Learning Monotone Term Decision Lists

David Guijarro[1], **Víctor Lavín**[2],
Department LSI, Universitat Politècnica de Catalunya
Pau Gargallo 5, Barcelona 08028, Spain
{david, vlavin}@goliat.upc.es

Vijay Raghavan[3]
Box 1679-B, Computer Science Department
Vanderbilt University, Nashville, TN 37235
raghavan@vuse.vanderbilt.edu

Abstract

We study the learnability of monotone term decision lists in the exact model of equivalence and membership queries. We show that, for any constant $k \geq 0$, k-term monotone decision lists are exactly and properly learnable with $n^{O(k)}$ membership queries in $O(n^{k^3})$ time. We also show $n^{\Omega(k)}$ membership queries are necessary for exact learning. In contrast, both k-term monotone decision lists ($k \geq 2$) and general monotone decision lists are not learnable with equivalence queries alone.

[1]Supported by the Esprit EC program under project 7141 (ALCOM-II), the Working Group 8556 (NeuroColt), and the Spanish DGICYT (project PB92-0709)

[2]Supported by grant FP93 13717942 from the Spanish Government

[3]This work was supported by NSF grant CCR-9510392.

1 Introduction

Decision lists were introduced by Rivest [Riv87], who showed that the class of k-decision lists is properly PAC-learnable in polynomial time, for constant $k \geq 0$. Here the constant k refers to the maximum number of literals in a term of the decision list. Related results on learning decision lists can be found in [Sim94].

It is not easy to generalize Rivest's results to allow arbitrary number of literals in a term. Such a generalization would include all boolean functions—moreover, the smallest decision list representation of a boolean function is no bigger than the DNF representation. Therefore, predictability of general decision lists would imply the predictability of DNF formulas, a hard open problem.

In this paper, we take a different approach to studying the learnability of decision lists with terms of arbitrary length—we restrict the terms to be monotone. This is not as big a restriction as may seem initially: we show that all boolean functions can still be represented with decision lists with monotone terms only. An advantage of this restriction is that monotone term decision lists are amenable to basic operations on representations. For example, we show that it is polynomial-time decidable if a given monotone term decision list is irredundant (i.e., if it contains no irrelevant variables in a term and no irrelevant term) and make it irredundant if it is not. We also show that there is a polynomial-time equivalence test for monotone term decision lists. In contrast, for general decision lists, these operations cannot be done in polynomial time unless $P = NP$.

Our main results in the learnability of monotone term decision lists are as follows. First, we show that k-term decision lists are representable as k-decision lists. Consequently, for any constant k, k-term decision lists are PAC-learnable as k-decision lists using Rivest's algorithm [Riv87]. Proper learnability of k-term monotone decision lists is a more difficult question. We do not know if this class is properly PAC-learnable. However, we do show that this class is properly and exactly learnable with $n^{O(k)}$ membership queries alone in $n^{O(k^3)}$ time by a *non-adaptive* learning algorithm (i.e., the algorithm always makes membership queries on a fixed set of assignments). From this it follows that k-term monotone decision lists are "simple-PAC" learnable in the sense of Li and Vitányi [LV91]. We also show that $n^{\Omega(k)}$ membership queries are necessary for exact learning of this class and that equivalence queries will not suffice for exact learning when $k \geq 2$. Finally, we use Angluin's approximate fingerprint technique [Ang90] to show that equivalence queries do not suffice for exact learning of monotone term decision lists of arbitrary number of terms. An open problem, not resolved in this paper, is whether general monotone term decision lists are exactly learnable with equivalence and membership queries.

The rest of the paper is organized as follows. Section 2 presents definitions used in the rest of the paper. (For definitions of exact/PAC, and improper/proper learnability, the reader is referred to [Ang88, Ang90, AB92, HPRW96, Val84].) Section 3 contains some essential properties of monotone decision lists. Sections 4 and 5, respectively, contain our results for k-term monotone decision lists and general monotone decision lists.

