Seminar in Computational Learning 21.12.1999

Lecture 9: Learning Boxes in High Dimension

Authors: Amos Beimel and Eyal Kushilevitz
Scribe: Ziv Ben-Eliahu

1 Introduction

The learnability (under various learning models) of geometric concept classes was studied in many papers. A particular attention was given to the case of discrete domains of points (i.e., $\{0, \ldots, \ell-1\}^n$) and concept classes which are defined as unions of boxes in this domain. We will see the definition of boxes in Section 2.

1.1 How Do Boxes Relate To Us

One of the reasons that unions of boxes seem to be interesting concepts is that they naturally extend DNF formulae (in other words, in the case $\ell = 2$ any union of t boxes is equivalent to a DNF formula with t-terms). That is, a box can be viewed as a conjunction of non-boolean properties of the form "the attribute x_i is in the range between a_i and b_i." Similar to the special case of DNF functions, the learnability of unions of boxes in time poly$(n, t, \log \ell)$ (where t is the number of boxes in the union) is an open problem in all models of learning. Note that to represent such a function $\Theta(t \cdot n \cdot \log \ell)$ bits are required. Hence efficiency is defined as polynomial in t, n, and $\log \ell$. Research on the problem of learning the class of unions of boxes (again, with similarity to the case of DNF formulae) attempts to learn sub-classes of this class. There are two main directions:

(1) Sub-classes in which the number of dimensions, n, is limited to $O(1)$. In this case unions of boxes (and more general geometric concept classes) are known to be learnable in the PAC model, and even in the weaker on-line model (i.e., using equivalence queries only).

(2) Sub-classes in which the number of boxes in the union is limited to $O(1)$ (but the number of dimensions in not restricted). Again, this sub-class is learnable in the PAC model, and in the on-line model.

1.2 Motivation

In this work we generalize some of the state-of-the-art results in *exact learning* of DNF formulae, hence strengthening several results in direction (2) above. In particular we show:

1. The class of all unions of $O(\log n)$ boxes can be learned in time $\text{poly}(n, \log \ell)$ (solving an open problem. Until now, this class was shown to be learnable in time $\text{poly}(n, \ell)$). This generalizes a similar result for the class of $O(\log n)$-term DNF.

2. The class of all unions of disjoint boxes (and, more generally, unions of boxes in which each point belongs to at most $O(1)$ boxes) can be learned in time $\text{poly}(n, t, \log \ell)$, where t is the number of boxes. (Until now, this class was shown to be learnable in time $\text{poly}(n, t, \ell)$). This generalizes similar results for learning disjoint DNF and satisfy-$O(1)$ DNF. In particular our algorithm learns the class of decision trees with comparison nodes in time $\text{poly}(n, t, \log \ell)$, where t is the number of leaves (it was shown that this class is learnable in time $\text{poly}(n, t, \ell)$).

In short, the main result of this paper: we show how to (adaptively) select a "small" subset of the domain $\{0, 1, \ldots, \ell - 1\}$ which is sufficient for the learning. Hence, we give a reduction that converts *any* algorithm for learning unions of boxes whose complexity is polynomial in ℓ to an algorithm with complexity which is polynomial only in $\log \ell$ (and the other parameters of the problem).

2 Preliminaries

2.1 The Learning Model

The learning model we use is the *exact learning* model.

2.2 Classes of Boxes

In this section we define the classes of boxes that we learn in this paper. We consider unions of n-dimensional boxes in $[\ell]^n$ (where $[\ell]$ denotes the set $\{0, 1, \ldots, \ell - 1\}$). Formally, a box in $[\ell]^n$ is defined by two corners $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ (in $[\ell]^n$) as follows:

$$B_{a_1,\ldots,a_n,b_1,\ldots,b_n} = \{(x_1, \ldots, x_n) \ : \ \forall i, \ a_i \leq x_i \leq b_i \ \}.$$

We view such a box as a boolean function that gives 1 for every point in $[\ell]^n$ which is inside the box and 0 to each point outside the box. More generally, the boolean function corresponding to the *union* of boxes $B_1, \ldots, B_t$ is defined to be 1 for every point in $[\ell]^n$ which is inside (at least) one of the t boxes and 0 otherwise.

