Mathematical Foundations of Machine Learning (CS 4783/5783)

Lecture 16: Computational Complexity of Learning

1 Setup

So far we only looked at the statistical complexity or sample complexity of learning problems. We did not pay attention to whether a problem is computationally efficiently learnable. In the next couple lectures we will look into this question more carefully. To start with we will in fact just focus on the realizable PAC setting. That is, where inout instances x_t's are drawn from a fixed distribution and $y_t = f^*(x_t)$ for some $f^* \in \mathcal{F}$ and $\mathcal{F}$ is a binary model class. First notice that computational complexity is at least as large as sample complexity because we would at least need to read in the minimum required samples. To be more precise, say we can represent the input instances x_t's using m bits, then minimum computation time for learning will be m times the sample complexity since we need to read as many samples as sample complexity $n(\epsilon, \delta)$ and each instance is m bits long. To be precise, we can define a model class $\mathcal{F}$ to be efficiently PAC learnable using the following definition.

Definition 1. *A class of models $\mathcal{F}$ is efficiently PAC Learnable if there exists a learning algorithm such that for any target model $f^* \in \mathcal{F}$ and any distribution D_X over instance space $\mathcal{X}$ and for any $\delta, \epsilon > 0$, the algorithm returns a model $\hat{f}_S$ s.t. with probability at least $1 - \delta$ over draw of samples*

$$P_{x \sim D_X}(\hat{f}_S(x) \neq f^*(x)) \leq \epsilon$$

and the algorithm runs in time $\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta}, m)$ *where m is the instance length.*

Basically to the PAC learnable definition we add the constraint that the learning algorithm has to also run in polynomial time in the relevant parameters. A list of models we can learn efficiently (in the realizable setting) are, half-spaces, conjugation of literals etc.

We will further say a problem is **properly efficiently PAC learnable** if the learning algorithm returns the model $\hat{f}_S \in \mathcal{F}$. That is the learning algorithm returns a model from the model class we are considering (like ERM algorithm).
Say we were able to solve the optimization or search problem of finding $f \in \mathcal{F}$ s.t.

$$\hat{L}_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \mathbf{1}\{f(x) \neq y\} = 0$$

efficiently, that is in time $\text{poly}(|S|, m)$, then it is clear that the problem is efficiently PAC learnable. This is because in such a case we are able to compute the ERM efficiently. For efficient proper PAC learning, it turns out the converse is also true. Given a model class $\mathcal{F}$, the consistency problem given a sample $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is the decision problem of computing $\text{CONS}_{\mathcal{F}}(S) = 1$ if and only if $\exists f \in \mathcal{F}, \text{s.t.} \hat{L}_S(f) = 0$. That is deciding if there is a model in the class $\mathcal{F}$ that makes 0

training error (clearly this is easier than finding such a model if one exists). We will basically argue that if for a class of models $\mathcal{F}$, the learning problem is **efficiently, properly PAC learnable**, then we will show that the consistency problem $\text{CONS}_{\mathcal{F}}$ is in the class RP (problems that can be solved in Randomized polynomial time). In other words, taking the converse, we will be claiming that if for a model class we can show that $\text{CONS}_{\mathcal{F}}$ is hard (say it belongs to NP class) then we can prove that such problems are not efficiently properly PAC learnable (unless RP = NP which we don't believe is true).

Theorem 1. *If a class of models $\mathcal{F}$ is* **efficiently properly PAC learnable***, then* $\text{CONS}_{\mathcal{F}} \in \text{RP}$*, that is, there exists a randomized algorithm C that runs in poly $|S|, m$ time and is such that for any input sample S:*

- $P(C(S) = 1 | \text{CONS}_{\mathcal{F}}(S) = 1) \geq 3/4$
- $P(C(S) = 0 | \text{CONS}_{\mathcal{F}}(S) = 0) = 1$