2 Definitions

A *decision list* is an ordered list of pairs of boolean functions. Each pair $\langle t, o \rangle$ in a decision list is called a *node* of the decision list, the function t being the *test* function and the function o the *output* function. The last node in a decision list is called the *default* node and has a constant test function that evaluates to 1 (true). The evaluation of a decision list $\langle \langle t_1, o_1 \rangle, \langle t_2, o_2 \rangle, ... \rangle$ on an assignment α is obtained by first finding the least i such that $t_i(\alpha) = 1$ and then outputting $o_i(\alpha)$. In the following sections, we say that assignment α *reaches* the node $\langle t_i, o_i \rangle$, if i is the least integer such that $t_i(\alpha) = 1$.

A k-decision list [Riv87] is a decision list in which all the test functions are monomials with at most k literals and the output functions are the constants 0 and 1. A *monotone decision list* is a decision list in which all the test functions (or *terms*) are monotone monomials and the output functions are the constants 0 and 1. If the output function associated with a term is 0, we say that the term is *negative*; otherwise it is *positive*. In this paper, we focus on monotone decision lists in general and k-term monotone decision lists in particular, where $k > 0$ is some fixed integer constant. (A k-term monotone decision list has at most k nodes, not counting the default.)

A monotone decision list is *minimal* (or *irredundant*) if no node or variable within a term can be deleted without changing the boolean function represented by the decision list. Without loss of generality, we will suppose that all decision lists considered in this paper are minimal.

We denote the set of assignments with at most l zeroes by A_l. The set of variables that are set to 1 in an assignment α is denoted by $ones(\alpha)$, and respectively, the set of variables that are set to 0 by $zeroes(\alpha)$. The natural partial order $\succ$ over assignments, given by $\alpha \succ \beta$ if and only if $ones(\alpha) \supset ones(\beta)$, defines a lattice called the *boolean lattice*. The set of variables of a term t is denoted by $vars(t)$. Finally, for any assignment α, the assignment $\alpha_{v \leftarrow b}$ is the assignment obtained by setting the variable v to b and all the other variables as in α.

3 Some Properties of Monotone Decision Lists

We prove some essential properties of monotone decision lists, which will be used in the next section to develop a learning algorithm. We begin by showing that the general class of monotone decision lists has the power to represent all boolean functions.

Property 1 *Every boolean function can be represented as a monotone decision list.*

Proof

Let f be any boolean function. To represent f using a monotone decision list, walk through the boolean lattice of assignments in topological order (i.e., starting at the assignment of all 1's and ending at the assignment of all 0's.) For each assignment α, create a node

$\langle t_\alpha, f(\alpha)\rangle$, where t_α is a monotone monomial containing precisely the variables in $ones(\alpha)$. (Note that the evaluation of f on the all 0's assignment will form the output of the default node.) The decision list thus constructed represents f since each assignment α in the boolean hypercube reaches the corresponding node $\langle t_\alpha, f(\alpha)\rangle$. □

We now show that the class of k-term decision lists is contained in the class of k-decision lists considered by Rivest. From this it follows that k-term decision lists are PAC-learnable as k-decision lists, using Rivest's algorithm [Riv87].

Property 2 *Every k-term decision list can be represented by a k-decision list.*

Proof

Let L be a k-term decision list. We transform L to an equivalent k-decision list L' as follows: Initially L' is the empty list. We begin by adding to L' a set of nodes that classify correctly the assignments that reach the default in L, i.e., we add to L' nodes $\langle t, b\rangle$, where b is the classification of the default of L, and each k-DNF term t is obtained by picking one literal from each (non-default) term of L and complementing the at most k literals. Next, for each node from the last node of L down to the second node of L , we use a similar procedure to classify all assignments that reach the node. That is, if $\langle t_i, b\rangle$ is the i-th node of L, we add to L' one node $\langle t, b\rangle$ for every term t obtained by picking one literal from each of the $i-1$ terms of L prior to $\langle t_i, b\rangle$ and complementing the at most $i-1$ literals. Finally, we set the default of L' to be the classification of the first node of L. □

The remaining properties in this section have to do with k-term monotone decision lists. We begin by showing that the set A_{2k} is a specification set [GK91] for k-term monotone decision lists.

