Leslie G. Valiant
TFs: Aayush Karan & Kevin Cong

CS 228: Computational Learning Theory
Homework 4 (Take Home Exam) Problems
Mar. 28th - Apr. 4th

Policy reminders: You are strongly encouraged to type your solutions using LATEX. You may *not* discuss this problem set with any one except the course staff. You are not allowed to use any materials except your own class notes, the course textbooks, and any material posted on the class website or handed out in class. Moreover, you are not allowed to query any LLM regarding any part of any problem. This problem set is due on **April 4**; *no late days are allowed.* For any clarifications or questions, please send an email to the teaching staff.

Problem 1 (10 pts)**. Few Relevant Variables Implies Efficient PAC Learnability.**

A variable x_i is said to be *relevant* to a boolean function f if there are two truth assignments A, B which differ only in their assignment to x_i but which have $f(A) \neq f(B)$. Let $\mathcal{C}$ be the class of boolean functions over $\{0,1\}^n$ that have at most $\log n$ relevant variables. Give a direct proof (i.e., without reducing to other classes) that $\mathcal{C}$ is PAC-learnable in polynomial time by membership queries.

Problem Setup: Let $f : 0, 1^n \to 0, 1$ be a Boolean function depending on at most $\log n$ variables. We give a direct polynomial-time learning algorithm using membership queries:

Identify relevant variables: For each $i \in 1, \ldots, n$, we test whether x_i is relevant to f. Fix some reference assignment $a \in 0, 1^n$ (for example, $a = (0, 0, \ldots, 0)$) and obtain $f(a)$ via a membership query. Then for each i, flip the i-th bit of a to get $a^{(i)}$ (so $a^{(i)}_j = a_j$ for $j \neq i$ and $a^{(i)}_i = 1 - a_i$) and query $f(a^{(i)})$. If $f(a^{(i)}) \neq f(a)$, then x_i is relevant; if $f(a^{(i)}) = f(a)$, then x_i is probably irrelevant. Since f has at most $\log n$ relevant variables, most coordinates will appear irrelevant under any single assignment. If a truly relevant variable x_i happens not to change f when flipped (i.e. $f(a^{(i)}) = f(a)$ despite x_i being relevant), it means that the particular assignment to the other relevant variables in a caused f to be insensitive to x_i. In that case, we can repeat the test under a different assignment. Because there are at most $\log n$ relevant variables, a random choice of a will, with high probability, assign the other relevant bits in a way that reveals x_i's influence. In fact, by trying at most $O(\log n)$ different random assignments as bases, we can ensure that every relevant coordinate flips the output in at least one trial. Thus, we can find the exact set $R \subseteq 1, \ldots, n$ of relevant indices using at most $O(n \log n)$ queries.

Learn f on relevant coordinates: Let $r = |R| \leq \log n$. Now f is effectively a function of the r bits in R. We can determine f exactly by querying all 2^r possible assignments to those r bits. For each $y \in 0, 1^R$ (an assignment to the bits in R), construct any $x \in 0, 1^n$ such that $x_R = y$ (bits in R set according to y) and $x_j = 0$ for $j \notin R$ (arbitrary values work for irrelevant bits). Query $f(x)$ to obtain the value $f(y)$ (since bits outside R do not matter). Collecting these results for all $2^r \leq 2^{\log n} = n$ assignments, we now have the truth-table of f as a function of its relevant inputs.

Conclusion: Finally, output any hypothesis h that depends only on R and matches this truth-table (for example, a DNF or decision tree consistent with those 2^r values). By construction, $h(x) = f(x)$

for every $x \in 0,1^n$. The algorithm uses at most $n \log n + n = O(n \log n)$ membership queries and runs in poly(n) time. Therefore, $\mathcal{C}$ is PAC-learnable (in fact, exactly learnable) in polynomial time with membership queries.

Problem 2 (15 pts)**. Boolean Threshold Functions Are Not PAC-learnable.** Given any $y \in \{0,1\}^n$ and any integer $k \geq 0$, the boolean threshold function $TH_{k,y}$ is defined as follows: an input $x \in \{0,1\}^n$ is a positive example iff $x \cdot y \geq k$, where $x \cdot y$ denotes the standard real-valued dot product of two n-dimensional vectors. Prove that if $\mathsf{NP} \neq \mathsf{RP}$ then the representation class of boolean threshold functions is not PAC learnable by boolean threshold functions.

Hint. Reduce from the Zero-One Integer Programming problem (ZIP) which is known to be NP-complete. An instance of ZIP is a set of s pairs $\langle c_i, b_i \rangle$ and the pair $\langle \bar{a}, B \rangle$, where $c_i \in \{0,1\}^n$, $\bar{a} \in \{0,1\}^n$, $b_i \in \{0,1\}$, and $0 \leq B \leq n$. The problem is to determine whether there exists a vector $\bar{d} \in \{0,1\}^n$ such that $c_i \cdot \bar{d} \leq b_i$ for $1 \leq i \leq s$ and $\bar{a} \cdot \bar{d} \geq B$.

