Complexity Parameters for First Order Classes *

Marta Arias (marta@cs.columbia.edu)†
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA

Roni Khardon (roni@cs.tufts.edu)
Department of Computer Science
Tufts University
Medford, MA 02155, USA

Abstract. We study several complexity parameters for first order formulas and their suitability for first order learning models. We show that the standard notion of size is not captured by sets of parameters that are used in the literature and thus they cannot give a complete characterization in terms of learnability with polynomial resources. We then identify an alternative notion of size and a simple set of parameters that are useful for first order Horn Expressions. These parameters are the number of clauses in the expression, the maximum number of distinct terms in a clause, and the maximum number of literals in a clause. Matching lower bounds derived using the Vapnik Chervonenkis dimension complete the picture showing that these parameters are indeed crucial.

Keywords: Inductive Logic Programming, Learning Theory, First-Order Logic, VC-Dimension, Query Learning

1. Introduction

Since the introduction of Inductive Logic Programming (ILP), several theoretical investigations have contributed to characterizing the complexity of learning classes of expressions in first order logic (FOL). While learnability is usually defined using the size of the target concept as complexity measure, the complexity of algorithms and related lower bounds in the literature are usually quantified with other complexity measures. It is therefore not clear what these imply for the standard notions of polynomial learnability.

A comparison to propositional logic can highlight the difficulty. Work on learnability in propositional logic typically uses the number of propositions n and the size m of the target formula as complexity parameters; see Kearns and Vazirani (1994) for an overview. This is

* This work has been partly supported by NSF Grant IIS-0099446. A preliminary version of this paper appeared in the proceeding of the conference on Inductive Logic Programming 2003.

† Most of this work was done while M.A. was at Tufts University

reasonable as it allows a learning algorithm to use more time and other resources when examples (length n) or the formula being learned (length m) are larger. The situation in FOL differs from the propositional case since we do not have a fixed instance size n and it has proved difficult to get upper bounds directly in terms of the target size m. Moreover several parameters are inter-related so the value of one affects the other and a bound in terms of one implicitly depends on the other. It is therefore harder to interpret complexity results in this context.

This paper clarifies the situation by studying explicitly the relations between various notions of size used in the literature. We show that there is a discrepancy between parameters which are often used and the standard notion of size, and give a setting and set of parameters which are in some sense the right ones for first order learnability.

Previous work has provided both lower bounds and upper bounds on the resources required for learnability. Upper bounds are typically obtained by analyzing concrete algorithms. In doing so several authors have used standard parameters from first order logic, such as the number of clauses, the number of literals per clause etc. Others introduce special syntactic parameters such as depth and determinacy or restrict the structure of clauses or background knowledge in their analysis (Muggleton and Feng, 1992, Džeroski et al., 1992) and (Kietz and Džeroski, 1994, Cohen, 1995, Arimura, 1997). See results of Reddy and Tadepalli, Horváth and Turán (1997, 2001) and also ours in (Arias and Khardon, 2002).

Lower bounds were derived using the notion of Vapnik-Chervonenkis (VC) dimension. VC based bounds apply in several models of learnability including the PAC model (Ehrenfeucht et al., 1989) and the model of exact learning with queries (Maass and Turán, 1992). Several lower bound results for first order learnability ignore some parameters and prove exponential or infinite growth w.r.t other parameters (Arimura, 1997, Khardon, 1999a, Maass and Turán, 1995). Work in (Arimura, 1997, Khardon, 1999a) shows that the complexity may be exponential in the arity of predicates. However, both papers do not highlight the fact that the number of literals in the expressions being learned is of the same order (also exponential in arity). Maass and Turán (1995) show that the VC dimension is infinite with a single binary predicate but does not highlight the fact that these cases allow for an infinite number of constants whose encoding is not accounted for in the size of expressions[1]. In fact, any such lower bound going beyond

[1] The case here is similar to learning classes with real valued parameters where each number is charged one unit of complexity, but nonetheless the VC dimension

the size of expressions must have a hidden unaccounted aspect: since the VC dimension is bounded by the logarithm of the class size, for discrete cases the lower bounds cannot be larger than the size of the learned expressions assuming a reasonable encoding scheme.

Therefore, the question is what constitutes a good set of parameters for first order learnability. Such a set should capture the size and avoid the confusion from inter-related parameter sizes. To answer this question we consider a setting where the parameters of the FOL signature (number of predicates, constants, function symbols, arity) are fixed in advance and are therefore numerical constants. The concept class is defined by the other parameters controlling the expressions (number of variables, terms, clauses etc).

We start our investigation by defining when two sets of parameters are "related" so that polynomial learnability transfers from one set to the other. Using this we show that there is no simple answer (set of parameters) if the standard notion of formula size is used: the standard notion of size for FOL is not polynomially bounded by the natural parameters of FOL. On the other hand if we use a more compact representation, where a repeated term is counted only once, then one can derive a polynomial bound for the total size. The crucial parameters turn out to be c, l, and t where c is the number of clauses in the Horn expression, l is the largest number of literals in a single clause, and t is the maximal number of distinct terms and subterms in a single clause. With this in mind we prove that the VC dimension is $\tilde{\Theta}(cl + ct)$ (where $\tilde{\Theta}()$ hides logarithmic factors in the standard $\Theta()$ notation). This holds for ILP both in the model of learning from interpretations (De Raedt and Džeroski, 1994) and for learning from entailment (Frazier and Pitt, 1993). Therefore, our results identify a natural separation of the parameters to fixed ones relating to the signature and variable ones relating to the construction of expressions. With this we give a new notion of size and corresponding set of parameters that capture it, and characterize the VC dimension which is polynomially related to these parameters.

The rest of the paper is organized as follows. The next section gives some technical preliminaries. Section 3 defines complexity measures for first order logic. Section 4 develops the notion of polynomially related sets of parameters and Section 5 applies this notion to first order logic.

---

of various concept classes is bounded. The negative result mentioned shows that this does not hold for first order logic except in very restricted cases. The work in (Maass and Turán, 1995, Grohe and Turán, 2002) identifies syntactic restrictions on formulas, examples, and background knowledge that give bounded VC dimension in this setting.

Section 6 develops the results on the VC dimension. The concluding section gives further discussion of the results and directions for future work.

2. Preliminaries

We assume familiarity with first order logic (Lloyd, 1987). The following gives the basic definitions for concept classes and learnability in this context.

A signature determines the variables, function symbols and predicate symbols (with their respective arity) over which formulas are built. Function symbols of arity zero are often called *constants*. A *term* is built bottom up from constants and variables by applying function symbols of the appropriate arity; if $t_1, .., t_a$ are terms and f is a function symbol of arity a, then $f(t_1, .., t_a)$ is a term. An *atom* is a predicate applied to a tuple of terms of the appropriate length. A *literal* is an atom or the negation of an atom.

We consider universally quantified first order Horn expressions. A clause is a disjunction of literals where all variables in the clause are (implicitly) universally quantified. A Horn clause has at most one positive literal. A Horn expression is a conjunction of Horn clauses. Note that any clause can be written as $C = (\wedge_{n \in \mathrm{Neg}} n) \rightarrow (\vee_{p \in \mathrm{Pos}} p)$ where Neg and Pos are the sets of atoms that appear in negative and positive literals of C respectively. When doing so we will refer to $(\wedge_{n \in \mathrm{Neg}} n)$ as the *antecedent* of C and to $(\vee_{p \in \mathrm{Pos}} p)$ as the *consequent* of C. A clause is *range restricted* if every term or sub-term that appears in its consequent also appears in its antecedent. A clause is *constrained* if every term or sub-term that appears in its antecedent also appears in its consequent.

Example 1. Consider a signature with a predicate p of arity 2, a constant b, a function symbol f of arity 1 and a variable x. When discussing concrete signatures, we will use the notation p/a to denote a predicate symbol with its arity (or a function symbol with its arity). Thus this signature has function symbols $b/0$ and $f/1$, and predicate $p/2$. The clause $C_1 = p(x, b) \wedge p(f(b), x) \rightarrow p(f(x), f(b))$ is constrained but not range restricted, the clause $C_2 = p(x, b) \wedge p(f(b), f(x)) \rightarrow p(f(x), b)$ is range restricted but not constrained, and the clause $C_3 = p(x, b) \wedge p(f(b), x) \rightarrow p(f(x), b)$ is neither range restricted nor constrained.

We often use sets of literals to denote clauses and set of clauses to denote their conjunction. Hence, when we write $\{L_1, .., L_l\}$ where L_i

are literals we mean $L_1 \vee .. \vee L_l$. When we write $\{C_1, .., C_c\}$ where C_i are clauses we mean $C_1 \wedge .. \wedge C_c$. The intension is clear from the context and helps simplify the presentation.