Property 3 *Let L_1 and L_2 be k-term monotone decision lists. $L_1 \equiv L_2$ if and only if for all assignments $\alpha \in A_{2k}$, $L_1(\alpha) = L_2(\alpha)$.*

Proof

The "only if" part is trivial. For the "if" part, suppose that α is an assignment such that $L_1(\alpha) \neq L_2(\alpha)$. In particular, let $\langle t_1, b\rangle$ and $\langle t_2, \bar{b}\rangle$ be the respective nodes of L_1 and L_2 reached by α. Let T be the set of terms that precede t_1 in L_1 or t_2 in L_2. Note that $ones(\alpha) \supseteq vars(t_1) \cup vars(t_2)$, and $zeroes(\alpha)$ must contain at least one variable from each term in T. This implies that an assignment α' in which $zeroes(\alpha')$ is a minimal subset of $zeroes(\alpha)$ which contains at least one variable from each term in T will also satisfy $L_1(\alpha') \neq L_2(\alpha')$. Such an assignment α' must be in A_{2k} since $|T| \leq 2k$. □

Property 3 implies that k-term monotone decision lists are polynomial-query learnable with $|A_{2k}| = n^{O(k)}$ membership queries alone. In the next section, we show that we can infer the target k-term monotone decision list from membership queries on A_{2k} in $n^{O(k^3)}$ time. The following property shows that it suffices to know the evaluation of a k-term monotone decision list L on A_{2k} in order to infer, in polynomial time, the evaluation of L on any assignment.

Property 4 *Let f be a boolean function representable as a k-term monotone decision list, α be any assignment, and $b \in \{0,1\}$. Then $f(\alpha) = b$ if and only if there exists some set $S \subseteq zeroes(\alpha)$ of at most k variables, such that for all assignments $\beta \in A_{2k}$ that satisfy $zeroes(\beta) \supseteq S$ and $ones(\beta) \supseteq ones(\alpha)$, $f(\beta) = b$.*

Proof

Assume that α reaches the i-th node $\langle t_i, b\rangle$ of some k-term monotone decision list representation L of f. For any pairwise disjoint sets Y and Z of variables, let $p(Y, Z)$ denote the partial assignment that sets to 0 all the variables in Y and to 1 the variables in Z.

To prove the "only if" part, let $S \subseteq zeroes(\alpha)$ be a set of at most $i-1$ variables whose negation falsify the first $i-1$ terms of L . Now consider the partial assignment $p(S, ones(\alpha))$. Clearly, all extensions of $p(S, ones(\alpha))$—in particular those that contain at most $2k$ 0's—reach $\langle t_i, b\rangle$, and are therefore classified as b.

Conversely, for any assignment α suppose there exists a set S as in the property. Suppose, for contradiction, that α reaches the node $\langle t_i, \bar{b}\rangle$ in some k-term monotone decision list representation L of f. Now consider the assignment β that sets to 0 the variables in S, sets to 0 one variable in $vars(t) - (vars(t_i) \cup ones(\alpha))$ for each term t in L before t_i that does not contain any variable of S, and sets to 1 the remaining variables. Clearly, β is in A_{2k} and is an extension of $p(S, ones(\alpha))$; however, β reaches $\langle t_i, \bar{b}\rangle$, which contradicts the supposition that every extension of $p(S, ones(\alpha))$ evaluates to b. □

The following corollary is immediate.

Corollary 5 *Let f be a boolean function representable as a k-term monotone decision list. Given $f(\beta)$ for each assignment $\beta \in A_{2k}$, we can determine $f(\alpha)$ for any assignment α in $n^{O(k)}$ time.*

Property 6 *Let L be a minimal k-term monotone decision list and let $\langle t, b\rangle$ be a node of L. Then there exists a set $S \subseteq A_k$ such that*

(i) $|S| \leq k$,

(ii) For each $\alpha \in S$, $L(\alpha) = b$, and

(iii) For each variable v, $v \in vars(t)$ if and only if $L(\alpha_{v\leftarrow 0}) = \bar{b}$, for some α in S.

