Computational Learning Theory
5 : Cryptographic Hardness of Learning

Lecturer: Varun Kanade

We have seen a few algorithms in the PAC learning framework for concept classes such as those of conjunctions, decision lists and linear halfspaces. We've also studied the Occam principle that "short consistent hypotheses" generalise well on unseen data. For finite hypothesis classes the sample complexity scales polynomially with $\log |H|$, $1/\epsilon$ and $1/\delta$. When using infinite hypothesis classes, the Vapnik Chervonenkis (VC) dimension plays a similar role as $\log |H|$. We've seen upper and lower bounds on sample complexity in terms of VC-dimension that almost match. In particular, provided one can identify a hypothesis that is consistent with the observed data (as long as the sample size is large enough as a function of the VC dimension, ϵ and δ), we obtain a hypothesis that has error bounded by ϵ with respect to the target concept and distribution.

Thus, in a sense the VC-dimension captures *learnability* exactly, if sample complexity is the only thing we care about. However, when we consider computational complexity the picture is considerably different. We've already shown that there are concept classes for which finding *proper* consistent learners is NP-hard. In the case of 3-term DNF formulae, we can avoid this NP-hardness by identifying the output hypothesis from a larger concept class, that of 3-CNF formulae. One may wonder, whether this is always the case, *i.e.,* can we always identify a larger hypothesis class from which we can identify a consistent learner?

In this lecture, we'll answer this question in the negative, provided a certain widely believed assumption in cryptography holds. We will show that there are concept classes that cannot be *efficiently* PAC-learnt, even in the case *improper* learning where the output hypothesis is allowed to come from any polynomially evaluatable hypothesis class.

1 The Discrete Cube Root Problem

Let p and q be two large primes that require roughly the same number of bits to represent. Furthermore, we'll assume that these primes are of the form $3k+2$. Let $N = pq$ be the product of these primes. It is widely believed that factoring such an N, when p and q are chosen to be random n bit primes, cannot be performed in polynomial time. Let φ denote Euler's totient function, then we have $\varphi(N) = (p-1)(q-1)$. As p and q are chosen to be of the form $3k+2$, 3 does not divide $\varphi(N)$.

Let $\mathbb{Z}_N^* = \{i \mid 0 < i < N, \gcd(i, N) = 1\}$. It is well-known that $\mathbb{Z}_N^*$ forms a group under the operation of multiplication modulo N. We consider a function $f_N : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ defined as $f_N(y) = y^3 \mod N$. As 3 does not divide $\varphi(N)$, it is straightforward to observe that f_N is a bijection. As $\gcd(3, \varphi(N)) = 1$, there exist $d, d' \geq 1$ such that $3d = \varphi(N)d' + 1$ (the existence of d, d' can be shown by a constructive proof of Euclid's algorithm to obtain the gcd). Then, we have,

$$(f_N(y))^d = y^{3d} = y^{\varphi(N)d'+1} = y \mod N$$

The last equality follows as $y^{\varphi(N)} = 1 \mod N$ for all $y \in \mathbb{Z}_N^*$.

Definition 1 (Discrete Cube Root Problem)**.** *Let two n-bit primes p and q of the form $3k+2$, and let $N = pq$. Let $\varphi(N) = (p-1)(q-1)$ and note that 3 does not divide $\varphi(N)$. Given N and $x \in \mathbb{Z}_N^*$ as input, output $y \in \mathbb{Z}_N^*$, such that $y^3 = x \mod N$.*

Observe that if we can factorise N, the discrete cuberoot problem is easy to solve. We can simply obtain $\varphi(N)$ and then find d such that $3d = 1 \mod \varphi(N)$ using Euclid's algorithm. However, factoring N is believed to be hard in general and in fact, no polynomial-time algorithm that finds the discrete cube root is known. Note that in this case, polynomial-time means polynomial in n, not N. The discrete cube root problem is also widely believed to be computationally intractable. We define the formal hardness assumption and use this to show that there are concept classes that are computationally hard to learn.

Definition 2 (Discrete Cube Root Assumption (DCRA))**.** *For any polynomial $P(\cdot)$, there does not exist any algorithm, A, that runs in time $P(n)$ and on input N and x, where N is the product of two random n bit primes of the form $3k+2$ and x is chosen randomly from $\mathbb{Z}_N^*$, outputs $y \in \mathbb{Z}_N^*$ that with probability at least $1/P(n)$ satisfies $y^3 = x \mod N$. The probability is over the random draws of p, q, x and any internal randomisation of A.*

Although this is not the main point of this course, let us quickly note how this assumption may be used in cryptography. The integer N is the public key, and any message that can be encoded as an element of $\mathbb{Z}_N^*$ can be encrypted simply by taking its cube modulo N. Under the discrete cuberoot assumption, this cannot be decrypted, except if one has access to d, such that $3d = 1 \mod \varphi(N)$. The integer d is the private key.