Given a signature $\mathcal{S}$, an $\mathcal{S}$-*interpretation* (sometimes called $\mathcal{S}$-model or $\mathcal{S}$-structure) assigns a "meaning" to symbols in the language in the following way. The interpretation includes a domain D whose elements are referred to as objects. Each function symbol is associated with a mapping from tuples of domain objects of appropriate arity to domain objects. Each predicate symbol is associated with a subset of tuples of the appropriate arity on which it is true; this is known as the *extension* of the predicate. We refer to the set of possible interpretations over $\mathcal{S}$ as $Int(\mathcal{S})$.

A formula is given a truth value on an interpretation in a natural way, by first extending the function mapping to a *term assignment* associating an object to each term and then evaluating the resulting atoms and logical connectives based on the extension of predicates in the interpretation.

If an expression T evaluates to true on interpretation I then we say that I satisfies T and denote this by $I \models T$. In this case, we also say that I is a model of T. If T evaluates to false under I, then we say that I falsifies T and denote this by $I \not\models T$. A first order expression T_1 *entails* (logically implies) another expression T_2, denoted $T_1 \models T_2$, if every model of T_1 is also a model of T_2. Two expressions T_1, T_2 are *logically equivalent*, denoted $T_1 \equiv T_2$, iff $T_1 \models T_2$ and $T_2 \models T_1$.

There exist several settings in ILP defining what constitute concepts and examples (Muggleton and De Raedt, 1994, De Raedt, 1997). We mainly consider the framework of *learning from interpretations* (De Raedt and Džeroski, 1994) where examples given to the learner are interpretations. Concepts are represented by first order formulas. A concept is associated with a set of interpretations for which it is true. Thus the concept represented by a formula ψ is given by the set of interpretations $\{M \mid M \models \psi \text{ and } M \in Int(\mathcal{S})\}$. A *concept class* is a set of concepts usually described by a family of formulas representing the concepts.

We also consider *learning from entailment* (Frazier and Pitt, 1993) where examples are clauses in the language. To minimize confusion we defer definition and discussion of this setting to Section 6.2.

The *size* of a concept is the size of the smallest formula representing it. If no such formula exists, then the concept's size is infinite. Usually the size of a formula is its *string length* but other notions of size are also possible and we discuss these in detail below. Given a concept class

$\mathcal{C}$ and a notion of size, we define $\mathcal{C}^{\leq m}$ as the concepts in $\mathcal{C}$ of size at most m. Naturally, $\mathcal{C} = \cup_{m \geq 1} \mathcal{C}^{\leq m}$.

While our discussion and results are largely independent of the learning model it will be useful to have a model in mind. We briefly review the model of *exact learning with equivalence queries and membership queries* (Angluin, 1988) in the context of learning from interpretations. Before the learning process starts, a concept is fixed among all the concepts in the concept class. We refer to this concept as *target* concept. The goal of the learner is to output an expression that represents the target concept. The learner (the learning algorithm) has access to an equivalence oracle and a membership oracle that provide information about the target concept. In an equivalence query, the learner presents a hypothesis in the form of a first order formula and the oracle answers `Yes` if it is a representation of the target concept. Otherwise, it answers `No` and provides a counterexample, that is, an example (interpretation) where target and hypothesis disagree. In a membership query, the learner presents an example (interpretation) and the oracle answers `Yes` or `No` depending on whether the example presented is a member of the target concept. We assume that the learner is given the signature $\mathcal{S}$ as input.

The following definitions are due to Hellerstein et al. (1996):

Definition 1. The *query complexity* of a learning algorithm $\mathcal{A}$ at any stage in a run is the sum of the sizes of the (i) inputs to equivalence queries, and (ii) inputs to membership queries made up to that stage.

Notice that the definition of query complexity uses two different notions of size, one capturing the complexity of the hypotheses, the other capturing the complexity of the examples. The following definition captures learnability with respect to query complexity (ignoring time complexity):

Definition 2. An algorithm $\mathcal{A}$ is a *polynomial query learning algorithm* for a concept class $\mathcal{C}$ if there exists a polynomial $r(\cdot, \cdot)$ such that, for any positive integer m, and for any unknown target concept $c \in \mathcal{C}^{\leq m}$:

(i) $\mathcal{A}$ uses membership queries and equivalence queries of the form $EQ(h)$ where h represents a concept in $\mathcal{C}$

(ii) $\mathcal{A}$ eventually halts and outputs a string h representing the target concept c, and

(iii) at any stage, if n is the size of the longest counterexample received so far in response to an equivalence query, the query complexity of $\mathcal{A}$ at that stage does not exceed $r(n, m)$.

3. Complexity parameters for first order logic

In this section we present several ways of quantifying the representation complexity of our two first order constructs of interest: interpretations and expressions.

3.1. Complexity of first order expressions

We start with a description of possible ways of quantifying the representation or description complexity of first order expressions. We illustrate these using the following first order expression E:

$$(\forall X\ add(zero, X, X)) \wedge$$
$$(\forall X\ \forall Y\ \forall Z\ add(X, Y, Z) \rightarrow add(succ(X), Y, succ(Z)))$$

First, we introduce parameters that quantify the complexity of a given signature. Since our model assumes that the signature is fixed and given in advance, these parameters are considered as numerical constants when reporting results.

***NPredicates*$(\cdot)$:** counts the number of distinct predicate symbols appearing in the input expression. In the example, $NPredicates(E) = 1$ corresponding to $\{add/3\}$. We denote this parameter by p.

***NFunctions*$(\cdot)$:** counts the number of distinct function symbols appearing in the input expression. In the example, $NFunctions(E) = 2$ corresponding to the function symbols $zero/0$ and $succ/1$. We denote this parameter by f.

***Arity*$(\cdot)$:** the largest arity of any predicate or function symbol appearing in the input expression. In the example, $Arity(E) = 3$ corresponding to the predicate $add/3$. We denote this parameter by a.

Next, we introduce parameters that quantify the global complexity of a given expression.

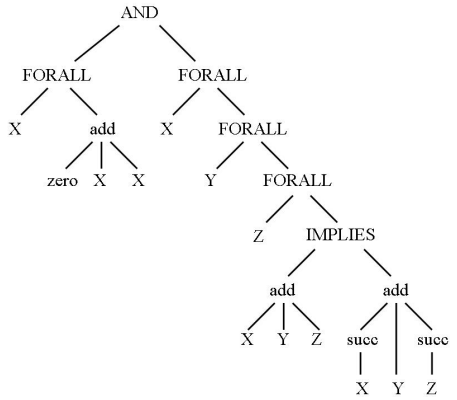


Figure 1. Tree representing the expression E.

***StringSize*$(\cdot)$:** as its name suggests, *StringSize* counts the number of syntactic symbols used to write down the input expression, ignoring spaces. Predicate and function symbols whose name is longer than one letter contribute just 1. In our example, $\mathit{StringSize}(E) = 44$: the first clause contributes 12 and the second clause contributes 31, and we have to count the connective $\wedge$ as well.

***TreeSize*$(\cdot)$:** this size measure counts the number of nodes in a tree constructed recursively in the following manner. If the expression is a quantified expression, then put the quantifier in the root (labeled with the quantifier, `FORALL` or `EXISTS`), the quantified variable as its left child and the rest of the expression as the right child. If the expression is a conjunct, then add as children to the root (labeled with `AND`) all its conjuncts. Disjuncts are treated analogously, having `OR` as the root and the disjuncts as children. For implications the root is labeled with `IMPLIES` and the left child is the antecedent and the right child the consequent. With a negation the node is labeled with `NOT` and the only child is the rest of the expression. For atomic formulas, the root is labeled with the predicate symbol and the children are its arguments. If the expression is a variable, then the root is a leaf labeled with the variable name. For functional terms, the root is the outermost function symbol and the children are its arguments. In our example, $\mathit{TreeSize}(E) = 24$; its associated tree is depicted in Figure 1.

***DAGSize*$(\cdot)$:** counts the number of nodes in a DAG constructed by unifying identical subtrees that correspond to terms in the tree constructed as explained above. We assume that expressions are *stan-*

dardized apart, that is, we avoid re-use of variable names that belong to scopes of different quantifiers. This converts our expression E into the equivalent E':

$$(\forall X'\ add(zero, X', X')) \wedge$$
$$(\forall X\ \forall Y\ \forall Z\ add(X, Y, Z) \rightarrow add(succ(X), Y, succ(Z)))$$

In the example, the only repetition of terms are of variables X, Y, Z and X' which appear 3 times each. We save $4 \times (3 - 1) = 8$, hence $DAGSize(E') = TreeSize(E) - 8 = 16$.

