right)/\epsilon$ samples, our $f \in \mathcal{F}$ has error at most ϵ with probability at least $1 - \delta$!



(Proper) Learnability

Setup: input domain \mathcal{X} , output $\mathcal{Y} = \{0, 1\}$, **function class** $\mathcal{F} \subset \{f : \mathcal{X} \to \{0, 1\}\}$

Learner: Computes $m(\epsilon, \delta)$, requests a dataset of that size, and finds best $f \in \mathcal{F}$ for the sample.

Definition (Learnable Class): If, for any $\epsilon, \delta \in (0, 1)$ and any distribution $\mathcal{D}: \mathcal{X} \to [0, 1]$ there exists $m(\epsilon, \delta)$ such that protocol output f_{ERM} satisfies

$$\Pr_{x \sim \mathcal{D}}[f_{ERM}(x) \neq y] \le \epsilon$$

with probability at least $1 - \delta$, then \mathcal{F} is learnable.

- Distribution independent
- Non-asymptotic (finite samples)
- Provides a learning scheme
- Makes a statement about a whole function class



(Proper) Learnability

Definition (Learnable Class): If, for any $\epsilon, \delta \in (0, 1)$ and any distribution $\mathcal{D}: \mathcal{X} \to [0, 1]$ there exists $m(\epsilon, \delta)$ such that protocol output f_{ERM} satisfies

$$\Pr_{x \to \mathcal{D}}[f_{ERM}(x) \neq y] \le \epsilon$$

with probability at least $1 - \delta$, then \mathcal{F} is learnable.

- Distribution independent
- Non-asymptotic (finite samples)
- Provides a learning scheme
- Makes a statement about a whole function class

We've shown...

Theorem: The class of thresholds is learnable!

It's also true:

Theorem: Any finite hypothesis class is learnable. (In general, $m \ge \log \left(\frac{|\mathcal{F}|}{\delta}\right) / \epsilon \dots$)

Corollary: One-way functions are learnable



Efficiently Learnable

Definition (Efficient Learnability): A class that is learnable by an algorithm with time complexity polynomial in $\log(\frac{1}{\delta})$ and $1/\epsilon$ We've shown...

Theorem: The class of thresholds is learnable efficiently!

Proof: Recall worst-watchable Algorithm

- Learnable with $m = O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon}\right)$ samples
- Runs in O(m)

Theorem (rest of this talk): There exists a finite class that is *not* learnable efficiently!

Proof sketch: Suppose we could learn each digit in f^{-1} efficiently (binary class!), then we can reconstruct $f^{-1}(x)$ efficiently and with good probability, thus violating hardness.



Discrete Cube Root Assumption

Special case RSA function...

 $f_N(x) = x^3 \pmod{N}$ where N = pq is n digits, and $3 \nmid (p-1)(q-1)$

Theorem (RSA Correctness): f_N permutes Z_N^*

Proof: $3d \equiv 1 \pmod{\phi(N)}$ exists uniquely by construction. Inverse mapping $f_N^{-1}(y) = y^d \pmod{\phi(N)}$ by Euler Theorem.

Theorem (RSA Decoding): f_N^{-1} is computable efficiently, given d

Proof: Recall the squaring trick.

Assumption (RSA Security): f_N^{-1} is hard to find given only N. In particular, f_N is one-way

Suspicion:
$$\Pi = \{ \pi: Z_N^* \to Z_N^* \}$$
 is hard to learn



From the sample set, you can't find permutation structure

Bitwise Dissection

Special case RSA function...

 f_{N}^{-1}

2

 $f_N(x) = x^3 \pmod{N}$ where N = pq is *n* digits, and $3 \nmid (p-1)(q-1)$

Suspicion: $\Pi = \{ \pi: Z_N^* \to Z_N^* \}$ is hard to learn

Convert to a hard *binary* class – isolate f_N^{-1} digit by digit!

 Z_{15}^{*}

8



y
 1
 2
 4
 7
 8
 11
 13
 14

$$f_{N,0}^{-1}$$
 0
 1
 1
 0
 0
 1
 0
 1

 $f_{N,i}^{-1}(y) \coloneqq$ i-th digit $f_N^{-1}(y)$

Observation: Learning \mathcal{F} means learning a digit in the inverse

${\mathcal F}$ cannot be efficiently learned

Recall \mathcal{F} from the previous slide, $\{f_{N,i}^{-1}(y) | 1 \le i \le n\}$

Observation: ${\mathcal{F}}$ has finite size

Proof: *N*! permutations *n* digit functions each.

Theorem: If \mathcal{F} is efficiently learnable, then we can reconstruct F_N^{-1} . *Proof*: Algorithm \mathcal{A} that (ϵ, δ) -learns any digit function with $m(\epsilon, \delta)$ many samples

 $\begin{array}{l} \textbf{Algorithm} \\ \text{set } \epsilon \leftarrow \frac{1}{n^2}, \delta \leftarrow O(\frac{1}{n}) \\ \text{generate } m(\epsilon, \delta) \text{ many samples } (y, f_N^{-1}(y)) \\ \text{parse each sample into } \{f_{N,0}^{-1}(y), f_{N,1}^{-1}(y), \dots, f_{N,n-1}^{-1}(y)\} \\ \text{for } i \in [n]: \\ \text{run subroutine } \mathcal{A} \text{ to learn } (y, f_{N,i}^{-1}(y)) \text{ and get } h_i^{-1} \\ \text{return } (h_1^{-1}, h_2^{-1}, \dots, h_n^{-1}) \end{array}$

All steps are efficient!



${\mathcal F}$ cannot be efficiently learned

Recall \mathcal{F} from the previous slide, $\{f_{N,i}^{-1}(y) | 1 \le i \le n\}$

Theorem: If \mathcal{F} is efficiently learnable, then we can reconstruct F_N^{-1} .

 $\begin{array}{l} \textbf{Algorithm} \\ \text{set } \epsilon \leftarrow \frac{1}{n^2}, \delta \leftarrow O(\frac{1}{n}) \\ \text{generate } m(\epsilon, \delta) \text{ many samples } (y, f_N^{-1}(y)) \\ \text{parse each sample into } \{f_{N,0}^{-1}(y), f_{N,1}^{-1}(y), \dots, f_{N,n-1}^{-1}(y)\} \\ \text{for } i \in [n]: \\ \text{run subroutine } \mathcal{A} \text{ to learn } (y, f_{N,i}^{-1}(y)) \text{ and get } h_i^{-1} \\ \textbf{return } (h_1^{-1}, h_2^{-1}, \dots, h_n^{-1}) \end{array}$

For each *i*,
$$\Pr[h_i^{-1}(y') \neq f_{N,i}^{-1}(y')] \le \epsilon = 1/n^2$$

 $\Pr[h_1^{-1}(y')||h_2^{-1}(y')|| \cdots ||h_{n-1}^{-1}(y') \neq F_N^{-1}(y)] \le n \cdot \epsilon = \frac{1}{n}$

So, we have learned a one-way function...

Conclusion

Recall \mathcal{F} from the previous slide, $\{f_{N,i}^{-1}(y) | 1 \le i \le n\}$

Theorem: If \mathcal{F} is efficiently learnable, then we can reconstruct F_N^{-1} .

Assumption: It's not possible to reconstruct F_N^{-1} efficiently

Conclusion: Under DCRA, the hypothesis class \mathcal{F} is not efficiently learnable