Proof

The minimality of L guarantees that for any variable v in $vars(t)$ there exists an assignment α that justifies the necessity of v in t, i.e. α reaches $\langle t, b\rangle$ but $\alpha_{v\leftarrow 0}$ reaches some node $\langle t', \bar{b}\rangle$ after $\langle t, b\rangle$. (We say that t' validates v in t).

We now prove that if a term t' validates each variable in $V = \{v_1, v_2, ..., v_l\} \subseteq vars(t)$, then there exists a single assignment $\alpha \in A_k$ that justifies all the variables in V. To do this, we claim that every term of nodes prior to $\langle t', \bar{b}\rangle$ in L either (a) contains a superset of V or (b) contains some variable not in $vars(t) \cup vars(t')$. To prove the claim, suppose, to the contrary, that there is some term r of a node prior to $\langle t', \bar{b}\rangle$ such that r does not contain some variable v of V and $vars(r) \subset vars(t) \cup vars(t')$. Let α be a justifying assignment for v in t, such that $\beta = \alpha_{v \leftarrow 0}$ reaches $\langle t', \bar{b}\rangle$. Since $ones(\beta) \supseteq vars(r)$, β cannot reach any node beyond the one that contains r, a contradiction. It follows from the claim that the assignment α that sets to 0 one variable not in $vars(t) \cup vars(t')$ for each term that satisfies case (b), and sets to 1 the remaining variables is a justifying assignment for all variables in V, and has at most k 0's.

All the variables in $vars(t)$ must be validated by at most k terms. To finish the proof of the property, let S be a set of assignments which contains, for each node $\langle t', \bar{b}\rangle$ that follows $\langle t, b\rangle$ in L, one assignment $\alpha \in A_k$ that reaches $\langle t, b\rangle$ and that justifies all the variables in $vars(t)$ validated by t'. The set S then contains assignments that validate all the variables in $vars(t)$. Moreover, no variable not in $vars(t)$ will be justified by assignments in S since all the assignments in S reach $\langle t, b\rangle$. $\square$

Property 6 motivates the following definition. We say that $\langle t, b\rangle$ is a *candidate node* (in a k-term monotone decision list representation) of a boolean function f if there exists a set $S \subseteq A_k$ with at most k assignments, such that for each $\alpha \in S$, $f(\alpha) = b$ and such that $vars(t)$ is precisely the set of variables justified by assignments in S. Note that all candidate nodes of f can be found in $n^{O(k^2)}$ time. Moreover, the set of candidate nodes of a boolean function is a superset of all possible nodes $\langle t, b\rangle$ which can appear in a minimal k-term monotone decision list representation of f.

4 Learning k-term Monotone Decision Lists

We begin by showing that k-term monotone decision lists are properly learnable in $n^{O(k^3)}$ time, using $n^{O(k)}$ membership queries. Next, we show that any proper membership-query learning algorithm for this class must use $n^{\Omega(k)}$ queries, proving that our algorithm is reasonably tight. An easy consequence of our learning algorithm is that k-term monotone decision lists are "simple-PAC" learnable. Finally, we show that this class is not learnable with equivalence queries alone.

The algorithm below follows directly from Properties 3 and 6 and Corollary 5 of the previous section.

Algorithm Learn_k-term_monotone_decision_lists

1. Make membership queries on all assignments in A_{2k}.

2. Find the classification of the default.

3. Find all candidate nodes of the target function.

4. Construct all possible k-term monotone decision lists using candidate nodes and the default classification.

5. From the k-term monotone decision lists constructed in step (4), output the first one which agrees with the target function on the classification of the assignmnents in A_{2k}.

Correctness: In Step 1, we ask membership queries with all the assignments in A_{2k}. By using Corollary 5 to establish the evaluation of the target function on the all-zeroes assignment (or with a direct membership query), we decide the classification of the default node in the target function (Step 2).