We next consider natural parameters of first order representations. Notice that some of these parameters apply only to clause-based expressions such as Horn expressions.

Depth*(·):* the maximum depth of any functional term appearing in the input expression. In the example, $Depth(E) = 2$ corresponding to the deepest term $succ(X)$ (or $succ(Z)$). We denote this parameter by d.

NTerms*(·):* counts the maximum number of distinct terms (including sub-terms) in any clause of the input CNF expression. In the example, $NTerms(E) = 5$, corresponding to term set in the second clause $\{X, Y, Z, succ(X), succ(Z)\}$. We denote this parameter by t.

NVariables*(·):* counts the maximum number of distinct variables appearing in any clause of the input CNF expression. In the example, $NVariables(E) = 3$, corresponding to variable set in the second clause $\{X, Y, Z\}$. We denote this parameter by v.

NLiterals*(·):* counts the maximum number of literals in any clause of the input CNF expression. In the example, $NLiterals(E) = 2$ from the second clause. We denote this parameter by l.

NClauses*(·):* counts the number of clauses in the input CNF expression. In our example, $NClauses(E) = 2$. We denote this parameter by c.

3.2. Complexity of first order interpretations

The complexity of a first order interpretation can be captured by a single parameter: the number of objects in its domain. The remaining

constituents of an interpretation (function mappings and extensions) are of polynomial size w.r.t. the number of domain objects if the arity, the number of function symbols and the number of predicate symbols are considered constant.

4. Relating parameters to "Size"

While learnability is usually defined in terms of the notion of size, it may be useful to provide bounds using other measures (as various authors have done). We therefore need to extend the definitions of query complexity and learnability to refer to a set of parameters. This is done in a natural way so that query complexity measures each of the parameters, and learnability requires a polynomial bound in every parameter. Thus, in Definition 1, instead of summing the sizes of hypotheses and examples, we sum each parameter separately and have a complexity measure per parameter. Similarly, in Definition 2 we need to replace n, m with lists of complexity parameters and replace $r(\cdot, \cdot)$ with a list of bounds, one for each parameter. This is done in Theorem 2 below. However, this is not sufficient. We must also identify when such a replacement preserves polynomial learnability. For this we define:

Definition 3. Let $\mathcal{C}$ be a class of first order expressions. Let k and j be positive integers. Let $\overline{C} = \{C_1, .., C_k\}$ be a list of complexity measures on expressions in $\mathcal{C}$, and let $\overline{D} = \{D_1, .., D_j\}$ be an alternative list of complexity measures on expressions in $\mathcal{C}$. We say that $\overline{C}$ and $\overline{D}$ are *polynomially related* w.r.t. $\mathcal{C}$ if there exist polynomials $p_1, .., p_k$ of arity j and polynomials $q_1, .., q_j$ of arity k such that for every $E \in \mathcal{C}$:

(i) for all $i = 1, .., k$: $C_i(E) \leq p_i(D_1(E), .., D_j(E))$, and

(ii) for all $i = 1, .., j$: $D_i(E) \leq q_i(C_1(E), .., C_k(E))$.

The next lemma follows directly from the definition of polynomial relation:

LEMMA 1. *The polynomial relation between sets of complexity measures is reflexive, transitive, and symmetric.*

The next theorem shows that this notion of polynomial relation among complexity measures captures exactly the situations in which one can substitute the related complexity measures without changing the learning model.

THEOREM 2. *Let $\mathcal{C}$ be a class of first order expressions. Let $C_1, .., C_k$ be a set of complexity measures that is polynomially related to Size w.r.t. the class $\mathcal{C}$, where Size is some notion of size for the expressions in $\mathcal{C}$. Let $p_1(\cdot), .., p_k(\cdot)$ and $q(\cdot, .., \cdot)$ be the polynomials witnessing their polynomial relation. Similarly, let $D_1, .., D_{k'}$ be a set of complexity measures that is polynomially related to $Size'$ w.r.t. the class $\mathcal{E}$, where $\mathcal{E}$ is a representation class for the examples, and $Size'$ is some notion of size for the example representations in $\mathcal{E}$. Let $p'_1(\cdot), .., p'_{k'}(\cdot)$ and $q'(\cdot, .., \cdot)$ be the polynomials witnessing their polynomial relation.*

Suppose that $\mathcal{A}$ is a learning algorithm for class $\mathcal{C}$ with query complexity (w.r.t. $C_1, .., C_k$ and $D_1, .., D_{k'}$) bounded by the polynomials $s_i(c_1, .., c_k, d_1, .., d_{k'})$ for $i = 1, .., k$, and $s'_j(c_1, .., c_k, d_1, .., d_{k'})$ for $j = 1, .., k'$, where $c_1, .., c_k$ bound the complexity of the target concept (w.r.t. $C_1, .., C_k$), and $d_1, .., d_{k'}$ bound the complexity of the counterexamples received (w.r.t. $D_1, .., D_{k'}$). Then, $\mathcal{A}$ is a polynomial query learning algorithm for $\mathcal{C}$.

Proof. Notice that items (i) and (ii) of Definition 2 on learnability hold trivially since we have assumed that $\mathcal{A}$ is a learning algorithm for $\mathcal{C}$ working in the same model. We show that item (iii) holds. Namely, there is a polynomial $r(\cdot, \cdot)$ s.t. at any stage, if n is the size of the longest counterexample received so far in response to an equivalence query, the query complexity of $\mathcal{A}$ at that stage does not exceed $r(n, m)$.

In the following, $\overline{f_{1..k}(args)}$ stands for $f_1(args), .., f_k(args)$. We define

$$r(n,m) = q(\overline{s_{1..k}(\overline{p_{1..k}(m)}, \overline{p'_{1..k'}(n)})}) + q'(\overline{s'_{1..k'}(\overline{p_{1..k}(m)}, \overline{p'_{1..k'}(n)})}).$$

Observe that all the functions $s_1, .., s_k, s'_1, .., s'_{k'}$, $p_1, .., p_k, p'_1, .., p'_{k'}$ and q, q' are polynomials and hence r is a polynomial. It is left to show that r bounds the query complexity for $\mathcal{A}$.

Notice that $c \in \mathcal{C}^{\leq m}$ implies $c \in \mathcal{C}^{\leq \overline{p_{1..k}(m)}}$ since $p_1(m), .., p_k(m)$ bound the complexity measures in $C_1, .., C_k$. By assumption, the query complexity (w.r.t. parameters $C_1, .., C_k$) of $\mathcal{A}$ is bounded by

$$\overline{s_{1..k}(\overline{p_{1..k}(m)}, \overline{p'_{1..k'}(n)})}$$

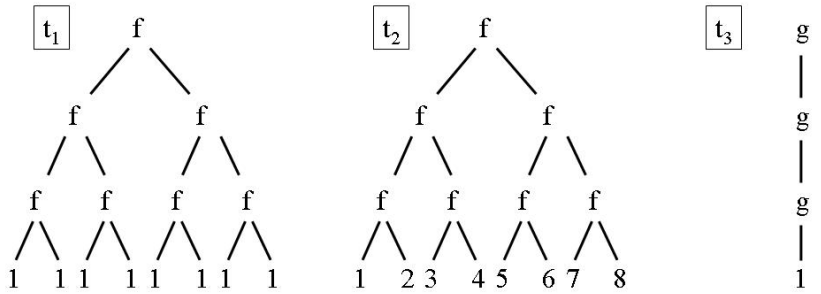
and by

$$\overline{s'_{1..k'}(\overline{p_{1..k}(m)}, \overline{p'_{1..k'}(n)})}.$$

Hence, the query complexity of $\mathcal{A}$ (w.r.t. $Size$ and $Size'$) is bounded by

$$q(\overline{s_{1..k}(\overline{p_{1..k}(m)}, \overline{p'_{1..k'}(n)})}) + q'(\overline{s'_{1..k'}(\overline{p_{1..k}(m)}, \overline{p'_{1..k'}(n)})}).$$

□



$TreeSize(t_1) = \Theta(2^d)$	$TreeSize(t_2) = \Theta(2^d)$	$TreeSize(t_3) = \Theta(d)$
$DAGSize(t_1) = \Theta(d)$	$DAGSize(t_2) = \Theta(2^d)$	$DAGSize(t_3) = \Theta(d)$

Figure 2. Three terms with different combinations of (asymptotic) *TreeSize* and *DAGSize*.

Remark 1. Note that we require polynomial bounds in both directions to guarantee learnability. This is needed for learning with queries and for proper PAC learnability (where hypothesis class is the same as concept class), whereas a one sided bound suffices for PAC predictability.

