Mathematical Foundations of Machine Learning (CS 4783/5783)

Lecture 19: Representation Free Computational Complexity of Learning

1 Improperly Learning 3-Term-DNF

In the past lecture we saw that proper learning of a 3-term-DNF is hard. That is, if we needed a learning algorithm to return a 3-Term-DNF that was Probably approximately right, based on samples, this problem we showed was computationally hard. But what if we are allowed to use improper learning, where the learning algorithm can return models outside of the class of target models $\mathcal{F}$?

Consider an 3-Term-DNF given by:

$$f(x) = (x[1] \wedge x[2] \wedge \neg x[3]) \vee (x[4] \wedge x[5]) \vee (x[6] \wedge x[7])$$

This function can be alternatively written as:

$$\begin{aligned} f(x) =& (x[1] \vee x[4] \vee x[6]) \wedge (x[1] \vee x[4] \vee x[7]) \wedge (x[1] \vee x[5] \vee x[6]) \wedge (x[1] \vee x[5] \vee x[7]) \wedge (x[2] \vee x[4] \vee x[6]) \\ & \wedge (x[2] \vee x[4] \vee x[7]) \wedge (x[2] \vee x[5] \vee x[6]) \wedge (x[2] \vee x[5] \vee x[7]) \wedge (\neg x[3] \vee x[4] \vee x[6]) \\ & \wedge (\neg x[3] \vee x[4] \vee x[7]) \wedge (\neg x[3] \vee x[5] \vee x[6]) \wedge (\neg x[3] \vee x[5] \vee x[7]) \end{aligned}$$

That is, as a conjunction of many OR's of three terms. In this, alternative representation, each of the three term or's can be one of $\binom{2m}{3} = O(m^3)$ possibilities ($2m$ because of possible negations). Each one of these $O(m^3)$ Or's are either included in the conjunction or not. Hence the total number of such conjunction is order $2^{O(m^3)}$. Hence if we take $\overline{\mathcal{F}}$ = Conjunction of 3-term-Or's then $\mathrm{VC}(\overline{\mathcal{F}}) \le \log |\overline{\mathcal{F}}| \le O(m^3)$.

The nice part is that, computationally, finding ERM over $\overline{\mathcal{F}}$ is easy! Why?

Answer: Start with the list of all $O(m^3)$ 3 term Or's possible. Take a give sample, and eliminate any 3-term-OR that is not consistent with the sample. The conjunction of the remaining terms in the list is consistent with the entire sample and does belong to $\overline{\mathcal{F}}$ and hence is an ERM.
But this means learning 3-TERM-DNF by using $\overline{\mathcal{F}}$ is computationally easy and requires $O(m^3 \log(1/\delta)/\epsilon^2)$ samples. It is clearly also polynomial time.

So are there models where improperly learning the model is also computationally hard but sample complexity is easy? Of course there are. The set of all models over m-bit variables that are polynomial time computable, it turns out have polynomial in m VC dimension and hence are sample efficiently learnable. But this class for instance is not efficiently learnable even improperly. But to prove such hardness result requires us to step out beyond NP style reductions.

2 A Quick Primer to Cryptographic Hardness

An area of the CS field where computational hardness of certain problems is given a positive spin is in Cryptography. The idea at a birds-eye-view is that we have a simple m bit, plain text message, M which we encode to get a code C. This encoding has to be easy (poly-time) given the Key K. Next, on the side of the real receiver, given the coded message C and the decoding key D, decrypting the message M should be easy (poly-time). On the other hard, we need to ensure that for a 3rd party that does not know D, cracking the code and retrieving the original message M from C must be hard. But hard in what sense? Well we could say in the worst case sense, that there is no poly-time algo for cracking for all C's but this would not suffice since this would mean most messages can be cracked. We cant hope to have a result like no poly-time algorithm for all messages because returning a constant message would work for at least one code. Hence, the right notion is: no poly-time algorithm works for random C.

Typically, the key K is a public key that everyone knows. Given this, we can think of our Crypto system as being secure (and usable) if:

1. For every public key $K \in \mathcal{K}$, the coded message $C = f_K(M)$ can be computed in time polynomial in number of message m.

2. For every key K, given the corresponding decoding key $D = D(K)$, we can compute $\text{Decode}(C, D) = f_K^{-1}(C) = M$ in poly-time.