Next, in Step 3, we use Property 6 to produce a set of $n^{O(k^2)}$ candidate nodes that are guaranteed to contain all the nodes of the target function represented as a k-term monotone decision list. Using all possible permutations of at most k nodes from the set of candidate nodes, we end up with a set of candidate k-term monotone decision lists (Step 4). Property 6 ensures that at least one of these decision lists will be equivalent to the target. By Property 3, testing the candidate decision lists for consistency with the target function over A_{2k} is enough to determine one which is equivalent to the target (Step 5).

Query Complexity: Only membership queries are used and then only in the first step. The number of queries is $|A_{2k}|$, which is at most $2kn^{2k} = n^{O(k)}$.

Time Complexity: The steps that take the most time are 4 and 5. Since Step 3 produces $n^{O(k^2)}$ candidate nodes, the number of candidate decision lists created in Step 5 is $n^{O(k^3)}$. This bound dominates the overall time, which remains $n^{O(k^3)}$.

The comments on correctness and complexity have effectively proved the following theorem.

Theorem 7 *The class of k-term monotone decision lists can be properly learned with $n^{O(k)}$ membership queries in $n^{O(k^3)}$ time.*

Note that the learning algorithm is non-adaptive, i.e., it uses membership queries only on the fixed set A_{2k}, independent of the target k-term monotone decision list. Therefore, it follows that k-term monotone decision lists are learnable in the "simple-PAC" model of Li and Vitányi [LV91].

Theorem 8 *Any membership-query learning algorithm for k-term monotone decision lists must use at least $\sum_{i=1}^{k-1} \binom{n}{i}$ queries.*

Proof

Let α be any assignment with at most $k-1$ zeroes. The singleton boolean function which is true precisely on the assignment α can be represented as a k-term monotone decision list. To construct such a representation, first create nodes $\langle v, 0\rangle$ for each variable v in $zeroes(\alpha)$, thus using at most $k-1$ nodes. Next, create a node $\langle t, 1\rangle$, where $vars(t) = ones(\alpha)$, and finally set the default to 0.

From the above, it follows that the number of k-term monotone decision lists that can represent singletons is at least $\sum_{i=0}^{k-1} \binom{n}{i}$. Using a standard adversary argument with singletons, we get the result. □

It is an easy exercise to show that 1-term monotone decision lists are learnable with equivalence queries alone. We now show that the class of k-term monotone decision lists has approximate fingerprints for $k \geq 2$, and therefore cannot be learned with equivalence queries alone. (For a definition of approximate fingerprints, and a proof that they are sufficient for non-learnability with equivalence queries alone, see [Ang90].) We start with a lemma.

Lemma 9 *Let L be a k-term monotone decision list, $k \geq 2$. Then there exists an assignment $\alpha = \alpha(L)$ such that either*

(a) α has at most $k-1$ zeroes, and $L(\alpha) = 0$, or

(b) α has at most n/k ones, and $L(\alpha) = 1$.

Proof

If L is not a monotone boolean function, or L has fewer than k terms, or if any two terms in L share a variable, then it is easy to construct an assignment that satisfies (a). Otherwise, all the k terms in L have distinct set of variables and L is a monotone Boolean function. Therefore, there must be a positive term in L with at most n/k variables. An assignment with ones exactly in the positions of the variables in such a term satisfies (b). □

Theorem 10 *The class of k-term monotone decision lists, $k \geq 2$, has approximate fingerprints.*

Proof

Let $T_{n,k}$ be the target class of k-term monotone decision lists in which all the terms are positive, the default is negative, and each term has exactly n/k variables that do not occur in any other term. To avoid floors and ceilings, assume that n is a multiple of k.

In order to prove the approximate fingerprint property, it suffices to show that the number of functions in $T_{n,k}$ that either classify an assignment of at most $k-1$ 0's as 0 or classify an assignment with at most n/k 1's as 1, is a superpolynomially small fraction of $|T_{n,k}|$.