2 A learning problem based on DCRA

Let us try to phrase the question of finding the cuberoot of $x \in \mathbb{Z}_N^*$ as a learning question. Let us suppose that we have access to a sample, $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, where $y_i^3 = x_i \mod N$ for $i = 1, \ldots, m$, and x_i are drawn uniformly at random from $\mathbb{Z}_N^*$. The learning question is given such examples, can we obtain $h : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, such that for x drawn uniformly at random from $\mathbb{Z}_N^*$, it holds with probability at least $1 - \delta$, that $\mathbb{P}\left[(h(x))^3 \neq x \mod N\right] \leq \epsilon$?

How can these pairs (x_i, y_i) help? It is easy to see that they cannot help, as it is easy to generate these pairs ourselves. For instance, although finding the cuberoot is hard, finding the cube is easy. As f_N is a bijection, we can choose $y_i \in \mathbb{Z}_N^*$ uniformly at random, and then set $x_i = y_i^3 \mod N$. Note that this implies that the distribution of x_i is uniform over $\mathbb{Z}_N^*$. Thus, clearly access to random examples of the form (x_i, y_i) where x_i is drawn from the uniform distribution over $\mathbb{Z}_N^*$ and $y_i^3 = x_i \mod N$, can't help to find $h : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, that satisfies $\mathbb{P}\left[(h(x))^3 \neq x \mod N\right] \leq \epsilon$, under the DCRA.

This almost fits into our notion of PAC learning, except that the output of the target function is not in $\{0, 1\}$. This can be fixed rather easily. We know that the output of f_N^{-1} is some $2n$ bit string. Thus, we can consider $2n$ different target functions, $(f_{N,i}^{-1})_{i=1}^{2n}$, where $f_{N,i}^{-1}$ is a function which outputs the i^{th} bit of the function f_N^{-1}. If we could learn all the function, $f_{N,i}^{-1}$ to accuracy $\frac{\epsilon}{2n}$, then we can reconstruct f_N^{-1} to accuracy ϵ. Thus, if learning f_N^{-1} is hard, then at least one of the boolean functions $(f_{N,i}^{-1})_{i=1}^{2n}$ must also be hard to learn.

2.1 A hard-to-learn concept class

So far, we've established that if we choose random n bit primes p and q of the form $3k+2$, there exists a boolean function, $f_{N,i}^{-1}$, such that if we get labelled examples from a *specific* distribution D over $2n$ bit strings, *viz.* the uniform distribution over bit representatations of elements in $\mathbb{Z}_N^*$, we cannot output a (polynomially evaluatable) hypothesis h, such that $\mathbb{P}_{x \sim D}\left[f_{N,i}^{-1}(x) \neq h(x)\right] \leq \frac{\epsilon}{2n}$. If we can identify a class, C, such that $f_{N,i}^{-1} \in C_{2n}$, then this also implies that the class C is not PAC-learnable.

Let us try and understand what such a concept class could be. First, we note that if d is known, there is a rather simple polynomial time algorithm to output $f_N^{-1}(x)$. All we need

to do is perform the operation $x^d \mod N$. Naïvely computing x^d is not efficient as d may be as large as $\varphi(N)$, *i.e.,* d may itself be $2n$ bit long. The first thing we need to ensure is that all operations are repeatedly performed modulo N; this way none of the representations get too large. The second is that we start by computing, $x \mod N, x^2 \mod N, x^4 \mod N, x^8 \mod N, \ldots, x^{2^{\lfloor \log \varphi(N) \rfloor}} \mod N$, *i.e.,* we compute $x^{2^i} \mod N$ for $i = 0, 1 \ldots, \lfloor \log \varphi(N) \rfloor$. To obtain $x^d \mod N$, we simply take the product of the terms $x^{2^i} \mod N$ such that the i^{th} bit of d is 1. This shows that there exists a circuit of polynomial size that computes f_N^{-1} where d is hard-wired into the circuit itself. In particular, this also implies that there exist polynomial-size circuits for $f_{N,i}^{-1}$ for all $i = 1, \ldots, 2n$. This gives us the following result.

