CSE 513T Theory of Artificial Intelligence & Machine Learning September 27th, 2016

Lecture 9

Lecturer: Brendan Juba *Scribe: Zhuoer Shen* *Edit: Shi Shu*

1 Today's Topics

1. Cryptography v.s. Learning
2. The Discrete Cube Root Assumption
3. Representation independent hardness of learning Boolean Circuits

2 Recall: Proper vs Improper Learning

3-Term DNF was not learnable using a 3-Term DNF as a representation. Unless NP has randomized poly-time algorithms, 3-Term DNF is learnable using a 3 CNF as a representation.

Q: Do there exist representations that are not efficiently learnable regardless of the hypothesis class that is used by the learning algorithm?
A: Yes. Boolean circuits are not PAC learnable.

We won't show "NP-hardness" (In fact, it probably is not true). Instead, we will show that learning Boolean Circuits would break modern Cryptography.

3 Cryptography

3.1 Meta-Theorem[1]

Claim 1 (Informal statement) If cryptography is possible, then there is no PAC-learning algorithm for Boolean circuits.

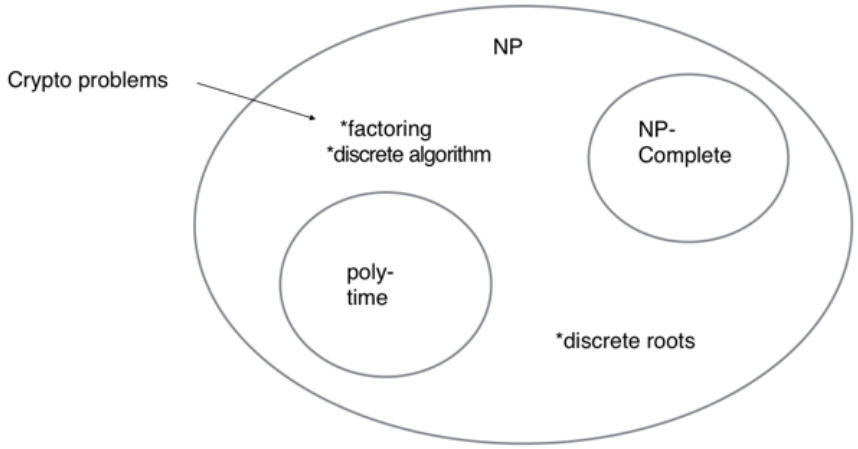
Informally in the reverse direction, [2] hard learning problems can (often) be used to design secure encryption such as pseudo random number generator.

For example, suppose it is very hard to learn for almost all $c \in C$. Encrypt "1", choose $x^{(1)}, x^{(2)}, ..., x^{(m)} \in \{0,1\}^n$ uniformly at random send $(x^{(1)}, c(x^{(1)})), ..., (x^{(m)}, c(x^m))$. Encrypt "0", choose $(x^{(1)}, b^{(1)}), ..., (x^{(m)}, b^{(m)}) \in \{0,1\}^{m+1}$ send $(x^{(1)}, b^{(1)}), ..., (x^{(m)}, b^{(m)})$.

It's hard to distinguish these two random vectors unless you hold c. (Given c, it's easy to distinguish them) only issues, raw "hard", how do I choose c (at random), etc.

[1] Goldreich-Goldwasses Micali '86 & Hastad Impagliazza, Levin, Luby '99
[2] Blum-Frust-Kearns Lipton '93 & Roger '09

3.2 Cryptographic Problems



The problems useful for cryptography are not believed to be NP complete; it must be easy to pick instances that are hard to solve, certain famous mathematical problems seem to have this structure. We'll show solving our learning problem will allow us to solve one of these famous problems.

4 The Discrete Cube Root Problem

(Special case of RSA with e=3, and the security of RSA is based on the assumption that the Discrete Cube Root is hard)

4.1 Preliminary Definitions

Let p and q be distinct n-bit primes. Such that 3 does not divide $(p-1)(q-1)$. And let $N = p*q$. Then it exist $d \leq (p-1)(q-1)$ s.t. for every nonzero $x \leq N$ s.t. neither p nor q divides x, $x^{3d} = x \ mod \ N$ (remainders after we divide by N)

Thus we have: $x \to x^3 \ mod \ N$ is a bijection.
Sketch of Proof: (Injection):Suppose $x^3 = y^3 \ mod \ N$. Then $x = x^{3d} = y \ mod \ N$. So $x = y$
(Surjection): Take any x s.t. neither p nor q divides x. Consider $x^d \ mod \ N$, then $x \ mod \ N = x$. So x is in the range of $x \to x^3$.

4.2 The Problem

Discrete Cube Root Problem. Let p and q be two primes. And 3 does not divide $(p-1)(q-1)$. Let $N = pq$. Then Let $x \in Z^*$ and $x \leq N$. Let $f_N(x) = x^3 \ mod \ N$. The problem is that given N and $y = f_N(x)$, find x.