Note that $|T_{n,k}| = \frac{n!}{[(n/k)!]^k k!}$. No function in $T_{n,k}$ classifies an assignment with at most $k-1$ zeroes as 0. Moreover, the fraction of the functions in $T_{n,k}$ that classify an assignment with at most n/k ones as 1 is at most

$$\frac{|T_{n-n/k,k-1}|}{|T_{n,k}|} = \frac{(n-n/k)!(n/k)!k}{n!} = \frac{k}{\binom{n}{n/k}}$$

which, after a routine use of Stirling's approximation, can be seen to be superpolynomially small in n for $k \geq 2$. □

Corollary 11 *The class of k-term monotone decision lists, $k \geq 2$, is not learnable with equivalence queries alone.*

Proof

Follows from Theorem 10 and Angluin's theorem [Ang90] on the approximate fingerprint property. □

5 General Monotone Decision Lists

General monotone decision lists are monotone term decision lists with no *a priori* bound on the number of nodes. As shown in section 2, this class can represent all boolean functions. Here we show that there is a polynomial time algorithm for testing the equivalence of two general decision lists. Our results on the learnability of this representation class are sparse and, unfortunately, all negative. First of all, it follows from Theorem 8 that general decision lists are not learnable with membership queries alone. We show below that equivalence queries alone also do not suffice for learnability.

Theorem 12 *There is an $O(n(p+q)pq)$ time algorithm that tests equivalence of two monotone decision lists on n variables, L_1 and L_2, with p and q nodes respectively. The algorithm also supplies an assignment that witnesses the inequivalence, if $L_1 \not\equiv L_2$.*

Proof

For any pair of terms t_1 and t_2, define the assignment $\beta(t_1, t_2)$ by $ones(\beta(t_1, t_2)) = vars(t_1) \cup vars(t_2)$. Clearly, if there exists a term t_1 in L_1 and a term t_2 in L_2 such that $L_1(\beta(t_1, t_2)) \neq L_2(\beta(t_1, t_2))$, then $L_1 \not\equiv L_2$. Conversely, if $L_1 \not\equiv L_2$, then there exists some assignment α such that $L_1(\alpha) \neq L_2(\alpha)$. Assume that α reaches nodes $\langle t_1, b\rangle$ in L_1 and $\langle t_2, \bar{b}\rangle$ in L_2. Consider the assignment $\beta = \beta(t_1, t_2)$. Now β reaches both $\langle t_1, b\rangle$ and $\langle t_2, \bar{b}\rangle$, and $L_1(\alpha) = L_1(\beta) \neq L_2(\beta) = L_2(\alpha)$.

From the above, it follows that testing L_1 and L_2 for equivalence is tantamount to checking if $L_1(\beta(t_1, t_2)) = L_2(\beta(t_1, t_2))$ for all pairs t_1 in L_1 and t_2 in L_2. There are $O(pq)$ pairs of terms to consider, and each evaluation takes $O(n(p+q))$ time. $\Box$

Note that the algorithm for testing equivalence can be used to test if a general monotone decision list is minimal—all one needs to do to decide if a particular variable in a term (or a particular node) is redundant is to delete the variable from the term (or the entire node) and test the resultant decision list for equivalence with the original.

We now show that monotone decision lists have approximate fingerprints. Note that this result does not follow immediately from the earlier proof of approximate fingerprints. We begin with a lemma.

Lemma 13 *Let L be a p-term monotone decision list on n variables. There is an assignment $\alpha = \alpha(L)$ such that either*

(a) $L(\alpha) = 0$ and α contains at most $\sqrt{n \ln p}$ 0's, or

(b) $L(\alpha) = 1$ and α contains at most $\sqrt{n \ln p}$ 1's.

Proof

Let $m = \sqrt{n \ln p}$. If the first term in L is negative, then clearly the all 1's assignment satisfies (a). If the default is positive, then the all 0's assignment satisfies (b). Otherwise, let $p' \leq p$ be the number of positive terms in L. If any such positive term, say t, has at most m variables, then the assignment α such that $ones(\alpha) = vars(t)$ satisfies (b). Finally, if all the positive terms of L have more than m variables, then we show the existence of an assignment α that satisfies (a) as follows.

