Decision Trees have Approximate Fingerprints

Víctor Lavín[1]
Universidad Politécnica de Cataluña[2]
Email: vlavin@lsi.upc.es

Vijay Raghavan[3]
Vanderbilt University[4]
Email: raghavan@vuse.vanderbilt.edu

NeuroCOLT Technical Report Series

NC-TR-97-016

March 1997[5]

Produced as part of the ESPRIT Working Group
in Neural and Computational Learning,
NeuroCOLT 8556

NeuroCOLT Coordinating Partner

Department of Computer Science
Egham, Surrey TW20 0EX, England

For more information contact John Shawe-Taylor at the above address
or email neurocolt@dcs.rhbnc.ac.uk

[1]This work was done while author Víctor Lavín was visiting the Computer Science Department of Vanderbilt University, supported by FP93 13717942 grant from the Spanish Government.

[2]Dept. of Software (LSI), Universidad Politécnica de Cataluña, Pau Gargallo 5, Barcelona 08028, Spain

[3]Supported by NSF grant CCR-9510392

[4]Dept. of Computer Science, Box 1679-B, Vanderbilt University, Nashville, TN 37235, USA

[5]Received February 28, 1997

Abstract

We prove that decision trees exhibit the "approximate fingerprint" property, and therefore are not polynomially learnable using only equivalence queries. A slight modification of the proof extends this result to several other representation classes of boolean concepts which have been studied in computational learning theory.

1 Introduction

Exact learning via queries is a well-studied model in computational learning. Of the many kinds of queries considered by Angluin [**Ang88**], membership and equivalence queries have established themselves as the standard combination to be used for exact learning.

Some results that characterize learnability with a polynomial number of polynomially sized queries are known. For example, Angluin [**Ang90**] showed that the presence of the *approximate fingerprint* property for a concept class is a sufficient condition for non-learnability with equivalence queries alone. Using this characterization, Angluin showed that equivalence queries do not suffice for learning deterministic or nondeterministic finite automata, context-free grammars, or general CNF or DNF formulas. Later on, Gavaldà [**Gav93**] proved this property to be not only sufficient but also a necessary condition. On the other hand, Goldman and Kearns [**GK91**] showed that if a concept class is learnable with a polynomial number of membership queries alone, then it has *polynomial teaching dimension*. Moreover, the converse holds provided the concept class is projection closed, as shown by Hellerstein et al. [**HPRW95**]. Finally, it is known [**HPRW95**] that a concept class is learnable with a polynomial number of equivalence and membership queries if and only if that class has *polynomial certificates*.

Decision trees are a popular representation of boolean functions, which form the basic inference engine used in many machine learning programs. The learnability of decision trees has been well-studied in the learning theory community. For example, it is known [**Han90**] that μ-decision trees are exactly learnable with equivalence and membership queries, but neither type of query alone suffices for polynomial learning. Bshouty [**Bsh95**] has shown that general decision trees are learnable with extended equivalence queries (the hypotheses used are depth-three formulas) and membership queries. It is not known whether general decision trees are learnable with proper equivalence queries and membership queries, nor is it known whether decision trees are properly PAC-learnable with or without membership queries.

In this paper, we focus on showing the presence of the approximate fingerprint property for some representation classes of boolean concepts, thereby showing that they are not exactly learnable with equivalence queries alone. In particular, we show that general decision trees, CDNF formulas, self-dual DNF formulas, $\log n$–DNF $\cap$ $\log n$–CNF formulas, and $\log n$-depth branching programs are all not exactly learnable with equivalence queries alone. These results complement the known learnability results for these classes.

2 Preliminaries

We use standard terminology in learning theory. See [**Ang90**] for more detailed descriptions of the learning model and a formalization of *concept classes*, *representation classes*, *examples*, etc.

We consider learning algorithms that make equivalence queries to a teacher. The teacher answers the queries according to a target concept $c \in \mathcal{C}$. More precisely, a query is a representation $r \in \mathcal{R}$ of some concept in $\mathcal{C}$, and the answer is either YES, if the concept represented by r coincides with c, or a counterexample in the symmetric difference of c and the concept associated to r, otherwise.