4.3 Discrete Cube Root Assumption

For any polynomial $p(n)$, there is no algorithm such that,

1. That runs in time $p(n)$ and
2. On input $N = pq$ from two randomly chosen n-bit primes p and q, such that 3 does not divide $(p-1)(q-1)$ and input $y \neq 0$ s.t. neither p nor q can divide y, chose uniformly at random and returns x, such that $x^3 \ mod \ N = y$. w.p$\geq \frac{1}{p(n)}$ (over p, q, y and algorithm choices)

Suppose we wish to learn to compute cube roots *mod* such N. Then, we are given examples $(y^{(1)}, y^{(1)d})$, ..., $(y^{(m)}, y^{(m)d})$) ($y^{(1)}$ (chosen uniformly at random) could access to these examples make it possible for us to learn to compute cube roots *mod* N? Suppose we had an efficient PAC-learning algorithm that could learn the function $\{y \to y^d \ mod \ N\}$

Claim : No such alogrithm exist. It would violate the Discrete Cube Root Assumption.
Proof: Since $x \to x^3 \ mod \ N$, then x is also distributed uniformly at random. So let me generate my data by choosing $x^{(1)}, ..., x^{(m)}$ uniformly at random, and use $(x^{(1)3}, x^{(1)}), ..., (x^{(m)3}, x^{(m)}) = (y^{(1)}, y^{(1)d}), ..., (y^{(m)}, y^{(m)d})$. If such algorithm existed, it would then allow us to compute hypothesis h s.t. $Pr[h(y) = y^d] \geq 1 - \varepsilon$ w.p $\geq 1 - \delta$ over any δ. (With failure $prob \leq \varepsilon + \delta$, $1 - (\varepsilon + \delta) \geq \frac{1}{p(n)}$).
Therefore, any representation class that includes the function that compute (the individual bits of) $y \to y \ mod \ N$ cannot be efficiently learned under the Discrete Cube Root Assumption. So we only need to argue that for fixed $N = p, q$ that there are polynomial size circuits to compute $y \to y^d \ mod \ N$.
Let's show that there is a polynomial time algorithm that would invoke our proposition.

Algorithm 1 Algorithm $\to$ circuits

Fixing N
On input x
for $j = 1, ..., 2n$ **do**
  compute $x^2 \ mod \ N$ by $(x^{2^{j-1}} \ mod \ N)(x^{2^{j-1}} \ mod \ N) \ mod \ N$
put $y \leftarrow 1$
for $j = 0, ..., 2n$ if the jth bit of d is 1 **do**
  put $y \leftarrow y \cdot 2^j \ mod \ N$
Return y

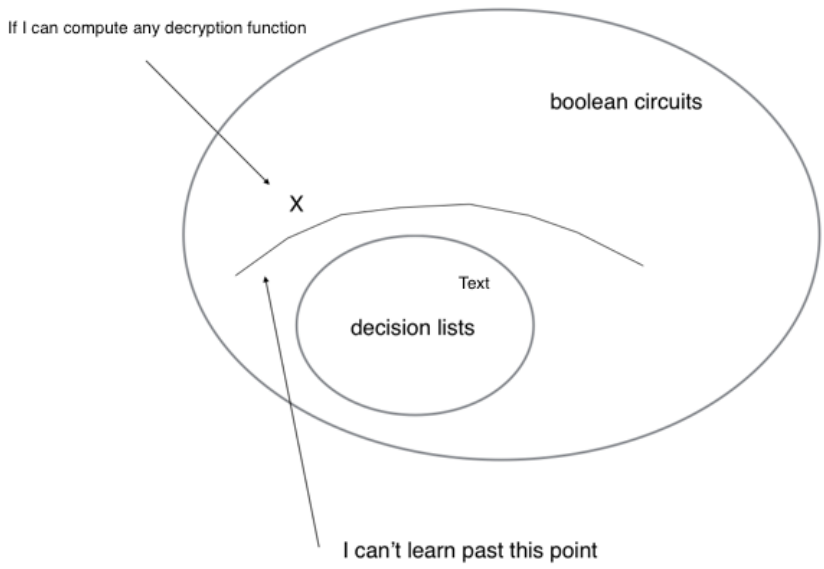
Notice $d = \sum_{j=0}^{2n} d_j 2^j$.
So at the end, $y = \prod_{i=0}^{2n} x^{d_j 2^j} = x^{\sum_{j=0}^{2n} d_j 2^j} \ mod \ N$
So yes, $y = x^d \ mod \ N$
The existence of this algorithm guarantees that there is some polynomial time algorithm computing each of the bits of $y \to y^d \ mod \ N$ accurately with a high probability. Therefore contradicting the Discrete Cube Root Assumption.

Theorem: If there is an efficient PAC learning algorithm for the class of Boolean circuits, then the Discrete Cube root assumption is false.

Informally, this same algorithm works for any problem that can be used as the basis for public key cryptography (the "public key" allows us to generate our own "sorted examples" of decryption).

The class that ends up being unlearnable is any class of functions that can compute the decryption function. Since if learnable, that would contradict with the Discrete Cube Root Assumption.

The really bad news... cryptography is surprisingly efficient.



For example Klivans-Sherstov.[3]
ANDs of "linear threshold functions" ($[\sum w_i x_i \geq \theta]$) cannot be learned under crypto assumptions.

[3]http://www.cs.utexas.edu/ klivans/crypto-hs.pdf