Computational Learning Theory
6 : Learning with Membership and Equivalence Queries

Lecturer: Varun Kanade

In the PAC learning framework, we receive labelled examples $(x, c(x))$, where x is drawn from some distribution over the instance space X, and $c(x) \in \{0,1\}$ is the label. We have focused on two different questions—the first regarding sample complexity asks how much data is necessary and sufficient for learning, the second regarding computational complexity asks how much computational power is necessary to run a learning algorithm. For sample complexity questions, notions such as the VC dimension give an answer that is essentially tight, in that the lower and upper bounds on sample complexity are quite close for general concept classes. When considering the question of computational complexity, we make a distinction between *proper learning*, where the learning algorithm is required to output a hypothesis from the concept class that is being learnt, and *improper learning*, where the learning algorithm may output any polynomially evaluatable hypothesis. For proper learning, we have already established that even relatively simple concept classes such as 3-TERM-DNF are hard to PAC-learn unless RP = NP. However, as we have seen it may be possible to learn these classes if the learning algorithm is allowed to output hypotheses from larger classes. We have also established that the class of log-depth circuits is not PAC-learnable under the discrete cube root assumption.

Today, we will consider a richer model of learning that allows the learning algorithm to be more "active". In addition to requesting random labelled examples from the target distribution and concept, we'll allow the learning algorithm to pick an instance $x \in X$ and request the label $c(x)$. We'll investigate this model in greater detail and show that there may be concept classes that can be learnt under this more powerful model of learning, that are not (most likely)[1] PAC-learnable.

1 Exact Learning with Membership and Equivalence Queries

We'll consider learning algorithms that are allowed to make two different types of queries– the first are *membership queries* or *value queries* and the second *equivalence queries*.[2] It is convenient to define the model in terms of oracles, which may be queried by a learning algorithm.

Definition 1 (Membership (Value) Query Oracle)**.** *A membership (or value) query oracle,* $\mathsf{MQ}(c)$*, when given input* $x \in X$ *returns the value* $c(x)$*.*

Next we define the *equivalence oracle*; this oracle takes as input (a representation of) $h : X \to \{0,1\}$ and either agrees that h is *identical* to the target concept c, or returns a "counterexample" x such that $h(x) \neq c(x)$, a proof that c and h are not equivalent. Formally, we define:

Definition 2 (Equivalence Oracle)**.** *An equivalence oracle,* $\mathsf{EQ}(c)$*, when given a representation of a hypothesis,* $h : X \to \{0,1\}$*, either returns "Yes" indicating that* h *and* c *are equivalent as boolean functions, or a counterexample* $x \in X$*, such that* $c(x) \neq h(x)$*.*

We can now define learning with access to membership and equivalence query oracles. Unlike the case of PAC-learning, we do not allow any failure probability, as there is no randomness in

[1] The reason we say most likely is that all hardness results we can establish are conditional. They require assumptions such as the discrete cube root assumption or something else to establish their truth.

[2] The name membership query originated from the fact that boolean functions may be viewed as subsets of the instance space; the instances that evaluate to 1 are members of the set and those that evaluate to 0 are not. Querying the value of a boolean function at a point can be thought of as querying the membership of this point in this set. The name *value query* is more suitable as it can be applied to non-boolean functions as well.

the data generation process, and we also do not allow any error at all! (It may make sense to consider randomised algorithms and allow these to fail with some small probability; however, for simplicity we'll only discuss deterministic algorithms.)

Definition 3 (Exact Learning with $\mathsf{MQ} + \mathsf{EQ}$)**.** *We say that a concept class C is efficiently learnable from membership and equivalence queries, if there exists a polynomial $p(\cdot, \cdot)$ and a learning algorithm L, such that for all $n \geq 1$, for all $c \in C_n$, L when given access to the oracles $\mathsf{MQ}(c)$ and $\mathsf{EQ}(c)$, and input $\mathrm{size}(c)$, halts in time $p(n, \mathrm{size}(c))$ and outputs a hypothesis $h_n : X_n \to \{0, 1\}$, such that for each $x \in X_n$, $h(x) = c(x)$, i.e., h is equivalent to c.*

In Section 2, we will show that the class of monotone DNF formulae is learnable when membership and equivalence queries are allowed. However, let us first comment about the suitability of this framework as a mathematical model for machine learning. We could imagine that one may be able to find *expert* human labellers to provide responses that count as membership (value) queries. It is harder to argue that one can simulate an *equivalence* query oracle. In a sense, this oracle can be treated as a mathematical convenience. In one of the problems on the sheets, you are asked to show that any concept class that is exact learnable using membership and equivalence query oracles, is PAC-learnable if the learning algorithm is also allowed acces to the membership query oracle $\mathsf{MQ}(c)$, in addition to the example oracle $\mathsf{EX}(c, D)$ in the standard definition of PAC-learning.