3. There is no randomized poly-time algorithm $\mathbf{B}$ to break the code such that, when messages $M \sim \text{Unif}\{0,1\}^m$ are drawn at random,

$$P\left(\mathbf{B}(K, f_K(M)) = M\right) \geq \frac{1}{\text{poly}(m)}$$

3 From Cryptography to Hardness of Improper Learning

Define a model class

$$\mathcal{F}_\mathcal{K} = \{(C, i) \mapsto f_K^{-1}(C)[i] : K \in \mathcal{K}, i \in [m]\}$$

Claim 1. *If the cryptography system is secure, then the model class $\mathcal{F}_\mathcal{K}$ is not efficiently PAC learnable.*

Proof Sketch. We will prove by contradiction. Say the model class was indeed efficiently PAC learnable, then given the efficient algorithm $\mathbf{A}$ for learning, we do the following:
Set, $\epsilon = \Delta/2$ and $\delta = \Delta/2m$. Now, for each $i \in [m]$, collect (polynomial) n samples
$S^i = ((f_K(x_1^i), i), y_1^i), \ldots, ((f_K(x_n^i), i), y_n^i))$ by drawing the x's from distribution $D_X = \text{Unif}\{0,1\}^m$ and set each $y_t^i = x_t^i[i]$. Set the learnt model $h_i = \mathbf{A}(S^i)$ for each i. Now given a code C return, $(h_1(C, 1), \ldots, h_m(C, m))$. If learning worked, then, for each $i \in [m]$ with probability at least $1 - \Delta/2m$,

$$P\left(h_i(f_K(M)) \neq M[i]\right) \leq \frac{\Delta}{2}$$

Taking union bound over i we have that with probability at least $1 - \Delta/2$,

$$P\left(\exists i \in [m] : h_i(f_K(M)) \neq M[i]\right) \leq \frac{\Delta}{2}$$

Or in other words, with probability at least $1 - \Delta/2$,

$$P\left((h_1(f_K(M)), \dots, h_m(f_K(M))) \neq M\right) \leq \frac{\Delta}{2m}$$

Hence taking expectation over randomization of the algorithm we can conclude that:

$$P\left((h_1(f_K(M)), \dots, h_m(f_K(M))) \neq M\right) \leq \frac{\Delta}{2} + \Delta/2 = \Delta$$

Hence if $\mathbf{B}(K, f_K(M)) = (h_1(f_K(M)), \dots, h_m(f_K(M)))$ then, we have that:

$$P\left(\mathbf{B}(K, f_K(M)) = M\right) \geq \Delta$$

Taking $\Delta = 1/\text{poly}(m)$, we can conclude that if $\mathcal{F}_{\mathcal{K}}$ is efficiently PAC Learnable, then the crypto system is not secure.

□

4 Hardness Of Learning and Discrete Cube Root Problem

RSA is a famous algorithm that has found many practical uses for secure cryptographic systems. In RSA, the receiver picks two large random numbers p and q and sets $K = p \cdot q$. Next, we compute $\phi(K) = (p-1)(q-1)$ and then choose numbers E, D such that $E \cdot D = 1(\text{mod } \phi(N))$. Finally, (e, K) are released as public key. D is used as private key. Encryption is computed as follows: First think of M the m-bit message as an m bit number. Next, we encrypt as:

$$C = f_K(M) = M^e \text{ (mod } K)$$

Decryption is done as:

$$C^D(\text{mod } K) = (M^e)^D(\text{mod } K) = M(\text{mod } K)$$

It is believed that RSA is hard to crack (ie. secure). Further, if one has to decode D, given access to public key, it can be shown that recovering D is equivalent to factoring which is again believed to be hard.

A special version of RSA that is also believed to be hard is the case when $E = 3$. In this case, cracking RSA is known as the discrete cube root problem. In this case, encoding just requires raising message to the third power and decoding requires that we find a $\tilde{M}$ such that $C = \tilde{M}^3(\text{mod } K)$. Hence we are trying to find a cube root of code C modulo the key K. This problem is believed to be hard as well. The nice thing about this problem is that encoding not only requires poly time but can in fact be carried out by depth $O(m)$ circuits and with a little modification, we can in fact show that it suffices to consider depth $O(\log m)$ circuits with at most $O(m^2)$ computations. This is nice because it tells us that not only is the set of all poly time computable functions (that has VC that is poly m) but in fact the set of all $O(m^3)$ time computable functions are hard to learn. An immediate corollary of this is in fact that that $O(\log(m)$ depth neural networks are not Efficiently PAC learnable.