

Computer Science 4252: Introduction to Computational Learning Theory
Problem Set #5 Fall 2011

Due 1:10pm Wednesday, December 7, 2011

Do **five** of the following six problems (there will not be extra credit for doing a sixth problem).

Problem 1 In this problem you will consider a new type of noise called *random coordinate noise*. In this learning scenario, the domain is $X = \{0, 1\}^n$ and there is an unknown distribution \mathcal{D} over X . There is an unknown *coordinate noise rate* $0 \leq \eta < 1/2$ and an oracle $\overline{EX}_\eta(c, \mathcal{D})$, where c is the target concept, which works as follows. When it is invoked, the oracle $\overline{EX}_\eta(c, \mathcal{D})$ draws $x \sim \mathcal{D}$ (a string in $\{0, 1\}^n$) and then independently flips each bit x_i with probability η to obtain a new string $x' \in \{0, 1\}^n$; then the oracle returns $\langle x', c(x) \rangle$ to the learner. (So in other words, each coordinate of the *example* is independently flipped with probability η , but the label is not flipped.) The learning algorithm is given an upper bound $\eta^* < 1/2$ on the coordinate noise rate η (i.e. it is guaranteed that $\eta \leq \eta^*$) but is not told η .

Suppose that you are learning an unknown monotone conjunction in the random coordinate noise model. (You may assume that the target monotone conjunction contains at least one variable.) Give an efficient algorithm that either

- estimates the unknown coordinate noise rate η to within an additive $\pm\epsilon$ with high probability (at least 99%); or
- outputs a hypothesis h for the unknown monotone conjunction c that with high probability (at least 99%) is ϵ -accurate with respect to \mathcal{D} .

Note that for this problem you *do not* have to give a full-fledged learning algorithm in this model – you just need to estimate η , or barring that, to output a high-accuracy hypothesis.

(It is known that monotone conjunctions can be efficiently learned in this model, and estimating η is an ingredient in the learning algorithm.)

Problem 2 Let X be a finite instance space. Given a concept class \mathcal{C} over X and a target concept $c \in \mathcal{C}$, we say that a sequence T of labelled examples is a *consistency sequence* for c in \mathcal{C} if c is the only concept in \mathcal{C} which is consistent with T . Let $T(c)$ be the set of all consistency sequences for c in \mathcal{C} . The *consistency number* of concept class \mathcal{C} is then defined to be

$$\text{CN}(\mathcal{C}) = \max_{c \in \mathcal{C}} \min_{T \in T(c)} |T|$$

where $|T|$ denotes the number of examples in the sequence T .

- (i) Give an example of a concept class \mathcal{C} for which $\text{CN}(\mathcal{C}) > \text{VC-DIM}(\mathcal{C})$.
- (ii) Give an example of a concept class \mathcal{C} for which $\text{CN}(\mathcal{C}) < \text{VC-DIM}(\mathcal{C})$.
- (iii) Show that for any concept class \mathcal{C} , $\text{CN}(\mathcal{C}) \leq |\mathcal{C}| - 1$.
- (iv) Show that for any concept class \mathcal{C} , $\text{CN}(\mathcal{C}) \leq |\mathcal{C}| + \text{VC-DIM}(\mathcal{C}) - 2^{\text{VC-DIM}(\mathcal{C})}$.

Problem 3 In this problem you'll explore how boosting algorithms can be used to learn linear threshold functions. For concreteness, we will just focus in on using Adaboost to find a linear threshold function that is consistent with a given set of examples that are labelled according to some linear threshold function.

(i) (easy) Suppose that h and f are both functions which take values in $\{-1, 1\}$. Show that for any distribution \mathcal{D} , h is a weak hypothesis for f with advantage γ if and only if $\mathbf{E}_{x \in \mathcal{D}}[h(x)f(x)] \geq 2\gamma$.

(ii) Suppose that $f(x_1, \dots, x_n) : \{-1, 1\}^n \rightarrow \{-1, 1\}$ is a linear threshold function $f(x) = \text{sign}(w \cdot x)$ where

[1] each x_i takes values in $\{-1, 1\}$;

[2] $w = (w_1, \dots, w_n)$ where each w_i is an integer value and $W = \sum_{i=1}^n |w_i|$;

[3] for all $x \in \{-1, 1\}^n$, we have $w_1x_1 + \dots + w_nx_n \neq 0$.

Show that for any distribution \mathcal{D} over $\{-1, 1\}^n$, there must be some x_i such that $|\mathbf{E}_{x \in \mathcal{D}}[f(x) \cdot x_i]| \geq \frac{1}{W}$. Hint: Use (and justify) the fact that $1 \leq \mathbf{E}_{x \in \mathcal{D}}[|w \cdot x|]$.

(iii) Use parts (i) and (ii) to explain how Adaboost can be used to find a linear threshold function that is consistent with a set S of m examples from $\{-1, 1\}^n$ that are labelled according to a linear threshold function f as described in part (ii). What is the weak learning algorithm that Adaboost uses? What are the weak hypotheses? What is (a lower bound on) the advantage of each weak hypothesis? How many stages must AdaBoost be run for in order to find a linear threshold function hypothesis which is consistent with S ? What is the overall runtime (as a function of n , m and W) of using Adaboost to find a consistent linear threshold function?

Problem 4 Our definition of efficient PAC learning in the presence of random classification noise at rate $\eta < 1/2$ requires that the algorithm run in time $\text{poly}(\frac{1}{1-2\eta})$ (ignore all the other parameters for simplicity for this problem). This is intuitively plausible, since (i) if $\eta = 0$ (no noise) the function $\frac{1}{1-2\eta}$ equals 1, and (ii) as η approaches $1/2$ (and learning becomes impossible) the function $\frac{1}{1-2\eta}$ approaches infinity. But why is $\frac{1}{1-2\eta}$ the "right" function, as opposed to some other function such as $1 + \log(\frac{1}{1-2\eta})$ or $\exp(\frac{1}{1-2\eta} - 1)$, that satisfies (i) and (ii)?

Argue as clearly and convincingly as you can that any PAC learning algorithm for learning in the presence of random classification noise at rate η must have runtime which grows as $\Omega(\frac{1}{1-2\eta})$.

Problem 5 Do Exercise 5.4 from the Kearns & Vazirani textbook.

Problem 6 A *branching program* is a directed acyclic graph which has two *leaf nodes* which are labeled with 0 and 1, and a collection of *internal nodes* each of which is labeled with a variable from x_1, \dots, x_n . Each internal node has two outgoing edges labeled 0 and 1; there is one internal node (with no incoming edges) which is an initial node. A branching program computes a Boolean function in a fashion similar to a decision tree: given an input $x \in \{0, 1\}^n$, we start at the initial node and follow the path indicated by the value of the variables at each node till we arrive at a leaf, whose label then determines the value of the function.

Show that there is probably no polynomial-time PAC learning algorithm for the class of branching programs. (You may use any hardness result stated in class or in the Kearns/Vazirani textbook.)