UNIVERSITY OF OXFORD

Computational Learning Theory
Hilary Term 2018
Week 4

Problem Sheet 2

Instructions: The problem sheets are designed to increase your understanding of the material taught in the lectures, as well as to prepare you for the final exam. You should attempt to solve the problems on your own after reading the lecture notes and other posted material, where applicable. Once you have given sufficient thought to a problem, if you are stuck, you are encouraged to discuss with others in the course and with the lecturer during office hours. Please avoid posting on Piazza until the Wednesday before the submission deadline. You are *not permitted* to search for solutions online.

1 Learning Parity Functions

Let $X_n = \{0,1\}^n$ be the instance space. A parity function over X_n is defined by some subset $S \subseteq \{1,\ldots,n\}$, and takes the value 1 if and odd number of the input literals in the set $\{x_i \mid i \in S\}$ are 1 and 0 otherwise. For example, if $S = \{1,3,4\}$, then the function $f = x_1 \oplus x_3 \oplus x_4$ computes the parity on the subset $\{x_1, x_3, x_4\}$. Note that any such parity function can be represented by a bit string of length n, by indicating which indices are part of S. Let PARITIES denote the class of all parity functions. Show that the class PARITIES is efficiently PAC-learnable by describing an algorithm, analysing its running time and proving its correctness.

Hint: The parity operation can be viewed as addition modulo 2.

2 Output Hypothesis as a Turing Machine

Recall that in the definition of PAC-learning, we require that the hypothesis output by the learning algorithm be evaluatable in polynomial time. Suppose we relax this restriction, and let H be the class of all Turing machines (not necessarily polynomial time)—so the output of the learning algorithm can be any program. Let C_n be the class of all boolean circuits of size at most $p(n)$ for some fixed polynomial p and having n boolean inputs. Show that $C = \bigcup_{n \geq 1} C_n$ is PAC-learnable using H (under this modified definition). Argue that this solution shows that the relaxed definition trivialises the model of learning.

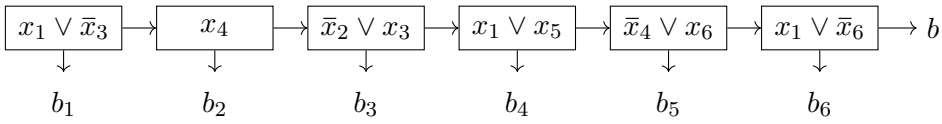
3 Learning Decision Lists

A k-decision list over n boolean variables $x_1, \ldots, x_n$, is defined by an ordered list

$$L = (c_1, b_1), (c_2, b_2), \ldots, (c_l, b_l),$$

and a bit b, where each c_i is a clause (disjunction) of at most k literals (positive or negative) and each $b_i \in \{0,1\}$. For $a \in \{0,1\}^n$ the value $L(a)$ is defined to be b_j, where j is the smallest index satisfying $c_j(a) = 1$ and $L(a) = b$ if no such index exists. Pictorially, a decision list

can be depicted as shown below. As we move from left to right, the first time a clause is satisfied, the corresponding b_j is output, if none of the clauses is satisfied the default bit b is output.



Give a consistent learner for the class of decision lists. As a first step, argue that it is enough to just consider the case where all the clauses have length 1, *i.e.,* in fact they are just literals.