It is useful to highlight what can go wrong if this does not hold. In Figure 2 we can see three terms: t_1 has *TreeSize* exponential in the depth while its *DAGSize* is just linear (further discussion of t_1 is given in Theorem 4 below); t_2 has both *TreeSize* and *DAGSize* exponential in the depth; finally t_3 has both *TreeSize* and *DAGSize* linear in the depth.

Now, if one has an algorithm that learns w.r.t. *TreeSize* then when learning an expression including t_1 the algorithm is allowed to include t_2 in a query but this is not possible for learning w.r.t. *DAGSize* since t_1 is just polynomial in the depth whereas t_2 is exponential. On the other hand, if one has an algorithm that learns w.r.t. *DAGSize* then

when learning an expression including t_3 the algorithm can use t_1 in its query. If we try to use this algorithm to learn w.r.t. *TreeSize* this query is too large.

5. Relating complexity measures for first order logic

The previous two sections give complexity parameters and a tool to relate them. We next investigate which subsets of the alternative complexity measures are polynomially related to our notions of size.

Definition 4. Let $\mathcal{P}$ be the set of alternative complexity parameters $\{NTerms, NVariables, Depth, NLiterals, NPredicates, NFunctions, Arity, NClauses\}$.

It is not hard to see that the tree representation can be padded with extra commas and parentheses and therefore:

LEMMA 3. *StringSize is polynomially related to TreeSize.*

As a result, while we typically think of *StringSize* as defining learnability, we can discuss complexity with respect to *TreeSize* without loss of clarity. The question is whether we can find a combination of the alternative parameters in $\mathcal{P}$ that is polynomially related to *TreeSize*. Suppose that E is a first order Horn expression s.t.

$$\begin{array}{lll} NTerms(E) = t & NVariables(E) = v & Depth(E) = d \\ NLiterals(E) = l & NPredicates(E) = p & NFunctions(E) = f \\ Arity(E) = a & NClauses(E) = c & \end{array}$$

Observe that any term appearing in E has (tree) size at most $O(a^d)$. Hence, any atomic formula has (tree) size at most $1 + O(a^{d+1}) = O(a^{d+1})$ (1 for the predicate symbol, a^{d+1} for the arguments). Hence, the tree size of any Horn clause is bounded by $1 + 2v + lO(a^{d+1}) = O(v + la^{d+1})$ (1 for the implication symbol in the clause, $2v$ for the quantifiers and quantified variables, and $O(a^{d+1})$ for each atom in the clause). Therefore:

$$TreeSize(E) = O(cv + cla^{d+1}) = O(cla^{d+1}),$$

where the last equality follows since the number of "slots" for variables in each clause is bounded above by la^{d+1}, and hence $v \leq la^{d+1}$.

On the other hand, it is clear that all the parameters above are bounded by *TreeSize*(E). The next theorem shows that the converse does not hold:

THEOREM 4. *TreeSize is not polynomially bounded by any subset of parameters in $\mathcal{P}$ for classes over signatures with at least one constant and one function symbol of arity at least 2.*

Proof. We give an expression E such that its *TreeSize* is exponential in *NTerms*. Let $E = p(t_1)$, where t_1 is a complete tree of degree a with internal nodes labeled with function symbol f and leaves labeled with constant 1:

$$p(\overbrace{f(..f(f(f}^{d \text{ times}}(\overbrace{1,..,1}^{a \text{ times}}),..,f(1,..,1)),..,f(f(1,..,1),..,f(1,..,1)))..))$$

The term t_1 is represented in Figure 2 for $a = 2$ and $d = 3$. The complexity measures for E are:

$NTerms(E) = d$	$NVariables(E) = 0$	$Depth(E) = d$
$NLiterals(E) = 1$	$NPredicates(E) = 1$	$NFunctions(E) = 2$
$Arity(E) = a$	$NClauses(E) = 1$	$TreeSize(E) = \Theta(a^d)$

Hence no polynomial combination of the available complexity measures upper bounds *TreeSize*(E). □

This is a surprising fact that has not been noticed in previous work working with these parameters. No polynomial combination of the parameters above can replace *TreeSize*.

PROPOSITION 5. *If there are no function symbols of arity greater than 1, then the set $\{NClauses, NLiterals, Depth\}$ is polynomially related to TreeSize.*

Proof. This follows from the fact that in this case $TreeSize = O(cv + clad) = O(clad)$. □

On the other hand, exponential lower bounds in terms of arity have been derived when ignoring *NLiterals*. These essentially reflect the following fact:

PROPOSITION 6. *If the number of literals is ignored then TreeSize and DAGSize are not polynomially bounded by Arity.*

Proof. Let p be a predicate of arity a. Let $\{1,..,t\}$ be a set of t distinct terms built e.g. by one constant and one unary function. Let P be the set of all different $p()$ atoms built from these terms; $|P| = t^a$. Let $\hat{p}$ be a particular element in P. Let E be the expression $E = P \backslash \{\hat{p}\} \rightarrow \hat{p}$. The complexity of E is given by:

$NTerms(E) = t$	$NVariables(E) = 0$	$Depth(E) = t$
$NLiterals(E) = t^a$	$NPredicates(E) = 1$	$NFunctions(E) = 2$
$Arity(E) = a$	$NClauses(E) = 1$	
$TreeSize(E) = \Omega(t^a)$	$DAGSize(E) = \Omega(t^a)$	

Hence, the tree size is exponential in the arity and is not polynomially bounded by other parameters when l is ignored. □

As a result, a linear lower bound in terms of size can be seen as an exponential lower bound in terms of arity.

Like in the case of *TreeSize*, *DAGSize* also gives an upper bound for all the alternative parameters in $\mathcal{P}$. But, unlike *TreeSize*, the relation in the other direction is polynomial for *DAGSize*. Notice that a DAG encodes terms in a smarter way, since multiple occurrences of a term are only counted once. Hence, t terms in a clause contribute only $\Theta(t)$ to the *DAGSize*. Each atomic formula contributes only 1 since its arguments have already been counted (encoded with the terms). Hence, every clause has size at most $O(v + t + l) = O(t + l)$ and

$$c + l + t \leq DAGSize(E) = O(ct + cl).$$

We therefore have:

THEOREM 7. *The set of parameters* $\{NTerms, NLiterals, NClauses\}$ *is polynomially related to DAGSize w.r.t. the class of first order Horn expressions.*

Notice that the theorem is true for any values of the other parameters. Proposition 6 shows that *DAGSize* can be exponential in arity but in such a case Theorem 7 guarantees that one of c, l, t must be large as well. It is also interesting to note that several results on learning with queries give upper bounds in terms of t^a and other parameters: Arimura (1997), Reddy and Tadepalli (1998), Krishna Rao and Sattar (1998), and Arias and Khardon (2002). While $l \leq pt^a$ these bounds do not directly relate to *DAGSize* or *TreeSize*.

6. The VC dimension of first order Horn expressions

This section characterizes the Vapnik-Chervonenkis dimension (VC dimension) of first order Horn expressions. It is known that the VC dimension provides tight bounds on the number of examples needed for PAC learning; see Vapnik and Chervonenkis (1971) and Blumer et al., Ehrenfeucht et al. (1989, 1989). It also provides a lower bound for the number of equivalence and membership queries needed for exact learning (Maass and Turán, 1992).

We start with the necessary definitions.

Definition 5. Let $\mathcal{I}$ be a set, $\mathcal{H} \subseteq 2^{\mathcal{I}}$, and $\mathcal{S} \subseteq \mathcal{I}$. Then $\Pi_{\mathcal{H}}(\mathcal{S})$ is the set $\{h \cap \mathcal{S} \mid h \in \mathcal{H}\}$, i.e. the set of subsets of $\mathcal{S}$ that can be obtained by intersection with elements of $\mathcal{H}$. If $|\Pi_{\mathcal{H}}(\mathcal{S})| = 2^{|\mathcal{S}|}$, then we say that $\mathcal{H}$ shatters $\mathcal{S}$. Finally, $VCDim(\mathcal{H})$ is the size of the largest set shattered by $\mathcal{H}$ (or ∞ if arbitrary large sets are shattered).

From the definition above it follows that for finite classes $\mathcal{T}$, we have $VCDim(\mathcal{T}) \leq \log |\mathcal{T}|$. Hence, in order to obtain an upper bound for the VC dimension of first order Horn expressions, we compute first how many concepts there are in the class $\mathcal{H}^{\leq c,t,l}$ of first order Horn expressions with at most c clauses, at most t terms per clause, and at most l literals per clause.