Theorem 3. *There exists a polynomial $P(\cdot)$, such that class of circuits, C, where C_n consists of circuits of size at most $P(n)$, is not PAC-learnable under the discrete cube root assumption.*

2.2 Reducing the depth

We've established that under DCRA, PAC-learning polynomial-sized circuits is hard. However, in a way, this is the weakest such result one could hope for. Polynomial-sized circuits are in some sense the most expressive class that we could ever hope to learn. The question is whether there exist significantly weaker concept classes that are also hard to learn. We will outline a proof that one such class, that of circuits whose depth is only logarithmic in the number of inputs, is also hard to PAC-learn under DCRA. The complete proof requires showing how low-depth circuits for repeated multiplication and division can be designed; we will not see these constructions here as they are quite involved. We will highlight the basic idea and point to the appropriate references for complete details.

One of the reasons why the circuit described above is deep (it has depth $\Theta(n)$) is that it requires computing the powers x^{2^i} for $i = 0, 1, \ldots, \lfloor \log \varphi(N) \rfloor$; the reason being that $x^{2^{i+1}} \mod N$ can only be computed after $x^{2^i} \mod N$ has been computed. However, we note that this computation does not require the knowledge of d, the secret key, at all! So, if instead of being given the input x, we were given a longer string of length $(2n)^2$ as input, the concatenation of $(x \mod N, x^2 \mod N, x^4 \mod N, \ldots, x^{2^{\lfloor \log \varphi(N) \rfloor}} \mod N)$, the question of learning $f_{N,i}^{-1}$ would still remainin tractable under the DCRA, as the additional input just represents something we could have computed ourselves in polynomial time. Note that a hard to learn target function and distribution now would involve having a distribution over strings of length $(2n)^2$, where the first $2n$ bit represent $x \in \mathbb{Z}_N^*$ chosen at the uniformly at random and the remaining bits are the powers, $x^{2^i} \mod N$, $i = 1, 2, \ldots, \lfloor \log \varphi(N) \rfloor$. It is relatively straightforward to show that there exists a circuit of depth $O((\log n)^2)$ that when given as input $(x \mod N, x^2 \mod N, \ldots, x^{2^{\lfloor \log \varphi(N) \rfloor}} \mod N)$ outputs $f_N^{-1}(x)$. To show that there is a circuit of depth $O(\log n)$ requires more work using the techniques of Beame et al. (1986). To summarise, we have the following theorem.

Theorem 4. *There exists a constant $\kappa_0 > 0$, such that the class of circuits, C, where C_n consists of circuits on n inputs with depth bounded by $\kappa_0 \log n$ is not PAC-learnable under DCRA.*

3 Bibliographic Notes

The material covered in this lecture is presented in greater detail in the textbook by Kearns and Vazirani (1994, Chap. 6). Further details, including complete proofs and reductions showing cryptographic hardness of learning other concept classes such as finite automata appear in the original paper by Kearns and Valiant (1994). The proof that the class of polynomial-size circuits cannot be PAC-learnt if one-way functions exist, even when membership queries are qllowed (cf. Lec. 6), first appeared in the work of Goldreich et al. (1984). The assumption that one-way functions exist is much weaker than the discrete cube root assumption; indeed, it is hardly

conceivable to have any cryptography at all if one-way functions don't exist. On the other hand, if the discrete cube root assumption were to be untrue, it would simply question the RSA cryptosystem, but not rule out the existence of other kinds of cryptosystems.

In general, making stronger assumptions leads to stronger hardness of PAC-learning results. Recently, there has been work using different kinds of assumptions, not directly related to cryptography, to establish hardness of PAC-learning of much smaller classes such as DNF (Daniely et al., 2014; Daniely and Shalev-Shwartz, 2014; Daniely, 2015). However, it is worth bearing in mind that the more obscure assumptions may not have been thoroughly tested; for example, recently Allen et al. (2015) showed that one of the assumptions made by Daniely et al. (2014) was in fact not true.

References

Sarah R Allen, Ryan ODonnell, and David Witmer. How to refute a random csp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 689–708. IEEE, 2015.

Paul W Beame, Stephen A Cook, and H James Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.

Amit Daniely. Complexity theoretic limitations on learning halfspaces. *arXiv preprint arXiv:1505.05800*, 2015.

Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning DNFs. *CoRR, abs/1404.3378*, 1(2.1):2–1, 2014.

Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 441–448, New York, NY, USA, 2014. ACM.

Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 464–479. IEEE, 1984.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

Michael J. Kearns and Umesh K. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1994.