2 Exact Learning MONOTONE-DNF using $\mathsf{MQ} + \mathsf{EQ}$

In this section, we show that the concept class MONOTONE-DNF that is not known to be PAC-learnable is in fact *exact learnable* if membership and equivalence queries are allowed. A term is a conjunction over the literals; we say that a term is *monotone* if the conjunction only contains *positive* literals, *i.e.,* literals that are variables (but not their negations). A monotone DNF formula is a disjunction of monotone terms. The class MONOTONE-DNF consists of concepts that can be expressed as monotone DNF formulae.

Let c be a monotone DNF formula that contains s terms. Any term T_i that is part of c can be associated with a subset $S_i \subseteq [n]$, *i.e.,* $T_i \equiv \bigwedge_{j \in S_i} x_j$. We assume that c is of the form that if T_i and T_j are both terms of c, with S_i and S_j being the corresponding subsets of variables appearing in them, then it is not the case that $S_i \subseteq S_j$. If it were the case, dropping T_j from c would yield a formula that represents the same boolean function.

Alg. 1 presents an algorithm for learning MONOTONE-DNF using membership and equivalence queries. We will prove the following theorem.

Theorem 4. *The class* MONOTONE-DNF *is exactly learnable.*

Proof. Let c be the target monotone DNF formula. Let n denote the number of variables and let s be the number of terms in c, where no term is redundant, *i.e.,* there isn't a term that implies another. The learning algorithm is allowed running time that is polynomial in n and s.

We argue that every iteration of the while loop on Line 3 of Alg. 1 finds a term T that is present in the target monotone DNF formula c, that we have not yet included in φ. First, we establish that if $\varphi(x) = 1$ at any stage in the algorithm, then $c(x) = 1$. Clearly, it is the case at the beginning of the algorithm; we'll show that if it holds at the beginning of the while loop (Line 3), then it continues to hold at the next iteration of the while loop.

Since, $\varphi(x) = 1$ implies $c(x) = 1$, any counterexample a that establishes that $\varphi \not\equiv c$ must be such that $c(a) = 1$ and $\varphi(a) = 0$. Let $P = \{i \mid T_i(a) = 1\}$ denote the indices of terms in c that are satisfied by a. As $c(a) = 1$, we know that P is non-empty. We claim that when the while loop ends (Line 21), it is the case that exactly one i in P is such that $T_i(a) = 1$, where a is now the updated assignment. Clearly, there is at least one such i (ensured by Lines 15, 16); if there

Algorithm 1 Learning MONOTONE-DNF using MQ + EQ oracles

1: Let $\varphi \equiv 0$ ▷ Always predict false
2: Let $s = 0$ ▷ Determine whether we have succeeded
3: **while** $s = 0$ **do**
4: Let ans be the response of $\mathsf{EQ}(c)$ to query φ
5: **if** ans = "Yes" **then**
6: $s \leftarrow 1$
7: `break`
8: **else**
9: Let $a =$ ans be the counterexample ▷ It must be that $\varphi(a) = 0$ and $c(a) = 1$
10: Let $S = \{i \mid a_i = 1\}$
11: **for** $j \in S$ **do**
12: $a' \leftarrow a$
13: $a'_j \leftarrow 0$
14: Let y be response to $\mathsf{MQ}(c)$ with query a'
15: **if** y = 1 **then** $a \leftarrow a'$
16: **end if**
17: **end for**
18: **end if**
19: $T \leftarrow \{i \mid a_i = 1\}$
20: $\varphi \leftarrow \varphi \vee \left(\bigwedge_{j \in T} x_i \right)$
21: **end while**

were two, say i and i', then let j be the first index such that $j \in T_i$ but $j \notin T_{i'}$—setting $a_i = 0$ would have continued to have $T_{i'}$ satisfied and hence this is what would have happened after Lines 15, 16. A similar argument also shows that all bits of a that could be 0 and still have a be a satisfying assignment of some term T_i for $i \in P$, would have been set to 0. Thus, Line 21 finds a new term that appears in c, but not in φ and adds it to φ. This also shows that at the end of the while loop, it continues to be the case that $\varphi(x) = 1$ implies $c(a) = 1$. Overall, a new term can only be added s times to φ, after which the algorithm has exactly identified c. □

Discusssion

One of the problems on the problem sheets also asks you to show that in the PAC-learning framework (without the membership query oracle), learning DNF reduces to learning MONOTONE-DNF. Currently, we know of no algorithm for learning either DNF or MONOTONE-DNF using only random examples. In fact, Angluin and Kharitonov (1991) have shown that for learning DNF, membership queries constitute no additional benefit, under a widely believed cryptographic assumption. This suggests that allowing membership queries is powerful, in the sense that concept classes that are probably not learnable in the PAC-setting without membership queries, become learnable when membership queries are allowed, *viz.* the class MONOTONE-DNF.

References

Dana Angluin and Michael Kharitonov. When won't membership queries help? In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 444–454. ACM, 1991.

Michael J. Kearns and Umesh K. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, 1994.