We show how to encode each concept in $\mathcal{H}^{\leq c,t,l}$ with a binary alphabet. In order to represent terms or literals we need to refer to function and predicate symbols; assume there are p predicates and f function symbols (of arity at most a) that we can refer to by using $\log p$ and $\log f$ bits, respectively. We assume that a, p and f are constant values, hence a, $\log p$ and $\log f$ are just $O(1)$. To encode a set of t distinct terms, we list them in a table with t rows, where each row is of size at most $\log f + a \log t$ ($\log f$ are the bits used to encode the head of the term, and $a \log t$ are the number of bits used to encode its arguments). This results in $t(\log f + a \log t) = O(t \log t)$ bits for the term table. Now, we just need $\log t$ bits to refer to terms in the expressions (the indices of the terms in the term table). To encode one clause, we use a table with at most l rows, each being of size at most $1 + \log p + a \log t$ (1 is to indicate whether the literal is negated or not). This results in $l(1 + \log p + a \log t) = O(l \log t)$ bits for the clause table. Hence, to encode a single clause we need $O(l \log t + t \log t)$ bits. To encode c clauses, we need to have a term and a clause table for each clause, and hence $O(cl \log t + ct \log t)$ bits are sufficient.

With $B = O(cl \log t + ct \log t)$ bits we can represent a maximum of 2^B different concepts. Note that this fact is valid regardless of representation of the examples, thus:

THEOREM 8. *$VCDim(\mathcal{H}^{\leq c,t,l}) = O(cl \log t + ct \log t)$. This bound holds for learning from interpretations and for learning from entailment.*

In the rest of this section we show that $VCDim(\mathcal{H}^{\leq c,t,l}) = \Omega(cl + ct)$. The two learning models are handled separately in the next two subsections.

6.1. Learning from interpretations

In the following sequence of lemmas we construct sets of interpretations of appropriate cardinality, and show how to shatter them by giving families of first order Horn expressions separating each possible dichotomy of the interpretation sets. We make extensive use of the interpretations' function mappings to ensure that terms evaluate to appropriate values so that separation is guaranteed.

LEMMA 9. *There exists a set of c interpretations of size $\Theta(\log c)$ that can be shattered using first order Horn expressions bounded by $NClauses \leq c$, $NTerms \leq \log c + 3$, $NLiterals = 2$, $NVariables = 0$, $Depth = \log c$, $Arity = 2$, $NFunctions = 4$ and $NPredicates = 2$.*

Proof. We construct a set of c different terms using a function f of arity 2 and three constants 1, 2 and 3 and by forming ground terms of depth $\log c$ in the following manner:

$$\hat{\mathcal{T}} = \{f(a_1, f(a_2, f(a_3, f(...f(a_{\log c}, 3)...)) \mid a_i \in \{1, 2\} \text{ for } 1 \leq i \leq \log c\}$$

Notice that there are exactly $2^{\log c} = c$ such terms (Figure 3 shows all of these terms when $c = 8$). Moreover, every term in $\hat{\mathcal{T}}$ contains at most $\log c + 3$ distinct subterms.

We define $\mathcal{I}$, the set of interpretations to be shattered, by giving an interpretation per element $\hat{t}$ of $\hat{\mathcal{T}}$. Hence, $|\mathcal{I}| = |\hat{\mathcal{T}}| = c$. The domain of the interpretation $I_{\hat{t}}$, consists of the $\Theta(\log c)$ objects corresponding to the subterms appearing in $\hat{t}$ (including itself) and a distinguished object $*$. The function mapping for f is defined to follow the functional structure of the distinguished term $\hat{t}$, and remaining entries are mapped to $*$. Notice that any term $t' \in \hat{\mathcal{T}}$ s.t. $\hat{t} \neq t'$ is mapped to the special object $*$ under the interpretation $I_{\hat{t}}$. The signature includes two predicates $P/1$ and $F/0$; the extension of $I_{\hat{t}}$ contains a single atom $P(\hat{t})$

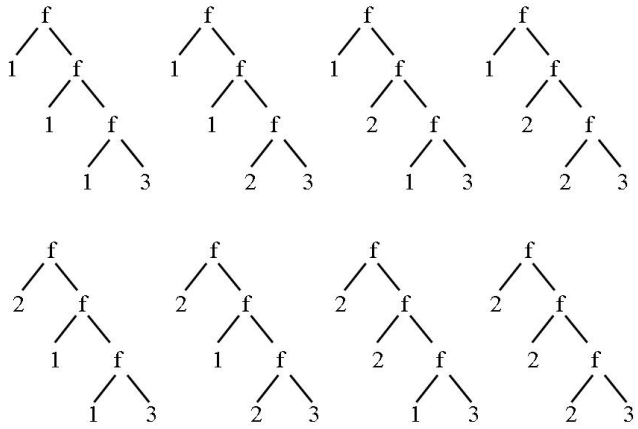


Figure 3. Trees representing the 8 terms in the set $\hat{\mathcal{T}}$ for $c = 8$ (Lemma 9).

in its extension, the extension for F is always empty and hence $F()$ is always false.

Given any subset $\mathcal{S} \subseteq \mathcal{I}$, define $H_{\mathcal{S}}$ as

$$H_{\mathcal{S}} = \left\{ P(\hat{t}) \rightarrow F() \;\middle|\; I_{\hat{t}} \in \mathcal{S} \right\}.$$

We now show that $H_{\mathcal{S}}$ separates interpretations in $\mathcal{S}$ from interpretations in $\mathcal{I} \setminus \mathcal{S}$. Interpretations I in $\mathcal{S}$ falsify one of the clauses in $H_{\mathcal{S}}$ (the one corresponding to I's distinguished term) and hence $I \not\models H_{\mathcal{S}}$. Interpretations I not in $\mathcal{S}$ falsify each clause's antecedent since the terms present in the clauses of $H_{\mathcal{S}}$ are all mapped to the special object $*$ under I. Hence, $I \models H_{\mathcal{S}}$. □

Example 2. Let $c = 4$ so that $\log c = 2$ and

$$\hat{\mathcal{T}} = \{f(1, f(1, 3)), f(1, f(2, 3)), f(2, f(1, 3)), f(2, f(2, 3))\}.$$

Recall that the signature used has function symbols $1/0, 2/0, 3/0, f/2$ and predicate symbols $P/1, F/0$. Each term in $\hat{\mathcal{T}}$ generates an interpretation, e.g. $I_{f(1,f(2,3))}$ consists of:

– Domain: $\{*, 1, 2, 3, f(2, 3), f(1, f(2, 3))\}$.

– Function mappings: $1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3$, $f(2,3) \mapsto f(2,3)$, $f(1,f(2,3)) \mapsto f(1,f(2,3))$ and $f(\cdot,\cdot) \mapsto *$ for every other combination of domain objects.

– Extension for P: $\{P(f(1,f(2,3)))\}$.

– Extension for F: $\{\}$.

The set $\mathcal{I}$ is $\mathcal{I} = \{I_{f(1,f(1,3))}, I_{f(1,f(2,3))}, I_{f(2,f(1,3))}, I_{f(2,f(2,3))}\}$. Suppose that $\mathcal{S} = \{I_{f(1,f(1,3))}, I_{f(1,f(2,3))}\}$ then

$$H_{\mathcal{S}} = \{[P(f(1,f(1,3))) \rightarrow F()], [P(f(1,f(2,3))) \rightarrow F()]\}.$$

The VC dimension construction of (Khardon, 1999a) uses a signature that grows with *NTerms*. The following lemma modifies this construction to use a fixed signature.

LEMMA 10. *For $l \leq t^a$, there exists a set of l interpretations of size $\Theta(t)$ that can be shattered using first order Horn expressions bounded by NTerms $= 2t$, NVariables $= t$, Depth $= \log t$, NLiterals $\leq l$, NPredicates $= 3$, NFunctions $= 1$, Arity $\leq a$ and NClauses $= 1$.*

Proof. We construct a set of interpretations $\mathcal{I}$ that is shattered using first order Horn expressions with parameters as stated. Fix a and t. The expressions use a predicate symbol F of arity 0, a unary predicate L and a predicate symbol Q of arity $\log_t l$. Notice that $\log_t l \leq a$ since $l \leq t^a$. Let

$$Q_{all} = \left\{ Q(i_1, \ldots, i_{\log_t l}) \,\middle|\, i_j \in \{1,..,t\} \text{ for all } j = 1, \ldots, \log_t l \right\},$$

where $\{1,..,t\}$ are some of the domain objects as described below. Notice that $|Q_{all}| = t^{\log_t l} = l$.

Let f be a binary function, and let τ be the term represented by a binary balanced tree of depth $\log t$ whose leaves are labeled by the objects $1 \ldots t$ (in order) and whose internal nodes are labeled by the function symbol f. Such a term contains $2t$ subterms. The term t_2 of Figure 2 represents τ for $t = 8$.

