Instructor: Rocco A. Servedio

Computer Science 4252: Introduction to Computational Learning Theory
Problem Set #1 Fall 2023

Due 11:59pm Wednesday, September 27, 2023

See the course Web page for instructions on how to submit homework. **Important:** To make life easier for the TAs, **please start each problem on a new page.**

Problem 1 In this problem you'll play around with some of the concept classes we discussed in class (and some of their close friends).

(i) True or false: Every $(\log n)$-DNF over $\{0,1\}^n$ is equivalent to a poly(n)-term DNF. (Justify your answer.)

(ii) True or false: every k-term DNF is equivalent to some k-CNF. (Prove your answer; i.e. if you answer "yes" you must give a proof, and if you answer "no" you should give a counterexample.)

(iii) True or false: every k-CNF is equivalent to some k-term DNF. (Prove your answer.)

Problem 2 Let us say that an online mistake-bound algorithm *takes it easy* if it changes its hypothesis only when a mistake is made.

All of the online mistake-bound algorithms we've seen in class (the elimination algorithm, the decision list algorithm, Winnow, etc.) are ones which take it easy. In this problem you'll show that this is not a coincidence:

Let $\mathcal{C}$ be any concept class, and let A be any online learning algorithm (not necessarily one which takes it easy) which has a finite mistake bound M for $\mathcal{C}$. Prove that there must exist a learning algorithm A' for $\mathcal{C}$ which takes it easy and which also has mistake bound M.

Problem 3

(i) Let the domain X be $\{0,1,2,\ldots,2^n-1\}$ and let the concept class $\mathcal{C}$ be the class of all *intervals*, where an *interval* is a set of consecutive elements of X, i.e. an interval $c \in \mathcal{C}$ is

$$c = \{a, a+1, \ldots, b-1, b\} \quad \text{for some } a, b \in X \text{ with } a \leq b.$$

(Note that this concept class is different from the concept class of "initial intervals" we discussed in class because the left endpoint of the interval need not be 0.)

Give a computationally efficient algorithm with a good mistake bound in the OLMB model for learning $\mathcal{C}$. Justify your claimed mistake bound (just a paragraph or two should be sufficient; and it's okay to use big-Oh notation in describing the mistake bound of your algorithm.)

(ii) Consider the same domain X as in part (i), but now let $\mathcal{C}$ be the class of "wrap-around intervals", (alternately intervals mod 2^n) — now an interval is allowed to "wrap around", i.e. we consider the number $2^n - 1$ as being consecutive with 0. In more detail, a wrap-around interval is either of the form

$$c = \{a, a+1, \ldots, b-1, b\} \quad \text{for some } a, b \in X \text{ with } a \leq b$$

or of the form

$$c = \{0, 1, \ldots, a\} \cup \{b, b+1, \ldots, 2^n - 1\} \quad \text{for some } a, b \in X \text{ with } a \leq b.$$

Give a computationally efficient algorithm with a good mistake bound in the OLMB model for learning $\mathcal{C}$. Justify your claimed mistake bound (just a paragraph or two should be sufficient; and it's okay to use big-Oh notation in describing the mistake bound of your algorithm.)

(It's possible that your solution for this problem will also solve part (a); if so that's fine and you don't need to do part (a) separately.)

Problem 4

Consider the decision list "if $x_3 = 1$ then output 1, else if $x_2 = 1$ then output 0, else if $x_1 = 1$ then output 1, else output 0." Since this is a length-r-decision list where r is a constant (it's 3), the analysis of the decision list learning algorithm which we gave in class gives us that the algorithm has a mistake bound of at most $O(n)$ mistakes for the instance space $X = \{0,1\}^n$.

(i) Show that the algorithm from class can sometimes make $\Omega(n)$ mistakes when it is used to learn the above length-3 decision list.

(ii) Describe a slight modification of the algorithm from class and argue that your modified algorithm has a mistake bound which improves on the mistake bound of the algorithm from class by an additive $\Omega(n)$ (i.e., you should argue that your algorithm makes, say, at least $n/2$, or $n/100$, many fewer mistakes than the mistake bound of the algorithm from class, when used to learn any length-r decision list).

Problem 5 In this problem we'll consider the domain $X = \{1, 2, \ldots, N\}^d$. A *$d$-dimensional hyper-rectangle* over this domain X is a subset $c \subseteq X$ defined by $2d$ values $1 \leq a_i \leq b_i \leq N$ for $i = 1, \ldots, d$; the subset is

$$c = \{(x_1, \ldots, x_d) \in X : a_i \leq x_i \leq b_i \text{ for all } i = 1, \ldots, d\}.$$

Let RECT denote the class of all d-dimensional hyper-rectangles over X.

(i) Explain how Winnow1 can be used to learn the concept class RECT. What is the mistake bound of your algorithm? What is the running time of the algorithm per trial? (Hint: Consider variables corresponding to inequalities over a single variable x_i, and aim for a running time of $O(dN)$ per trial.)

(ii) Now make your solution exponentially more efficient, by explaining how Winnow1 can be used to learn RECT with the same mistake bound you achieved in (i) but with a running time of only poly$(d, \log N)$ per trial.