Denote by BOX_t the set of all functions that correspond to unions of (at most) t boxes and by DISJ-BOX_t the set of all functions that correspond to unions of (at most) t *disjoint* boxes. For the case $\ell = 2$ the class BOX_t corresponds to the class of t-term DNF and the class DISJ-BOX_t corresponds to the class (t-term) disjoint DNF.

Notation: Given a set $L \subseteq [\ell]$ (such that $0 \in L$) and a letter $a \in [\ell]$ we denote $\lfloor a \rfloor \triangleq \max\{\sigma \in L : \sigma \leq a\}$. Whenever we use these notations the ground set L will be clear from the context. Denote by f_L the function f restricted to the range L^n, that is $f_L : L^n \to \{0,1\}$ and $f_L(x) = f(x)$ for every $x \in L^n$. Note that if f is a union of at most t boxes then so is f_L.

3 Learning Boxes Using Hankel Matrices

The purpose of this section is to define a "Black-Box"-algorithm for us. This "Black-Box"-algorithm was presented in lecture 5-6.

Theorem 1 *There is an algorithm, Learn-Hankel, that learns the class* $\text{BOX}_{O(\log n)}$ *in time* $\text{poly}(n, \ell)$.

Theorem 2 *There is an algorithm, Learn-Hankel, that learns the class* DISJ-BOX *in time* $\text{poly}(n, t, \ell)$ *(where t is the number of boxes in the target functions).*

4 Reducing the Dependency on ℓ

In this section we reduce the dependency of our algorithm on ℓ. For this, we define the notion of *sensitive* letters:

Definition 3 *A letter $\sigma \in [\ell]$ is called i-*sensitive *with respect to f if there exist letters $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n \in [\ell]$ such that*

$$f(c_1, \ldots, c_{i-1}, \sigma - 1, c_{i+1}, \ldots, c_n) \neq f(c_1, \ldots, c_{i-1}, \sigma, c_{i+1}, \ldots, c_n).$$

A letter σ is called sensitive *with respect to f if σ is i-sensitive for some i.*

Lemma 4 *Let f be a function in BOX_t. Then, there are at most $2nt$ sensitive letters with respect to f.*

Proof: If $f(c_1 \ldots c_{i-1}, \sigma - 1, c_{i+1} \ldots c_n) = 0$ and $f(c_1 \ldots c_{i-1}, \sigma, c_{i+1} \ldots c_n) = 1$ then for some box B_j the letter σ is the i-th coordinate of the lower corner. If $f(c_1, \ldots, c_{i-1}, \sigma - 1, c_{i+1}, \ldots, c_n) = 1$ and $f(c_1, \ldots, c_{i-1}, \sigma, c_{i+1}, \ldots, c_n) = 0$ then for some box B_j the letter $\sigma - 1$ is the i-th coordinate of the upper corner. Since there are at most t boxes, in n dimensions, and each is defined by its two corners, the lemma follows. ■

Lemma 5 *If L contains all sensitive letters then $f(x_1, \ldots, x_n) = f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$.*

Proof: Let's assume that $f(x_1, \ldots, x_n) \neq f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$ and reach a contradiction – we will find a sensitive letter not in L.

First, we build the sequence $f(x_1, \ldots, x_n), f(x_1, \ldots, x_{n-1}, \lfloor x_n \rfloor), \ldots, f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$. From our assumption, the first and the last elements of the sequence are different. Thus, there are two different adjacent elements in the sequence:

$$f(x_1, \ldots, x_i, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor) \neq f(x_1, \ldots, \lfloor x_i \rfloor, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor). \tag{1}$$

Let's look at the sequence for these two elements (remember, the range is of integers): $f(x_1, \ldots, x_i, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor)$, $f(x_1, \ldots, x_i, \lfloor x_{i+1} \rfloor - 1, \ldots, \lfloor x_n \rfloor)$, $\ldots$, $f(x_1, \ldots, \lfloor x_i \rfloor + 1, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor)$, $f(x_1, \ldots, \lfloor x_i \rfloor, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor)$.

