CS 391L: Machine Learning	**Spring 2024**

Homework 1 - Theory

Lecture: Prof. Adam Klivans

Keywords: Boolean functions, mistake bounds, PAC learning

Instructions: Please either typeset your answers (LaTeX recommended) or write them very clearly and legibly and scan them, and upload the PDF on edX. Legibility and clarity are critical for fair grading.

1. Often in binary classification we are interested in the differences in the output of our current classifier, g, and an unknown function f that we are trying to learn. It is common in these cases to examine the quantity produced by $f(x)g(x)$ for a given input x. For this problem, let D be an arbitrary distribution on the domain $\{-1,1\}^n$, and let $f, g : \{-1,1\}^n \to \{-1,1\}$ be two Boolean functions.

   (a) [**6 points**] Prove that

   $$\mathbb{P}_{x\sim D}[f(x) \neq g(x)] = \frac{1 - \mathbb{E}_{x\sim D}[f(x)g(x)]}{2}.$$

   (b) [**4 points**] Would this still be true if the domain were some other domain (such as $\mathbb{R}^n$, where $\mathbb{R}$ denotes the real numbers, with say the Gaussian distribution) instead of $\{-1,1\}^n$? If yes, justify your answer. If not, give a counterexample.

   Note: Only the *domain* changes here. The output is still boolean.

2. [**10 points**] Let f be a decision tree with t leaves over the variables $x = (x_1, \ldots, x_n) \in \{-1,1\}^n$. Explain how to write f as a multivariate polynomial $p(x_1, \ldots, x_n)$ such that for every input $x \in \{-1,1\}^n$, $f(x) = p(x)$. (You may interpret -1 as FALSE and 1 as TRUE or the other way round, at your preference.) *(Hint: try to come up with an "indicator polynomial" for every leaf, i.e. one that evaluates to the leaf's value if x is such that that path is taken, and 0 otherwise.)*

3. [**10 points**] Compute a depth-two decision tree for the training data in table 1 using the Gini function, $C(a) = 2a(1-a)$ as described in class. What is the overall accuracy on the training data of the tree? For clarity, this will be a full binary tree and a full binary tree of depth-two has four leaves.

X	Y	Z	Number of positive examples	Number of negative examples
0	0	0	10	20
0	0	1	25	5
0	1	0	35	15
0	1	1	35	5
1	0	0	5	15
1	0	1	30	10
1	1	0	10	10
1	1	1	15	5

Table 1: decision tree training data

4. [**10 points**] Suppose the domain X is the real line, $\mathbb{R}$, and the labels lie in $Y = \{-1, 1\}$, Let $\mathcal{C}$ be the concept class consisting of simple threshold functions of the form h_θ for some $\theta \in \mathbb{R}$, where $h_\theta(x) = -1$ for all $x \leq \theta$ and $h_\theta(x) = 1$ otherwise. Give a simple and efficient PAC learning algorithm for $\mathcal{C}$ that uses only $m = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ training examples to output a classifier with error at most ϵ with probability at least $1 - \delta$.

5. [**6 points**] In this problem we will show that the existence of an efficient mistake-bounded learner for a class $\mathcal{C}$ implies an efficient PAC learner for $\mathcal{C}$.

   Concretely, let $\mathcal{C}$ be a function class with domain $X \in \{-1, 1\}^n$ and binary labels $Y \in \{-1, 1\}$. Assume that $\mathcal{C}$ can be learned by algorithm/learner A with some mistake bound t. You may assume you know the value t. You may also assume that at each iteration, A runs in time polynomial in n and that A only updates its state when it gets an example wrong. The **concrete goal of this problem** is to create a PAC-learning algorithm, B, that can PAC-learn concept class $\mathcal{C}$ with respect to an arbitrary distribution D over $\{-1, 1\}^n$ using algorithm A as a sub-routine.

   In order to prove that learner B can PAC-learn concept class $\mathcal{C}$, we must show that there exists a finite number of examples, m, that we can draw from D such that B produces a hypothesis whose true error is more than ϵ with probability at most δ. First, fix some distribution D on X, and we will assume that the examples are labeled by an unknown $c \in \mathcal{C}$. Additionally, for a hypothesis (i.e. function) $h : X \to Y$, let $\text{err}(h) = \mathbb{P}_{x \sim D}[h(x) \neq c(x)]$. Formally, we will need to bound m such that the following condition holds:

   $$\forall \delta, \epsilon \in [0, 1], \ \exists m \in \mathbb{N} \mid \ \ \mathbb{P}_{x \sim D}[\text{err}(B(\{x\}^m)) > \epsilon] \leq \delta \qquad x \sim D \tag{1}$$

   where $B(\{x\}^m)$ denotes a hypotheses produced from B with m random draws of x from an arbitrary distribution D.

   To find this m, we will first decompose it into blocks of examples of size k and make use of results based on a single block to find the bound necessary for m that satisfies condition 1.

   Note: Using the identity $\mathbb{P}[\text{err}(h) > \epsilon] + \mathbb{P}[\text{err}(h) \leq \epsilon] = 1$, we can see that $\mathbb{P}[\text{err}(h) > \epsilon] \leq \delta \Leftrightarrow \mathbb{P}[\text{err}(h) \leq \epsilon] \geq 1 - \delta$, which makes the connection to the definition of PAC-learning discussed in lecture explicit.

   (a) Fix a single arbitrary hypothesis $h' : X \to Y$ produced by A and determine a lower bound on the number of examples, k, such that $\mathbb{P}[\text{err}(h') > \epsilon] \leq \delta'$. (The contrapositive view would be: with probability at least $1 - \delta'$, it must be the case that $\text{err}(h') \leq \epsilon$. Make sure this makes sense.)

   (b) From part 5a we know that as long as a block is at least of size k, then if that block is classified correctly by *a fixed arbitrary hypothesis* h' we can effectively upper bound the probility of the 'bad event' (i.e. A outputs h' s.t. $\text{err}(h') > \epsilon$) by δ'. However, our bound must apply to every h that our algorith B could output for an arbitrary distribution D over examples. With this in mind, how large should m be so that we can bound all hypotheses that could be output? (You may assue that algorithm B will know the mistake bound throughout the question.)

   (c) Put everything together and fully describe (with proof) a PAC learner that is able to output a hypothesis with a true error at most ϵ with probability at least $1 - \delta$, given a mistake bounded learner A. To do this you should first describe your pseudocode for algorithm B which will use A as a sub-routine (no need for minute details or code, broad strokes will suffice). Then, prove there exists a finite number of m examples for B to PAC-learn $\mathcal{C}$ for all values of δ and ϵ by lower bounding m by a function of ϵ, δ, and t (i.e. finding a finite lower bound for m such that the PAC-learning requirements in 1 are satisfied).