Computational Learning Theory
Michaelmas Term 2022

Problem Sheet 1

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. Problems marked with an asterisk are optional. 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. You are *not permitted* to search for solutions online.

1 Learning Hyper-rectangles

The concept class of hyper-rectangles over $\mathbb{R}^n$ is defined as follows:

$$C_n = \{[a_1, b_1] \times \cdots \times [a_n, b_n] \mid a_i, b_i \in \mathbb{R}, a_i < b_i\}$$

Generalise the algorithm discussed in class (for rectangles in $\mathbb{R}^2$) and show that it *efficiently* PAC learns the class of hyper-rectangles. Give bounds on the number of samples required to guarantee that the error is at most ϵ with probability at least $1 - \delta$.

Note: You may assume that the distribution D over $\mathbb{R}^n$ can be expressed using a continuous density function that is defined over all of $\mathbb{R}^n$. (*Optional*: As an extra challenge, argue why the algorithm still works even when such an assumption regarding the distribution does not hold.)

Hint: For the first part, follow exactly the same algorithm as in the lectures. For the optional part, we need to change the definition of C_n slightly to allow $a_i \leq b_i$ (rather than $a_i < b_i$). Suppose R is the target hyper-rectangle. Let $p = \mathbb{P}_{x \sim D}[x \in R]$. Suppose $p \leq \epsilon$, then the algorithm is always correct. Otherwise, define,

$$v_1 = \min \left\{ v \geq a_1 \mid \mathbb{P}_{\mathbf{x} \sim D}\left[[a_1, v] \times [a_2, b_2] \times \cdots \times [a_n, b_n]\right] \geq \frac{\epsilon}{2n} \right\},$$

by assumption that $p \geq \epsilon$ the set over which min is taken is non-empty. Argue that the min above is well defined, i.e. using inf is not required, and define T_1 to be the hyper-rectangle $[a_1, v_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$. Similarly, define $T_2, T_3, \ldots, T_{2n}$, and follow the same analysis.

2 PAC Learning : Confidence Parameter

Say that an algorithm L *perhaps learns* a concept class C using hypothesis class H, if for every n, for every concept $c \in C_n$, for every distribution D over X_n and for every $0 < \epsilon < 1/2$, L given access to $\mathsf{EX}(c, D)$ and inputs ϵ and size(c), runs in time polynomial in n, size(c) and $1/\epsilon$, and outputs a polynomially evaluatable hypothesis $h \in H_n$, that with probability at least $3/4$ satisfies $\text{err}(h) \leq \epsilon$. In other words, we've set $\delta = 1/4$ in the definition of *efficient* PAC learning.

Show that if C is "perhaps learnable" using H, then C is also *efficiently* PAC learnable using H.

Hint: You will have to use the Chernoff-Hoeffding bound. Your proof should not rely on $\delta < 1/2$, in particular it should work for $\delta = 3/4$, even though we have required $0 < \delta < 1/2$ in the definition of PAC learning.

3 Hardness of Learning Boolean Threshold Functions

We will consider the question of learning boolean threshold functions. Let $X_n = \{0,1\}^n$ and for $w \in \{0,1\}^n$ and $k \in \mathbb{N}$, $f_{w,k} : X_n \to \{0,1\}$ is a boolean threshold function defined as follows:

$$f_{w,k}(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i \cdot x_i \geq k \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathsf{TH}_n = \{f_{w,k} \mid w \in \{0,1\}^n, 0 \leq k \leq n\}$ and $\mathsf{TH} = \bigcup_{n \geq 1} \mathsf{TH}_n$. Show that unless $\mathsf{RP} = \mathsf{NP}$, there is no efficient PAC-learning algorithm for learning TH, if the output hypothesis is also required to be in TH, *i.e.,* a *proper* PAC-learning algorithm.

Hint: You should reduce from Zero-One Integer Programming (ZIP) which is known to be NP-complete. An instance of ZIP consists of an $s \times n$ matrix A with entries in $\{0,1\}$, a vector $\mathbf{b} \in \{0,1\}^s$ and a pair $(\mathbf{c}, B)$ (the objective) with $\mathbf{c} \in \{0,1\}^n$ and $B \in \mathbb{Z}$. The decision problem is to determine wither there exists an assignment for the n variables $z_1, \ldots, z_n$, each variable taking a value in the set $\{0,1\}$, such that for each $1 \leq i \leq s$, $\sum_{j=1}^{n} A_{ij} z_j \leq b_i$ and $\sum_{j=1}^{n} c_j z_j \geq B$.

4 Learning Parity Functions

Let $X_n = \{0,1\}^n$ be the instance space. A parity function, χ_S, 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 $\{z_i \mid i \in S\}$ are 1 and 0 otherwise. For example, if $S = \{1,3,4\}$, then the function $\chi_S(z_1, \ldots, z_n) = z_1 \oplus z_3 \oplus z_4$ computes the parity on the subset $\{z_1, z_3, z_4\}$. 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 $\mathsf{PARITIES}_n$ denote the concept class consisting of all 2^n parity functions; observe that the the concept class $\mathsf{PARITIES}_n$ has representation size at most n. Show that the class $\mathsf{PARITIES}$, defined as $\mathsf{PARITIES} = \bigcup_{n \geq 1} \mathsf{PARITIES}_n$, is *efficiently proper* PAC learnable. You should clearly describe a learning algorithm, analyse its running time and prove its correctness.

Hint: The parity operation can be viewed as addition modulo 2. Use the fact that the set $\{0,1\}$ under addition and multiplication modulo 2 is a field.

5 Output Hypothesis as a Turing Machine (*)

Recall that in the definition of *efficient* 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 "efficiently" PAC-learnable using H (under this modified definition). Argue that this solution shows that the relaxed definition trivialises the model of learning.

Hint: Outsource the computationally demanding work of finding a consistent classifier to h, the hypothesis that makes predictions. Your learning algorithm only needs to indentify what input needs to be given to the Turing Machine h.