We are interested in boolean concepts or, in other words, functions defined over a finite set of variables $\{x_1, \ldots, x_n\}$. The classes of representations we consider here are the following:

1. *DT (Decision Trees).* A decision tree d is a binary tree where the leaves are labeled either 0 or 1, and each internal node is labeled with a variable. Given an *assignment* $\alpha \in \{0,1\}^n$, $d(\alpha)$ is evaluated by starting at the root and iteratively applying the following rule, until a leaf is reached: let the variable at the current node be x_i; if the value of α at position i is 1 then branch right; otherwise branch left. If the leaf reached is labeled 0 (resp. 1) then $d(\alpha) = FALSE$ (resp. $TRUE$). The size of a decision tree is its number of leaves. The decision-tree size (DT-size) of a boolean function f is the size of the smallest decision tree that can represent f.

2. *CDNF.* A CDNF formula is a pair (f, g), where f is a DNF formula and g is a CNF formula and $f \equiv g$. It is convenient to define the size of such a formula as $\max\{n, m, p\}$, where n is the number of variables over which the formulas f and g are defined, m is the number of terms in f, and q is the number of clauses in g. The CDNF-size of a boolean function is the size of the smallest CDNF formula that can represent it.

   Equivalent but slightly different definitions of CDNF formulas and their sizes can be found in [**Bsh95**] and [**HPRW95**]. In [**Bsh95**], it is shown that CDNF formulas can be exactly learned in polynomial time with extended equivalence queries (of depth-3 formulas) and membership queries. In [**HPRW95**], it is shown that CDNF formulas are polynomial-query learnable with proper equivalence queries and membership queries. Here, we show that this latter result is tight: we show that proper equivalence queries alone do not suffice for polynomial-query learnability.

3. *Self-Dual DNF (Self-Dual CNF).* A DNF (CNF) formula f is self-dual if $f(\alpha) \neq f(\bar{\alpha})$ for all $\alpha \in \{0,1\}^n$, where $\bar{\alpha}$ is the bitwise complement of α.

4. $\log n$*–DNF* $\cap$ $\log n$*–CNF.* This class contains the functions representable both as DNF formulas whose terms have at most $\log n$ literals, and CNF formulas whose clauses have at most $\log n$ literals. For our purposes, we consider a representation in this class to be a pair (f, g), where f is a

$\log n$-DNF formula and g is a $\log n$-CNF formula. The size of such a pair is the same as its CDNF size.

There are several results in learning theory about this interesting class. Notice that this class contains the class of $\log n$-depth decision trees, shown to be exactly learnable in polynomial time [**KM93**] with membership queries. (The output of the learning algorithm is not a $\log n$-depth decision tree but a representation based on Fourier coefficients.) In [**BCKT94**], it was shown that $\log n\text{–}DNF \cap \log n\text{–}CNF$ is learnable with membership queries and an NP-oracle, in a representation of depth-3 boolean formulas. Finally, in [**HPRW95**] it was shown that this class is properly learnable with membership queries and an oracle for NP $\cap$ co-NP. It is open whether this class is properly learnable with membership queries alone in polynomial time. Here we show that the class is not polynomial-query learnable with equivalence queries alone.

5. *BP (Branching programs).* A branching program is a directed, acyclic graph, with a unique node of in-degree 0 (called the *root*), and two nodes of out-degree 0 (called *leaves*) , one labeled 0, and the other labeled 1; each non-leaf node of the graph contains a variable, and has outdegree exactly two. Assignments are evaluated following the same rule as for decision trees.

   A branching program is in the class $\log n\text{–}depth\ BP$ if its longest path has length at most $\log n$. The size of a branching program is the number of nodes in it.

General branching programs are known to be not learnable with equivalence and membership queries. (See [**Wil95**] for a detailed study of subclasses of branching programs according to learnability.) In [**HPRW95**], it is shown that the subclass of $\log n$-depth branching programs is properly learnable with membership queries and an oracle for NP $\cap$ co-NP—it is open whether the oracle can be dispensed with. Here we show that the class is not polynomial-query learnable with equivalence queries alone.

3 Approximate Fingerprints

We recall the following definitions from [**Ang90**, **Gav93**]. Let $\mathcal{C}$ denote a class of concepts defined over an instance space of Σ^* (where Σ is a finite alphabet, $\mathcal{R}$ a representation class for $\mathcal{C}$, w a string from Σ^*, b an element of $\{0,1\}$, and $\alpha > 0$ a real number. We say that the pair $\langle w, b\rangle$ is an *α-approximate fingerprint with respect to C* if

$$|\{c \in \mathcal{C} : \chi_c(w) = b\}| < \alpha|\mathcal{C}|.$$

That is, the number of concepts in $\mathcal{C}$ that agree with the classification b of w is strictly less than the fraction α of the total number of concepts in $\mathcal{C}$.

