CIS 625, Theory of Machine Learning
Problem Set 1

Prof. Michael Kearns

Assigned September 21, 2020; Due October 6

You are free to work in groups of up to *four* people, but every student should think about every problem, and every student should participate in their group's writeup; please indicate which student wrote up each solution. You are also free to work alone. The quality of a group's work should be commensurate with its size.

Try to provide as much detail and rigor as possible in your solutions, similar to the level in lecture. If you can't solve a problem, get as far as you can, and write down what your ideas were and where you got stuck. If you feel you need to make additional assumptions to solve a problem, carefully state what they are.

1. In lecture we saw that some classes are PAC learnable from positive examples only (such as rectangles in the plane and conjunctions of Boolean variables) — that is, by an algorithm that simply ignores any negative examples in the sample drawn. Suppose that $\mathcal{C}_1$ is PAC learnable by $\mathcal{H}_1$ from positive examples only, and that $\mathcal{C}_2$ is PAC learnable by $\mathcal{H}_2$ from positive examples only. Show that $\mathcal{C}_1 \wedge \mathcal{C}_2$ is PAC learnable by $\mathcal{H}_1 \wedge \mathcal{H}_2$ from positive examples only. Here $\wedge$ is the logical and operator, and $\mathcal{C}_1 \wedge \mathcal{C}_2$ is the class of all functions of the form $c_1(x) \wedge c_2(x)$ for $c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2$.

2. Suppose $\mathcal{C}_1$ is PAC learnable by $\mathcal{H}_1$ from positive examples only, and that $\mathcal{C}_2$ is PAC learnable by $\mathcal{H}_2$. Show that $\mathcal{C}_1 \wedge \mathcal{C}_2$ is PAC learnable by $\mathcal{H}_1 \wedge \mathcal{H}_2$.

3. Suppose that $\mathcal{C}_1$ is PAC learnable by $\mathcal{H}_1$ from positive examples only, and that $\mathcal{C}_2$ is PAC learnable by $\mathcal{H}_2$ from positive examples only. Then is it true in general that $\mathcal{C}_1 \vee \mathcal{C}_2$ is PAC learnable by $\mathcal{H}_1 \vee \mathcal{H}_2$? Why or why not?

4. A 3-term DNF is *monotone* if no variable appears negated. For example, $x_1x_5 \vee x_2x_3x_7 \vee x_1x_4x_6$ is monotone, but $x_1\neg x_5 \vee x_2x_3x_7 \vee \neg x_1x_4\neg x_6$ is not due to the negations. Show that if monotone 3-term DNF is PAC learnable by monotone 3-term DNF, then 3-term DNF is PAC learnable by 3-term DNF. Can you make a more general statement for any class of Boolean formulae $\mathcal{C}$ (perhaps meeting some natural conditions) and its monotone counterpart?

5. A 3-term DNF is *read-once* if no variable appears more than once in the formula. For example, $x_1\neg x_5 \vee x_2\neg x_3 x_7 \vee x_4 x_6$ is read-once, but $x_1\neg x_5 \vee x_2\neg x_3 x_7 \vee \neg x_2 x_4 x_6$ is not because x_2 appears twice. Show that if read-once 3-term DNF is PAC learnable by read-once 3-term DNF, then 3-term DNF is PAC learnable by 3-term DNF. Can you make a more general statement for any class of Boolean formulae $\mathcal{C}$ (perhaps meeting some natural conditions) and its read-once counterpart?

6. Show that the dependence on δ in the sample complexity of PAC learning never needs to be worse than order $\log(1/\delta)$ — that is, if L is an algorithm for PAC learning algorithm $\mathcal{C}$ by $\mathcal{H}$ whose required sample size has a polynomial dependence on $1/\delta$, there is another PAC learning algorithm L' whose required sample size has only order $\log(1/\delta)$ dependence on δ.

7. In the *consistency problem* for a class $\mathcal{C}$, we are given an arbitrary labeled sample $S = \{< x_1, y_1 >, \ldots, < x_m, y_m >\}$, and must find a $c \in \mathcal{C}$ such that $c(x_i) = y_i$ for all i if one exists, or output "none". Show that if $\mathcal{C}$ is PAc learnable by $\mathcal{C}$, there is a randomized polynomial time algorithm for the consistency problem that succeeds with probability at least 99/100 on every possible input sample S.

8. In lecture we showed that if, given a sample S of some $c \in \mathcal{C}$, we could find an hypothesis h (not necessarily in $\mathcal{C}$) whose length was *sublinear* in $m = |S|$, we could PAC learn C by such hypotheses, provided m is sufficiently large. In this problem you are asked to prove the converse. Suppose we have a PAC learning algorithm L for $\mathcal{C}$ that outputs hypotheses of some unknown form. Then use L to construct an algorithm that, given a sample S of some $c \in \mathcal{S}$, outputs an efficiently evaluatable hypothesis h (whose form you may determine) that is consistent with S, and has size that is sublinear in m. Hint: design a simulation of L in which the accuracy parameter ϵ depends on the degree of the polynomial running time of L.