Pick $r = m$ variables at random and set them to 0, setting the remaining variables to 1. The probability that such an assignment is accepted by some positive term of L is at most

$$(1 - m/n)^r p' \leq \left(1 - \sqrt{\frac{\ln p}{n}}\right)^r p < e^{-r\sqrt{\frac{\ln p}{n}}} p$$

For our choice of r, the last quantity in the above expression is 1. Therefore, there exists some set X of at most r variables such that the assignment α formed by setting all the variables in X to 0 and the remaining variables to 1 is not accepted by any positive term of L. Such an assignment satisfies (a). □

Theorem 14 *The class of general monotone decision lists has approximate fingerprints.*

Proof

Let T_n be the class of $q = \sqrt{n}$-term monotone decision lists in which all the terms are positive, the default is negative, and each term has exactly q variables which do not occur in any other term. (Assume that q is an integer in order to avoid floors and ceilings.) Note that this is the same target class as the one used in [AHK93] to show that read-once formulas cannot be identified with equivalence queries alone. The number of logically distinct Boolean functions in T_n is $\frac{n!}{(q!)^{q+1}}$.

Let $p = n^c$ for some fixed constant $c > 0$ and let L be any monotone decision list of p terms. Let $\alpha = \alpha(L)$ be the assignment whose existence is guaranteed by Lemma 13. To show the approximate fingerprint property, it suffices to show that the fraction of concepts in T_n that classify α as L does is superpolynomially small. As in Lemma 13, let $m = \sqrt{n \ln p}$. If $L(\alpha)$ is 1, the number of concepts in T_n that classify α as L does is at most $\binom{m}{q}\frac{(n-q)!}{(q!)^{q-1}(q-1)!}$. If $L(\alpha)$ is 0, the number of concepts in T_n that classify α as L does is at most $\binom{m}{q}\frac{(n-q)!}{[(q-1)!]^q q!}$). In either case, the fraction of concepts in T_n that classify α as L does is at most $\binom{m}{q}\frac{(n-q)! q^q}{n!}$, which (after plugging in the values of m and q) is less than $\frac{(\sqrt{nc \ln n})^{\sqrt{n}}}{(\sqrt{(n)})!} \cdot \frac{(\sqrt{n})^{\sqrt{n}}}{(n-\sqrt{n})^{\sqrt{n}}}$. This quantity can be seen to be superpolynomially small in n. □

Corollary 15 *The class of general monotone decision lists is not learnable with equivalence queries alone.*

Proof

Follows from Theorem 14 and Angluin's theorem [Ang90] on the approximate fingerprint property. □

6 Acknowledgments

The authors wish to thank Professors José L. Balcázar, Ricard Gavaldà and Jorge Castro for suggesting the problem and participating in the discussion.

References

[Ang88] D. Angluin. "Queries and Concept Learning." Machine Learning 2, 319–342, 1988.

[Ang90] D. Angluin. "Negative Results for Equivalence Queries." Machine Learning, vol. 5, 121–150, 1990.

[AB92] M. Anthony and N. Biggs. "Computational Learning Theory: An Introduction." Combridge University Press, 1992.

[AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. "Learning Read-Once Formulas with Queries." Journal of the ACM, vol. 40, number 1, 185–210, 1993.

[GK91] S. Goldman and M. Kearns. "On the Complexity of Teaching." Proceedings of the Fourth Annual Workshop on Computational Learning Theory, 303–314, 1991.

[HPRW96] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, D. Wilkins. "How Many Queries are Needed to Learn?" Journal of the ACM, September 1996 (to appear).

[LV91] M. Li and P. Vitányi. "Learning Simple Concepts under Simple Distributions." SIAM Journal of Computing 20, 911–935, 1991.

[Riv87] R. Rivest. "Learning Decision Lists." Machine Learning 2, 229–246, 1987.

[Sim94] H. U. Simon. "Learning Decision Lists and Trees with Equivalence Queries." Proceeding of the 2nd European Conference EUROCOLT, pp 322-336, 1995.

[Val84] L. Valiant. "A Theory of the Learnable." Communications of the ACM 27:11, 1134–1142, 1984.