A *sequence of concept classes* is a sequence $C_1, C_2, C_3, ...$ such that each C_i is a class of concepts. Such a sequence is *polynomially bounded* (with respect to a given representation $\mathcal{R}$) if there exists a polynomial $p(n)$ such that for all $c \in C_n$, there is a representation $r \in \mathcal{R}$ such that $c(r) = c$ and $|r| \leq p(n)$.

A representation of concepts $\mathcal{R}$ is said to have the *approximate fingerprint* property if there exists a polynomially bounded sequence of concept classes $T_1, T_2, T_3, ...$, and a polynomial $p(n)$ such that for any polynomial $q(n)$, for infinitely many n, T_n contains at least two concepts and if $r \in \mathcal{R}$ and $|r| \leq q(n)$ then there exists a string $w \in \Sigma^*$ of length at most $p(n)$ such that $\langle w, \chi_{c(r)}(w) \rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. That is,

$$|\{c \in T_n : \chi_c(w) = \chi_{c(r)}(w)\}| < |T_n|/q(n),$$

where $c(r)$ is the concept associated to r.

Notice that, since we are dealing with boolean concepts defined over n variables, the polynomial $p(n)$, that bounds the length of the words, can be set to n. The notion of approximate-fingerprints was introduced as a means of proving non-learnability of certain classes in the equivalence model. Namely:

Theorem 1 *[**Ang90**]*
Let $\mathcal{R}$ be a representation class of concepts with the approximate fingerprint property. Then there is no algorithm for exact identification of $\mathcal{R}$ using polynomially many equivalence queries of polynomial size.

4 The Results

Our results for all the classes considered here use the same central idea. We first illustrate the technique for the class DT of decision trees.

In order to prove approximate fingerprints for DT, we need the following lemma.

Lemma 2 *Let $k > 0$ be some fixed constant. If d is a decision tree of size at most n^k, over n variables, then there exists an assignment α such that either*

(a) α contains at most $k \log n$ 1's and $d(\alpha) = 1$, or

(b) α contains at most $k \log n$ 0's and $d(\alpha) = 0$.

Proof: Since each of the 2^n assignments reaches some leaf of d after being evaluated, and d has at most n^k leaves, d must have a path p of length at most $\log(n^k) = k \log n$. The assignment α can be constructed by satisfying p first and then filling the rest of the positions with 1's or 0's as necessary.

Theorem 3 *The class DT has approximate fingerprints.*

Proof: Let T_n be the target class of functions that represent the majority of $\log n$ variables chosen out of n variables. Clearly, $|T_n| = \binom{n}{\log n}$, and the DT-size of each function in T_n is at most n. Therefore, it suffices to show that for any fixed constant k, and for infinitely many n, and any DT d of size at most n^k, the assignment α whose existence is guaranteed by the Lemma 2 satisfies the property that fewer than $|T_n|n^{-k}$ of the the functions in T_n classify α the same way as d.

Without loss of generality, assume that $\log n$ is an even integer. (This avoids floors and ceilings.) The number of functions in T_n that accept (reject) an assignment with $k \log n$ 1's (respectively, 0's) is

$$\sum_{p=\frac{\log n}{2}}^{\log n} \binom{k \log n}{p} \binom{n - k \log n}{\log n - p}.$$

Using n^k as an upperbound for $\binom{k \log n}{p}$ this sum turns out to be less than

$$n^k \sum_{p=0}^{\frac{\log n}{2}} \binom{n - k \log n}{p},$$

which is less than

$$n^k (1 + \frac{\log n}{2}) \binom{n - k \log n}{\frac{\log n}{2}} = r(n).$$

Now $\frac{r(n)}{|T_n|}$ can be shown to be less than

$$\frac{n^k (1 + \frac{\log n}{2})(\log n)^{\frac{\log n}{2}}}{(n - \log n + 1)^{\frac{\log n}{2}},}$$

which is less than n^{-k} for $n > 2^{ck}$, where c is some constant. ■

We now show that the remaining classes considered here have approximate fingerprints.