The domain for all the interpretations in $\mathcal{I}$ includes an object for each subterm of τ (including $1,..,t$) and a special object $*$. The function mappings for f follow the functional structure of τ with undefined entries completed by the special domain object $*$. Interpretations include in their extension the atom $L(\tau)$ and all the atoms in Q_{all} except one. The extension for F is always empty. There are l interpretations in $\mathcal{I}$.

Given a subset $\mathcal{S} \subseteq \mathcal{I}$ we define $H_{\mathcal{S}}$ as follows. Let τ' be the result of replacing $j \in \{1,..,t\}$ by the corresponding variable $x_j \in \{x_1,..,x_t\}$

in τ. Let $Q_{\mathcal{S}}$ be the intersection of the $Q()$ atoms in the extensions of all the interpretations in $\mathcal{S}$ after the same substitution. Since all interpretations coincide in the domain and function mappings, an interpretation can be viewed as simply describing a set of true atoms. When we take the intersection of these sets of true atoms we are constructing the set of atoms that are true simultaneously in all the interpretations. Then

$$H_{\mathcal{S}} = L(\tau') \wedge Q_{\mathcal{S}} \rightarrow F().$$

We show that $H_{\mathcal{S}}$ separates $\mathcal{S}$ from its complement $\mathcal{I} \setminus \mathcal{S}$. Suppose $I \in \mathcal{S}$. Take the substitution $\{x_j \mapsto j\}$. Then $I \not\models H_{\mathcal{S}}$ because the antecedent is satisfied (it is a subset of the extension of I) but $F()$ is not. Suppose on the other hand that $I \notin \mathcal{S}$. Substitutions other than $\{x_j \mapsto j\}$ falsify $L(\tau')$. The clause $H_{\mathcal{S}}$ is also satisfied under the substitution $\{x_j \mapsto j\}$ because the "omitted Q" in I's extension is present in $Q_{\mathcal{S}}$. Hence $I \models H_{\mathcal{S}}$. □

Example 3. Suppose $l = 4$ and $t, a = 2$. Then $Q_{all} = \{Q(1,1), Q(1,2), Q(2,1), Q(2,2)\}$ and $\tau = f(1,2)$. Notice that the interpretations coincide in everything except in the Q atom that they leave out. Hence, let us denote by $I_{\hat{Q}}$ the interpretation that leaves atom $\hat{Q}$ out. As an example, the interpretation $I_{Q(2,1)}$ is:

- Domain: $\{*, 1, 2, f(1,2)\}$.
- Function mappings: $1 \mapsto 1, 2 \mapsto 2$, $f(1,2) \mapsto f(1,2)$ and $f(\cdot,\cdot) \mapsto *$ for every other combination of domain objects.
- Extension for Q: $\{Q(1,1), Q(1,2), Q(2,2)\}$ (notice atom $Q(2,1)$ missing!)
- Extension for L: $\{L(f(1,2))\}$.
- Extension for F: $\{\}$.

Suppose that $\mathcal{S} = \{I_{Q(2,1)}, I_{Q(2,2)}\}$. The atoms included in $I_{Q(2,1)}$'s extension for Q are $\{Q(1,1), Q(1,2), Q(2,2)\}$ and the ones in $I_{Q(2,2)}$'s extension for Q are $\{Q(1,1), Q(1,2), Q(2,1)\}$. Hence their intersection is $\{Q(1,1), Q(1,2)\}$ and $Q_{\mathcal{S}} = \{Q(x_1,x_1), Q(x_1,x_2)\}$. $H_{\mathcal{S}}$ is

$$L(f(x_1,x_2)) \wedge Q(x_1,x_1) \wedge Q(x_1,x_2) \rightarrow F().$$

Now the previous two constructions can be combined to get:

LEMMA 11. *For $l \leq t^a$, there exists a set of cl interpretations of size $\Theta(\log c + t)$ that can be shattered using range-restricted and constrained first order Horn expressions bounded by NClauses $\leq c$, NTerms $= \Theta(\log c + t)$, NLiterals $\leq l$, NVariables $= t$, Depth $= \Theta(\log c + \log t)$, Arity $\leq a$, NFunctions $= 5$ and NPredicates $= 4$.*

Proof. Let $\mathcal{I}$ be the set shattered in Lemma 10. We create a new set of interpretations $\mathcal{I}^+$ of cardinality cl in the following way. We have an additional set of c terms constructed in the same way as in Lemma 9, let us denote this set $\hat{\mathcal{T}}_c$. $\hat{\mathcal{T}}_c$ contains c distinct terms of depth $\log c$ each.

We augment the interpretations in the construction of Lemma 10 by associating each $I \in \mathcal{I}$ with a new term $\hat{c}$ in $\hat{\mathcal{T}}_c$ (and hence we create c new interpretations in $\mathcal{I}^+$ for each old interpretation in $\mathcal{I}$). This adds to each interpretation $\log c$ new objects (corresponding to $\hat{c}$ and its subterms) and function mappings following $\hat{c}$'s structure, completing undefined entries with the special object $*$. Additionally, we use a new predicate $P/1$ and include the atom $P(\hat{c})$ in the extension of interpretations with distinguished term $\hat{c} \in \hat{\mathcal{T}}_c$. Hence $|\mathcal{I}^+| = cl$.

Given a subset $\mathcal{S} \subseteq \mathcal{I}$ we define $H_{\mathcal{S}}$ as:

$$H_{\mathcal{S}} = \left\{ L(\tau') \wedge Q_{\mathcal{S}_{\hat{c}}} \wedge P(\hat{c}) \rightarrow F(\tau', \hat{c}) \;\middle|\; \hat{c} \in \hat{\mathcal{T}}_c \right\}, \tag{1}$$

where τ' is the same as above, $\mathcal{S}_{\hat{c}}$ is the subset of interpretations in $\mathcal{S}$ with distinguished term $\hat{c}$, and $Q_{\mathcal{S}_{\hat{c}}}$ is constructed as in Lemma 10. Notice that $H_{\mathcal{S}}$ is both range-restricted and constrained.

We show that $H_{\mathcal{S}}$ separates $\mathcal{S}$ from its complement $\mathcal{I} \setminus \mathcal{S}$. Let I be any interpretation in $\mathcal{I}$. Suppose that $\hat{c}$ is the distinguished term in $\hat{\mathcal{T}}_c$ associated to I. Terms $c' \in \hat{\mathcal{T}}_c$ s.t. $c' \neq \hat{c}$ evaluate to $*$ under I, and every clause in $H_{\mathcal{S}}$ containing $P(c')$ is satisfied. The clause containing $P(\hat{c})$ is falsified iff $I \in \mathcal{S}$ by the same reasoning as in Lemma 10. □

The next result shows that by varying the number of terms we can shatter arbitrarily large sets with a fixed signature.

LEMMA 12. *There exists a set of t interpretations of size $O(t)$ that can be shattered using Horn expressions bounded by NClauses $= 1$, NTerms $\leq 4t$, NLiterals $= 2$, NVariables $= 0$, Depth $= 2\log t + 2$, Arity $= 2$, NFunctions ≤ 9 and NPredicates $= 2$.*

Proof. Let $t = k \log k$ for some $k \in N$. Using the same signature as in Lemma 9 we generate a set $\hat{\mathcal{T}}$ of k terms of depth $\log k$ each. We associate to every interpretation a term in $\hat{\mathcal{T}}$ and an index $i \in \{1, .., \log k\}$ and we denote by $I_{\hat{t},i}$ the interpretation associated to

$(\hat{t}, i) \in \hat{\mathcal{T}} \times \{1, .., \log k\}$. Thus, we have a set of interpretations $\mathcal{I}$ of cardinality $|\mathcal{I}| = |\hat{\mathcal{T}}|\,|\{1, .., \log k\}| = k \log k = t$.

The signature used in this construction uses function symbols $a/0$, $1/0$, $2/0$, $3/0$, $f/2$, $f_0/1$, $f_1/1$, $g/2$, $h/2$, and predicate symbols $M/1$, $F/0$.