Proof by reduction from ZIP: Assume, for sake of contradiction, that there is a PAC learning algorithm $\mathcal{A}$ that can properly learn the class of Boolean threshold functions in polynomial time (with membership or random examples). We use $\mathcal{A}$ to decide the Zero-One Integer Programming problem, which is NP-complete.

An instance of ZIP consists of vectors $c_1, \ldots, c_s, \bar{a} \in 0,1^n$, integers $b_1, \ldots, b_s \in 0,1$ and B (with $0 \leq B \leq n$). We need to determine if there exists $d \in 0,1^n$ such that for all $1 \leq i \leq s$, $c_i \cdot d \leq b_i$, and $\bar{a} \cdot d \geq B$. Construct the following labeled sample S for a learning problem:

- For each constraint i with $b_i = 0$, include the example $x = c_i$ with label $\ell(x) = 0$.
- For each constraint i with $b_i = 1$, include, for every distinct pair of 1-coordinates (p, q) in c_i, an example $x^{(i)}_{p,q}$ that has $x^{(i)}_{p,q}(p) = x^{(i)}_{p,q}(q) = 1$ and zeros in all other coordinates, with label $\ell(x^{(i)}_{p,q}) = 0$.
- Include the example $x = \bar{a}$ with label $\ell(x) = 1$.

Intuitively, negative examples encode the constraints (violation should yield output 1) and the single positive example encodes the $\bar{a} \cdot d \geq B$ requirement. Now we claim:

1. If a satisfying assignment d exists for the ZIP instance, then there is a threshold function $h(y) = 1 \iff y \cdot d \geq 1$ that correctly classifies all examples in S. Specifically, set the weight vector to d and threshold $k = 1$. For each original constraint i:
   - If $b_i = 0$, d satisfies $c_i \cdot d \leq 0$, hence $c_i \cdot d = 0$ and $h(c_i) = 0$ (correct label).
   - If $b_i = 1$, d satisfies $c_i \cdot d \leq 1$. Then for any pair (p, q) of ones in c_i, at most one of p, q lies in d. Thus $x^{(i)}_{p,q} \cdot d \leq 1$, so $h(x^{(i)}_{p,q}) = 0$ (correct).
   - $\bar{a} \cdot d \geq B \geq 1$, so $h(\bar{a}) = 1$ (correct). Therefore h (a threshold function in our class) perfectly fits all examples in S.
2. If no satisfying d exists, then no threshold function can perfectly classify S. In particular, consider any threshold hypothesis $h_{k,y}(x) = \mathbf{1}y \cdot x \geq k$. We examine its performance on the sample:
   - If h outputs 1 for $\bar{a}$ (to get $\bar{a}$'s label correct), then $y \cdot \bar{a} \geq k$. Since $\bar{a}$ has at most n ones, $k \leq n$. For each constraint i, because d doesn't exist, any y must violate at least one constraint. Suppose y violates constraint j, meaning $c_j \cdot y \geq b_j + 1$ (if $b_j = 0$, then

$c_j \cdot y \geq 1$; if $b_j = 1$, then $c_j \cdot y \geq 2$). Then one of the negative examples derived from c_j will satisfy $y \cdot x \geq k$ and be labeled 1 by h, causing an error. For example, if $b_j = 0$ and $c_j \cdot y \geq 1$, then $h(c_j) = 1$ (since $y \cdot c_j \geq 1 \geq k$) but $\ell(c_j) = 0$. If $b_j = 1$ and $c_j \cdot y \geq 2$, pick two coordinates p, q where y has ones (contributing to the sum); then $y \cdot x^{(j)}_{p,q} \geq 2 \geq k$, so $h(x^{(j)}_{p,q}) = 1$ but $\ell(x^{(j)}_{p,q}) = 0$.

- If h outputs 0 for $\bar{a}$ (to avoid violating some constraint), then $\bar{a} \cdot y < k$. But $\ell(\bar{a}) = 1$, so h fails on $\bar{a}$.

In either case h makes an error on S. In fact, any threshold hypothesis must err on at least one example when no solution exists.

Consequently, if $\mathcal{A}$ could learn threshold functions, we would run $\mathcal{A}$ on the sample S. With high probability, it would output a hypothesis h. We then simulate h on the examples in S to check its accuracy. If h labels all examples correctly (zero errors), we conclude a satisfying d exists (and indeed y from h is a valid solution for ZIP). If h has any error on S, we conclude no solution exists. This procedure runs in randomized polynomial time, deciding the NP-complete problem ZIP. Therefore, NP = RP would follow. By contrapositive, if NP $\neq$ RP, no such polynomial PAC learning algorithm $\mathcal{A}$ can exist. In other words, the class of boolean threshold functions is not PAC-learnable by (proper) threshold hypotheses under the NP $\neq$ RP assumption.