Corollary 4 *The following classes have approximate fingerprints:*

1. *CDNF Formulas*
2. *Self-Dual DNF Formulas and Self-Dual CNF Formulas.*
3. $\log n$*–DNF* $\cap$ $\log n$*–CNF*
4. $\log n$*–depth BP*

Proof: The proof follows easily if we first use the following statements for each of the above classes instead of Lemma 2.

Let $k > 0$ be some fixed constant.

- ◇ *For every CDNF formula $h = (f, g)$ over n variables, of size at most n^k, there exists an assignment α that either (a) contains at most $k \log(2n)$ 1's and $h(\alpha) = 1$, or (b) contains at most $k \log(2n)$ 0's and $h(\alpha) = 0$.*

  To see this, note that the DNF formula $f + \overline{g}$ has at most $2 \cdot n^k$ terms and is identically true, i.e., it accepts all 2^n assignments. Consequently, such a formula must have a term t that accepts at least $\frac{2^n}{2 \cdot n^k}$ assignments, whence the length of t is at most $k \log(2n)$. The assignment α can now be constructed by satisfying the literals in t and then padding the remaining variables with 0's or 1's according to whether t is a term in f or corresponds to a clause in g respectively.

- ◇ *For every self-dual DNF formula f of at most n^k terms over n variables, there exists an assignment α that either (a) contains at most $k \log(2n)$ 1's and $f(\alpha) = 1$, or (b) contains at most $k \log(2n)$ 0's and $f(\alpha) = 0$.*

  Let $\sim f$ denote the DNF formula obtained from f by complementing every literal in every term of f. Then, by the definition of self-dual formulas, it follows that $\sim f \equiv \overline{f}$. Therefore, $f + \sim f$ is identically true and the existence of α is proved in an identical manner to that of CDNF formula h above.

  Note that an analogous statement holds for self-dual CNF formulas.

- ◇ *For every $\log n$–DNF $\cap$ $\log n$–CNF formula f of size at most n^k, there exists an assignment α that either (a) contains at most $\log(n)$ 1's and $f(\alpha) = 1$, or (b) contains at most $\log(n)$ 0's and $f(\alpha) = 0$.*

  This is obvious.

- ◇ *For every $\log n$-depth BP h of size at most n^k over n variables, there exists an assignment α that either (a) contains at most $k \log(n)$ 1's and $h(\alpha) = 1$, or (b) contains at most $k \log(n)$ 0's and $h(\alpha) = 0$.*

  This follows along the same lines as the proof of Lemma 2.

With this change, the proof of Theorem 3 can be used with very slight modifications to show that all of these classes have approximate fingerprints. In particular, the target class T_n remains the same for all the classes, and each function in T_n has a representation of size at most n in each of these classes. ■

A final note. Since the target class defined in Theorem 3 is monotone, the monotone versions of all the classes mentioned above (including DT) also have approximate fingerprints, and are therefore not learnable with equivalence queries alone.

References

[**Ang88**] D. Angluin. Queries and Concept Learning. *Machine Learning*, 2, (1988), 319-342.

[**Ang90**] D. Angluin. Negative Results for Equivalence Queries. *Machine Learning*, 5, (1990), 121-150.

[**Bsh95**] N. Bshouty. Exact Learning Boolean Functions via the Monotone Theory. *Information and Computation*, 123(1):146–153, Nov 1995.

[**BCKT94**] N. Bshouty, R. Cleve, S. Kannan, and C. Tamon. Oracles and Queries that are Sufficient for Exact Learning. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 130–139, 1994.

[**Gav93**] R. Gavaldà. On the Power of Equivalence Queries. *Proceedings of the 2nd European Conference on Computational Learning Theory*, (1993), 193-203.

[**GK91**] S. Goldman, M. Kearns. On the Complexity of Teaching. *Proceedings of the 4rd Workshop on Computational Learning Theory*, (1991), 303-314.

[**Han90**] T. Hancock. Identifying μ-Formula Decision Trees with Queries. *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 23–37, 1990.

[**HPRW95**] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, D. Wilkins. How Many Queries are Needed to Learn? *Proceedings of the 27th ACM Symposium on the Theory of Computing,* (1995), 190-199.

[**KM93**] E. Kushilevitz and Y. Mansour. Learning Decision Trees Using the Fourier Spectrum. *SIAM Journal of Computing*, 22(6):1331–1348, 1993.

[**Wil95**] D. Wilkins. Learning Restricted-Read Branching Programs with Queries. *Ph.D. Thesis*, Vanderbilt University, 1995.