Again, this sequence has two different adjacent elements: there is some $\lfloor x_i \rfloor \leq \sigma \leq x_i$ such that,

$$f(x_1, \ldots, \sigma - 1, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor) \neq f(x_1, \ldots, \sigma, \lfloor x_{i+1} \rfloor, \ldots, \lfloor x_n \rfloor). \tag{2}$$

So, σ is a sensitive letter by definition. But, there is no sensitive letter between $\lfloor x_n \rfloor$ and x_n - contradiction. ■

Corollary 6 *Given set L and a point $(x_1, \ldots, x_n)$ such that $f(x_1, \ldots, x_n) \neq f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$, we can find a sensitive letter that isn't in L on polynomial time in $\log n$ and $\log \ell$.*

Proof: We first find an index i such that Equation (1) holds. This is done asking $O(\log n)$ membership queries using binary search. We than find $\lfloor x_i \rfloor \leq \sigma \leq x_i$ such that Equation (2) holds. This is done asking $O(\log \ell)$ membership queries using binary search. ■

In Fig. 1 we describe an algorithm to learn the classes BOX$_{O(\log n)}$ and DISJ-BOX. The idea behind the algorithm is to learn the set of sensitive letters with respect to f (using Corollary 6) as part of learning the function f. At each stage, with the set of current letters, denoted L, the algorithm tries to learn the function using the algorithm LEARN_ HANKEL as a black box. Either the algorithm succeeds, or it finds another sensitive letter and starts a new execution of the algorithm LEARN_ HANKEL.

While executing the algorithm LEARN_ HANKEL as a black box, the algorithm simulates the membership and equivalence queries to f_L asked by this black box, using its membership and equivalence oracles for f. To answer a membership query about f_L, it simply uses the oracle for f (this is correct since $f_L(x) = f(x)$ for every $x \in L^n$). To simulate an equivalence query $EQ(h_L)$ to f_L, using the corresponding oracle for f, the hypothesis h_L is extended to a hypothesis $h : [\ell]^n \to \{0, 1\}$ by $h(x_1, \ldots, x_n) = h_L(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$. If the equivalence query $EQ(h)$ returns a counterexample y then there are two cases. If $f(y_1, \ldots, y_n) = f(\lfloor y_1 \rfloor, \ldots, \lfloor y_n \rfloor)$ then $(\lfloor y_1 \rfloor, \ldots, \lfloor y_n \rfloor)$ is a counterexample to h_L, and the algorithm passes this counterexample to the black box. Otherwise, if $f(y_1, \ldots, y_n) \neq f(\lfloor y_1 \rfloor, \ldots, \lfloor y_n \rfloor)$, then we find a sensitive letter with $\log n + \log \ell$ membership queries using Corollary 6. Then, this sensitive letter is added to L and a new execution of LEARN_ HANKEL is started.

Using the algorithm LEARN_ SENSITIVE we prove the following results:

Theorem 7 *The class* $\text{BOX}_{O(\log n)}$ *can be learned in time* $\text{poly}(n, \log \ell)$.

Proof: We use the algorithm LEARN_ SENSITIVE to learn the class $\text{BOX}_{O(\log n)}$. As remarked, for every L the function f_L is also a union of $O(\log n)$ boxes. By Corollary 1, learning the function f_L (Step (2)) ends within time $\text{poly}(n, |L|)$. It ends either by identifying the target function f, or by adding a new sensitive letter to L (Step (3)). In addition, once L contains all sensitive letters then, by Lemma 5, for every point $(y_1, \ldots, y_n) \in [\ell]^n$,

$$f(y_1, \ldots, y_n) = f(\lfloor y_1 \rfloor, \ldots, \lfloor y_n \rfloor) \ . \tag{3}$$

At this point, since there are no more sensitive letters, the algorithm for f_L will find a hypothesis $h_L \equiv f_L$ which by Equation (3) implies $h \equiv f$. By Lemma 4 the size of L, and hence the number of times Step (3) is executed, is $O(n \log n)$. Each time that a new sensitive letter is inserted to L we spend $\text{poly}(n, \log \ell)$ time in searching for such a letter. All together, algorithm LEARN_ SENSITIVE learns the class $\text{BOX}_{O(\log n)}$ in time $\text{poly}(n, \log \ell)$. ■

The following theorem shows the learnability of disjoint boxes in time $\text{poly}(n, t, \log \ell)$.