Given a subset $\mathcal{S} \subseteq \mathcal{I}$, we construct a ground term $TREE_{\mathcal{S}}$ that associates to every possible term $\hat{t}$ in $\hat{\mathcal{T}}$ a set of indices $l_{\hat{t}}$ where $l_{\hat{t}} = \left\{ i \,\middle|\, I_{\hat{t},i} \in \mathcal{S} \right\}$. The function mappings in each interpretation $I_{\hat{t},i}$ ensure that the term $TREE_{\mathcal{S}}$ evaluates to a special domain object y if and only if the index i appears in the set of indices for term $\hat{t}$ encoded in $TREE_S$. The expression $H_{\mathcal{S}}$ is now defined as:

$$H_{\mathcal{S}} = M(TREE_{\mathcal{S}}) \rightarrow F().$$

Each interpretation includes in its extension the atom $M(y)$ so that the clause $H_{\mathcal{S}}$ is falsified by I iff the term $TREE_{\mathcal{S}}$ evaluates to y under I, i.e., iff $I \in \mathcal{S}$.

We first describe the structure of the term $TREE_{\mathcal{S}}$. We encode the set $l_{\hat{t}}$ with the term $f_{i_1}(f_{i_2}(\cdots f_{i_{\log k}}(a))\cdots)$ where $i_j = 0$ if $j \notin l_{\hat{t}}$ and $i_j = 1$ otherwise. Denote this term by $t_{l_{\hat{t}}}$. As an example, assume $\log k = 6$ and let $l_{\hat{t}} = \{1, 4, 5\}$. Then, $t_{l_{\hat{t}}} = f_1(f_0(f_0(f_1(f_1(f_0(a))))))$. Notice that we are using two unary functions f_0 and f_1 and a constant a. Next we use a binary function g to encode the association between terms $\hat{t}$ and their sets of indices $l_{\hat{t}}$ as $g(\hat{t}, t_{l_{\hat{t}}})$. Finally, $TREE_{\mathcal{S}}$ is constructed as a balanced tree, using a binary function h, whose leaves are terms of the form $g(\hat{t}, t_{l_{\hat{t}}})$, for every $\hat{t} \in \hat{\mathcal{T}}$.

Example 4. Let $k = 4$. Then $\hat{\mathcal{T}} = \{\hat{t}_1, \hat{t}_2, \hat{t}_3, \hat{t}_4\}$, where

- $\hat{t}_1 = f(1, f(1, 3)), \quad \hat{t}_2 = f(1, f(2, 3))$
- $\hat{t}_3 = f(2, f(1, 3)), \quad \hat{t}_4 = f(2, f(2, 3))$

If $\mathcal{S} = \{I_{\hat{t}_1,1}, I_{\hat{t}_2,2}, I_{\hat{t}_3,1}, I_{\hat{t}_3,2}\}$, then:

- $l_{\hat{t}_1} = \{1\}$, $l_{\hat{t}_2} = \{2\}$, $l_{\hat{t}_3} = \{1, 2\}$ and $l_{\hat{t}_4} = \{\}$.
- $t_{l_{\hat{t}_1}} = f_1(f_0(a))$, $t_{l_{\hat{t}_2}} = f_0(f_1(a))$, $t_{l_{\hat{t}_3}} = f_1(f_1(a))$ and $t_{l_{\hat{t}_4}} = f_0(f_0(a))$.
- $TREE_{\mathcal{S}}$ is depicted in Figure 4.

Let us now describe in detail the domain and function mappings for interpretation $I_{\hat{t},i}$. The domain objects are:

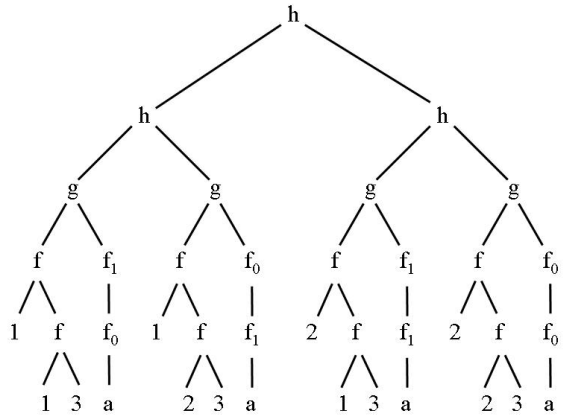


Figure 4. Tree representing $TREE_{\mathcal{S}}$ in Example 4.

- Three special objects $*, y, n$.
- Up to $\log k + 3$ distinct objects that represent all terms and subterms present in the distinguished term $\hat{t}$.
- Up to $2k + 1$ objects representing all the possible terms and subterms of the vector indices $f_{i_1}(f_{i_2}(\cdots f_{i_{\log k}}(a))\cdots)$ for all possible $i_j \in \{0, 1\}$ where $1 \leq j \leq \log k$.

The function mappings are:

- The constants $1, 2, 3$ are mapped to objects $1, 2, 3$. The mapping for binary function f follows the functional structure of $\hat{t}$, with undefined entries mapped to the special object $*$.
- The constant a is mapped to object a. Unary functions f_0 and f_1 also mimic the functional structure of terms and subterms of $f_{i_1}(f_{i_2}(\cdots f_{i_{\log k}}(a))\cdots)$ for all possible $i_j \in \{0, 1\}$ where $1 \leq j \leq \log k$.
- The binary function $g(t_1, t_2)$ is mapped to special object y iff $t_1 = \hat{t}$ and the unary function used at depth i in term t_2 is f_1. Otherwise it is set to the special object n. Note that while function mappings

cannot be based on term structure, we have identified each subterm of t_2 with a domain object so that this is a valid mapping.

– Finally, the binary function $h(a1, a2)$ is mapped to domain object y iff either $a1 = y$ or $a2 = y$, otherwise it is mapped to object n.

Finally, for each interpretation $I_{\hat{t},i}$ the only atom true in the interpretation is $M(y)$.

We prove that $I_{\hat{t},i}$ falsifies $H_{\mathcal{S}}$ iff $I_{\hat{t},i} \in \mathcal{S}$. Notice that $I_{\hat{t},i}$ falsifies $H_{\mathcal{S}}$ iff $I_{\hat{t},i}$ satisfies the atom $M(TREE_{\mathcal{S}})$ iff the term $TREE_{\mathcal{S}}$ is mapped to the domain object y under $I_{\hat{t},i}$ iff some term $g(t_1, t_2)$ is mapped to y iff term $g(\hat{t}, t_2)$ is mapped to y (other terms $g(t_1, t_2)$ where $t_1 \neq \hat{t}$ are mapped to n by construction) iff the unary function used at depth i in term t_2 is f_1 iff $I_{\hat{t},i} \in \mathcal{S}$.

We finally quantify the complexity of the parameters used in $H_{\mathcal{S}}$: it has 1 clause, 2 literals, no variables, uses one single term of depth $\Theta(\log k)$ (that is $O(\log t)$) which contains $\Theta(k \log k)$ subterms (that is $\Theta(t)$ subterms) that are built from 4 constants, 5 function symbols whose maximal arity is 2. □

Example 5. Building on Example 4, we illustrate how $I_{\hat{t}_1,1} \in \mathcal{S}$ satisfies $H_{\mathcal{S}}$ but $I_{\hat{t}_1,2} \notin \mathcal{S}$ does not. This is clear if we look at both interpretations' domain and function mappings in detail. Recall that $\hat{t}_1 = f(1, f(1, 3))$. Since the distinguished term for both interpretations is $\hat{t}_1$, they have the same domains, consisting of the set:

– $\{*,\ y,\ n,\ 1,\ 2,\ 3,\ f(1,3),\ f(1,f(1,3)), a,\ f_0(a), f_1(a),\ f_0(f_0(a)),\ f_0(f_1(a)),\ f_1(f_0(a)),\ f_1(f_1(a))\}$.

The functional mappings for the constants and functions $f/2$, $f_0/1$, $f_1/1$, and $h/2$ are the same for both interpretations:

– $1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3, a \mapsto a$

– $f(1,3) \mapsto f(1,3), f(1,f(1,3)) \mapsto f(1,f(1,3))$,
o/w map to $*$

– $f_0(a) \mapsto f_0(a), f_0(f_0(a)) \mapsto f_0(f_0(a)), f_0(f_1(a)) \mapsto f_0(f_1(a))$,
o/w map to $*$

– $f_1(a) \mapsto f_1(a), f_1(f_0(a)) \mapsto f_1(f_0(a)), f_1(f_1(a)) \mapsto f_1(f_1(a))$,
o/w map to $*$

– $h(y, \cdot) \mapsto y, h(\cdot, y) \mapsto y,$ o/w map to n

The functional mapping for $g/2$ in $I_{\hat{t}_1,1}$ is (notice that first index is f_1 always):

– $g(f(1,f(1,3)),\underline{f_1}(f_0(a))) \mapsto y, g(f(1,f(1,3)),\underline{f_1}(f_1(a))) \mapsto y$, o/w map to n

The functional mapping for $g/2$ in $I_{\hat{t}_1,2}$ is (notice second index is f_1 always):

– $g(f(1,f(1,3)),f_0(\underline{f_1}(a))) \mapsto y, g(f(1,f(1,3)),f_1(\underline{f_1}(a))) \mapsto y$, o/w map to n.