Problem 3 (15 pts)**. Exact Learning DNFs via Membership Queries.** Prove that the class of $(\log n)$-term DNF is learnable in polynomial time in the exact learning model using membership and equivalence queries.

Hint: Reduce this problem to that of learning a DFA (deterministic finite automaton).

We show how to learn a DNF formula with at most $\log n$ terms by reducing the problem to learning a polynomial-size DFA.

Construction of DFA: Let c be the unknown DNF, consisting of $t \leq \log n$ terms $T_1, \ldots, T_t$ (each T_j is a conjunction of literals). We construct a DFA M over the alphabet $\{0, 1\}$ that reads strings of length n (each n-bit string is an input to c) and accepts exactly those strings x for which $c(x) = 1$. The state of M after reading a prefix of the input will record which of the t terms have not yet been falsified by that prefix. Formally, define the state set

$$Q = \{S \subseteq \{1, \ldots, t\} : S \text{ is the set of terms consistent with the prefix read so far}\}.$$

Initially, M starts in state $S_0 = \{1, \ldots, t\}$ (all terms are alive). The transition function δ is defined as follows: if the automaton is in state $S \subseteq [t]$ and the next input bit is b in position i, then

$$\delta(S, b) = \{j \in S : \text{the } i\text{-th literal of term } T_j \text{ is consistent with bit } b\}.$$

After reading all n bits, the DFA accepts if and only if $S \neq \emptyset$, i.e., at least one term is satisfied. Since there are at most $\log n$ terms, the number of subsets S is at most $2^{\log n} = n$, so M has at most n states.

Learning via Angluin's `L*` algorithm: We can simulate the exact learning of M using Angluin's algorithm, which requires membership and equivalence queries. A membership query to M can be answered by querying c directly. An equivalence query for a hypothesized DFA can also be answered by querying c for agreement. Since M has $O(n)$ states, the DFA learning algorithm runs in polynomial time. Once M is learned, it can be converted back into a DNF with at most $\log n$ terms. Thus, $(\log n)$-term DNF is exactly learnable in polynomial time with membership and equivalence queries.

Problem 4. (16pt) **Teaching dimension.** Let X be a finite instance space. Given a concept class $\mathcal{C}$ and a target concept $c \in \mathcal{C}$, we say that a sequence T of labelled examples is a *teaching sequence* for c in $\mathcal{C}$ if c is the only concept in $\mathcal{C}$ that is consistent with T. Let $T(c)$ be the set of all teaching sequences for c in $\mathcal{C}$. The *teaching dimension* of concept class $\mathcal{C}$ is then defined to be

$$\mathrm{TD}(\mathcal{C}) = \max_{c \in \mathcal{C}} \min_{T \in T(c)} |T|$$

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

1. (4pt) Give an example of a concept class $\mathcal{C}$ for which $\mathrm{TD}(\mathcal{C}) > \text{VC-dim}(\mathcal{C})$.
2. (4pt) Give an example of a concept class $\mathcal{C}$ for which $\mathrm{TD}(\mathcal{C}) < \text{VC-dim}(\mathcal{C})$.
3. (4pt) Show that for any concept class $\mathcal{C}$, $\mathrm{TD}(\mathcal{C}) \leq |\mathcal{C}| - 1$.
4. (4pt) Show that for any concept class $\mathcal{C}$, $\mathrm{TD}(\mathcal{C}) \leq \text{VC-dim}(\mathcal{C}) + |\mathcal{C}| - 2^{\text{VC-dim}(\mathcal{C})}$.

Problem 5 (15 pts)**. Persistent Classification Noise Model.** For learning from noisy examples using membership queries, the random *persistent* classification noise model is as follows: Given concept $c \in \mathcal{C}$ and noise rate η, a noisy concept c' is produced by flipping the label of c at each point with probability η. Then a learning algorithm gets all examples and membership queries labeled according to c'. This implies that in this model, with certain (albeit negligible) probability, c' might be very far from c. For simplicity, we only require that a learning algorithm succeeds with probability at least $1/2$, where the probability of success also depends on the random choice of c', and, as before, the running time can be polynomial in

$$\frac{1}{1-2\eta}.$$

Let $\mathcal{P}$ be the class of parity functions over $\{0,1\}^n$ and $\mathcal{U}$ be the uniform distribution over $\{0,1\}^n$.

(a) (5pts) Give an algorithm that learns $\mathcal{P}$ using membership queries in the presence of random persistent classification noise (that is, for any $\eta < 1/2$).

(b) (5pts) Now assume that c' is chosen by an adversary such that

$$\mathbf{Pr}_{\mathcal{U}}[c'(x) \neq c(x)] \leq \eta.$$

Give an algorithm that learns $\mathcal{P}$ using membership queries in the presence of such malicious noise of rate $\eta = 1/5$.

(c) (5pts) Prove that $\mathcal{P}$ is not learnable using membership queries with the malicious noise as above of rate $\geq 1/4$.