Theorem 8 *The class* DISJ-BOX *can be learned in time* $\text{poly}(n, t, \log \ell)$ *(where t is the number of boxes in the target function).*

Proof: Again, we use the algorithm LEARN_ SENSITIVE to learn this class. By Lemma 4 the size of L, and hence the number of times Step (3) is executed, is $O(nt)$. At each time that a new sensitive letter is inserted to L we spend $\text{poly}(n, \log \ell)$ time in searching for such a letter. By Theorem 2 learning the function f_L (Step (2)) takes time $\text{poly}(n, t, |L|)$ (observe that the function f_L always consists of a union of at most t disjoint boxes). All together, algorithm LEARN_ SENSITIVE learns any function in DISJ-BOX_t in time $\text{poly}(n, t, \log \ell)$. ■

A special case of disjoint boxes are functions which are represented by decision trees where each node contains a boolean query of the form "Is $x_i \geq \theta$?" (and where the variables x_i and the constants θ take values in $[\ell]$). It was shown that this class is learnable in time $\text{poly}(n, t, \ell)$ (where t here denotes the number of leaves in the tree corresponding to f). Algorithm LEARN_ SENSITIVE shows that this class can in fact be learned in time $\text{poly}(n, t, \log \ell)$.

Algorithm learn_ sensitive:

1. $L \leftarrow \{0\}$

2. Learn the function f_L using the algorithm learn_ hankel.
   To answer membership queries about f_L simply use the oracle for f.
   To simulate an equivalence query $EQ(h_L)$ to f_L:

   (a) Define $h : [\ell]^n \rightarrow \{0,1\}$ as $h(x_1,\dots,x_n) = h_L(\lfloor x_1 \rfloor, \dots, \lfloor x_n \rfloor)$. Ask whether $h \equiv f$.
   If the answer is "YES" halt with output h.
   Otherwise, we have a counterexample $(y_1,\dots,y_n) \in [\ell]^n$.

   (b) If $f(y_1,\dots,y_n) = f(\lfloor y_1 \rfloor, \dots, \lfloor y_n \rfloor)$ (this is checked using a membership query) then $(\lfloor y_1 \rfloor, \dots, \lfloor y_n \rfloor)$ is a counterexample to h_L – pass this counterexample to the algorithm for learning f_L and continue its execution.
   Otherwise, proceed to Step (3).

3. $(f(y_1,\dots,y_n) \neq f(\lfloor y_1 \rfloor, \dots, \lfloor y_n \rfloor).)$
   Find an index $1 \leq i \leq n$ such that

$$f(\lfloor y_1 \rfloor \dots \lfloor y_{i-1} \rfloor, y_i \dots y_n) \neq f(\lfloor y_1 \rfloor \dots \lfloor y_{i-1} \rfloor, \lfloor y_i \rfloor, y_{i+1} \dots y_n)$$

   (this is found with $O(\log n)$ MQs using a binary search).
   Then, find a letter σ such that $\lfloor y_i \rfloor + 1 \leq \sigma \leq y_i$ and

$$f(\lfloor y_1 \rfloor ... \lfloor y_{i-1} \rfloor, \sigma - 1, y_{i+1} ... y_n) \neq f(\lfloor y_1 \rfloor ... \lfloor y_{i-1} \rfloor, \sigma, y_{i+1} ... y_n)$$

   (this is found with $O(\log \ell)$ MQs using a binary search).
   Set $L \leftarrow L \cup \{\sigma\}$ and start Step (2) again.

Figure 1: An algorithm for learning the sensitive letters.

Remark 9 Obviously, Theorem 2 and Theorem 8 can be easily extended to the case where no point x is contained in more than s boxes, for $s = O(1)$. Also, Theorem 7 can be extended to learning decision trees of depth $O(\log n)$, where each node contains a boolean query of the form "does x belong to a box B?"

In particular this shows the learnability of *any* function of $O(\log n)$ boxes, and not only *unions* of boxes.