Proof Sketch. Given sample S, consider distribution $D = \text{Unif}(S)$. Now if the problem is efficiently properly PAC learnable, by our premise, there exists an algorithm, say $\mathcal{A}$ such that if we run $\mathcal{A}(D, \delta, \epsilon)$, it will return a model in $\mathcal{F}$ that with probability at least $1 - \delta$ has error smaller than ϵ. Run this algorithm to get $\hat{f} = \mathcal{A}(\text{Unif}(S), 1/4, 1/2|S|)$. Check $\hat{L}_S(\hat{f})$, if it is 0 output 1 and if not output 0. Now note that if there is not model in $\mathcal{F}$ that achieves 0 error on S then clearly we will output a 0 (since we only ever output 1 if we actually find a consistent model). On the other hand, since $\epsilon = 1/2|S|$, if $\text{CONS}_{\mathcal{F}}(S) = 1$, then with probability $1 - \delta = 3/4$ we will have that $\hat{L}_S(\hat{f}) < 1/2|S|$. But since its a binary classification problem, this would mean that in fact $\hat{L}_S(\hat{f}) = 0$ and so with probability 3/4 we do actually output a 1. Clearly running time is polynomial in $m|S|$

□

2 Properly Learning 3-TERM-DNF is Hard

We will prove hardness of properly learning 3-TERM-DNF model classes. Say $\mathcal{X} = \{0, 1\}^m$ is the m bit representation of our inputs. Say $\mathcal{F} = \{T_1 \wedge T_2 \wedge T_3 : T_i \in \text{Conj}_m\}$. That is, each $f \in \mathcal{F}$ is of the form

$$f(x) = (x[1] \wedge \neg x[4] \wedge x[i] \wedge \neg x[m]) \vee (x[2] \wedge \neg x[3] \wedge x[6]) \vee (x[20] \wedge \neg x[7])$$

First note that $|\mathcal{F}| = 3^{O(m)}$ and so $\text{VC}(\mathcal{F}) = O(m)$. Hence sample complexity if $O(m \log(1/\delta)/\epsilon)$. However, we will show that for 3-TERM-DNF, the $\text{CONS}_{\mathcal{F}}$ problem is NP hard! We will do this by showing that the notoriously hard problem of three coloring of a graph can be solved efficiently if $\text{CONS}_{\mathcal{F}}$ can be solved efficiently. That is we will provide an efficient reduction from graph 3-colorability.

To recall, the graph 3-colorability problem is one where given an undirected graph you are asked to decide it you can color its vertices with 3 colors in such a way that no 2 neighbors have the same color. 2-coloring is easy but it turns out that 3-coloring is hard, it is one of the NP complete problems which we reasonably expect to be very hard. If we prove a problem is NP hard, then unless P = NP (which we really dont believe is the case), the problem will not be solvable in poly time.

Claim 2. *Let $\mathcal{F}$ represent the 3-TERM-DNF model class, the problem* $\text{CONS}_{\mathcal{F}}$ *is NP hard.*

Proof Sketch. Given a graph $G = (V, E)$, consider the following mapping to sample S_G such that $|S_G| = |V| + |E|$ given as follows:

$$S_G = \{(\mathbf{1} - e_i, 1) : i \in [V]\} \cup \{(\mathbf{1} - e_i - e_j, 0) : (i, j) \in E\}$$

That is x's are $|V|$ dimensional binary vectors describes as above and corresponding labels are 0 or 1 as described above. Now we claim that the reduction is simple: If there exists a 3 coloring say c for the graph then, if we define $T_k = \wedge_{c(i)\neq k} x[i]$ for each $k \in [3]$, then S_G is satisfied by the model $T_1 \vee T_2 \vee T_3$. On the other hand, if $T_1 \vee T_2 \vee T_3$ satisfies S_G, then if we set for each vertex i the color $c(i) = \min\{k \in [3] : T_k \text{ satisfies } (\mathbf{1} - e_i, 1)\}$ □

From the above results we can conclude that If NP $\neq$ RP, then 3-Term-DNF model class is not efficiently properly PAC learnable.