Hence, the term of $TREE_{\mathcal{S}}$ corresponding to the term $\hat{t}_1$ (left-most application of g in $TREE_{\mathcal{S}}$, see Figure 4), evaluates to y for $I_{\hat{t}_1,1}$. However, for $I_{\hat{t}_1,2}$, this term evaluates to n. Since the remaining g's of $TREE_{\mathcal{S}}$ evaluate all to n in both interpretations, the final evaluation for $I_{\hat{t}_1,1}$ is y whereas the evaluation of $TREE_{\mathcal{S}}$ is n for $I_{\hat{t}_1,2}$.

Recall that $H_{\mathcal{S}} = M(TREE_{\mathcal{S}}) \rightarrow F()$. Since the extension of the interpretations contain $M(y)$ but nothing else, $I_{\hat{t}_1,1}$ violates $H_{\mathcal{S}}$ but $I_{\hat{t}_1,2}$ does not.

As before we can extend the previous construction to introduce a dependence on c:

LEMMA 13. *There exists a set of ct interpretations of size* $O(\log c + t)$ *that can be shattered using range-restricted and constrained first order Horn expressions bounded by NClauses* $\leq c$, *NTerms* $= \Theta(t + \log c)$, *NLiterals* $= 2$, *NVariables* $= 0$, *Depth* $= O(\log t + \log c)$, *Arity* $= 2$, *NFunctions* ≤ 9, *and NPredicates* $= 3$.

Proof. We extend the previous construction. Let $\mathcal{I}$ be the set shattered in Lemma 12. We create a new set of interpretations $\mathcal{I}^+$ of cardinality ct in the following way. We have an additional set of c terms constructed in the same way as in Lemma 9 using the constants 1,2,3 and a binary function symbol g. Let us denote this set $\hat{\mathcal{T}}_c$. As in Lemma 9, $\hat{\mathcal{T}}_c$ contains c distinct terms of depth $\log c$ each. Notice that we can safely re-use 1,2,3 and g since these are never combined in the construction of Lemma 12.

As before, we augment the interpretations in the construction of Lemma 12 by associating $I \in \mathcal{I}$ with a new term in $\hat{\mathcal{T}}_c$ (and hence we create c new interpretations in $\mathcal{I}^+$ for each old interpretation in $\mathcal{I}$), adding $\log c$ new objects and the corresponding functional mappings following the term's structure. Hence $|\mathcal{I}^+| = ct$. In addition we modify

the predicates M and F that now have arity 2. The only atom true in I is $M(\hat{c}, y)$, where $\hat{c}$ is the distinguished term associated to I.

For each subset $\mathcal{S} \subseteq \mathcal{I}$ we define

$$H_{\mathcal{S}} = \left\{ M(\hat{c}, TREE_{\mathcal{S}_{\hat{c}}}) \rightarrow F(\hat{c}, TREE_{\mathcal{S}_{\hat{c}}}) \;\middle|\; \hat{c} \in \hat{\mathcal{T}}_c \right\},$$

where $\mathcal{S}_{\hat{c}}$ is the subset of interpretations in $\mathcal{S}$ with distinguished term $\hat{c}$. Notice that $H_{\mathcal{S}}$ is both range-restricted and constrained.

We finally prove that I falsifies $H_{\mathcal{S}}$ iff $I \in \mathcal{S}$. Suppose that $\hat{c}$ is the distinguished term in $\hat{\mathcal{T}}_c$ associated to I. I contains the atom $M(\hat{c}, y)$ in its extension, and every clause $M(c', TREE_{\mathcal{S}_{c'}}) \rightarrow F(c', TREE_{\mathcal{S}_{c'}})$ in $H_{\mathcal{S}}$ s.t. $\hat{c} \neq c'$ is satisfied since term c' does not evaluate to domain object $\hat{c}$ under I. The clause $M(\hat{c}, TREE_{\mathcal{S}_{\hat{c}}}) \rightarrow F(\hat{c}, TREE_{\mathcal{S}_{\hat{c}}})$ is falsified iff $I \in \mathcal{S}_{\hat{c}}$ by the same reasoning as in Lemma 12. □

Combining Lemmas 11 and 13 we conclude:

THEOREM 14. *Let $\mathcal{S}$ be a signature with at least 9 function symbols and 4 predicates of arity at least 2. The VC dimension of the class of range-restricted and constrained first order Horn expressions over $\mathcal{S}$ with at most c clauses, each using up to l literals and $t + \log c$ terms is $\Omega(cl + ct)$.*

COROLLARY 15. *The VC dimension of the class of range-restricted and constrained expressions in $\mathcal{H}^{\leq c,t,l}$ for learning from interpretations is $\tilde{\Theta}(cl + ct)$.*

6.2. Learning from entailment

In the model of learning from entailment (Frazier and Pitt, 1993), examples are clauses and class membership is determined by logical consequence. That is, a clause C is a member of the concept represented by target expression T iff $T \models C$. Thus a concept is associated with the set of clauses that it implies. The notions of equivalence and membership queries are adapted so that the examples used are clauses rather than interpretations.

In some cases it is easy to transform a lower bound from learning from interpretations to learning from entailment. In particular the construction in Lemma 11 uses interpretations whose term structure is simple. Any object that appears in the extension of any predicate has a unique maximal term that describes it. Thus in some sense one can think of the relation $I \not\models H_{\mathcal{S}}$ as subsumption between the clauses in $H_{\mathcal{S}}$ and the "term structure" of the extension in I.

Example 6. To illustrate this property consider a signature with one predicate p of arity 1, two constants a, b, and one function f of arity 1. Consider two interpretations with the same domain $\{1, 2, *\}$, same extension where $p(2)$ is the only true atom, and same mapping for f with $f(1) = 2$, $f(2) = *$, and $f(*) = *$. The first interpretation maps $a \to 1$, $b \to *$. In this case we can give a "maximal atom" $p(f(a))$ to describe what is true in the interpretation. The antecedent of any clause that is falsified by the interpretation must subsume $p(f(a))$. The second interpretation maps $a \to 1$, $b \to 2$. In this case there are two possible "maximal atoms" $p(f(a))$ and $p(b)$ describing what is true in the interpretation and we cannot make the same claim regarding subsumption.

If every true fact refers to a unique description of a maximal object, then we can turn things around and make an antecedent of a clause C_I from the extension of predicates in the interpretation. If we can also choose an appropriate consequent then such a construction would satisfy $I \not\models H_{\mathcal{S}}$ iff $H_{\mathcal{S}} \models C_I$. We can therefore construct a set of clauses that are shattered from the previous construction. One can abstract this idea and show how such a transformation can be done (see related discussion in (Khardon, 1999b)) and that we get a shattered set. But in our case a direct application as given in the following lemma is easier to see:

LEMMA 16. *For $l \leq t^a$, there exists a set of cl clauses that can be shattered using range-restricted and constrained first order Horn expressions bounded by NClauses $\leq c$, NTerms $= \Theta(\log c + t)$, NLiterals $\leq l$, NVariables $\leq t$, Depth $= \Theta(\log c + \log t)$, Arity $\leq a$, NFunctions $= 5$ and NPredicates $= 4$.*

Proof. We give a set of clauses $\mathcal{C}l$ and show that it can be shattered. Let $\mathcal{I}^+$ be as in Lemma 11 and let $\mathcal{C}l = \{C_I | I \in \mathcal{I}^+\}$ where for $I \in \mathcal{I}^+$ whose associated term is $\hat{c}$ we have

$$C_I = H_{\{I\}}(\text{from Eq. (1)}) = L(\tau') \wedge Q_{\{I\}} \wedge P(\hat{c}) \to F(\tau', \hat{c}).$$

Given a subset $\mathcal{S} \subseteq \mathcal{C}l$ we define $H_{\mathcal{S}}$ as in Eq. (1) i.e.

$$H_{\mathcal{S}} = \left\{ L(\tau') \wedge Q_{\mathcal{S}_{\hat{c}}} \wedge P(\hat{c}) \to F(\tau', \hat{c}) \;\middle|\; \hat{c} \in \hat{\mathcal{T}}_c \right\},$$

where we use the interpretations corresponding to the clauses $\mathcal{S}$ in the definition of $H_{\mathcal{S}}$. Notice that in our case implication and subsumption are equivalent since no chaining of rules or self subsumption is possible (Gottlob, 1987). Let $C \in \mathcal{C}l$ be a clause with distinguished term $\hat{c}$. If

$C \in \mathcal{S}$, the corresponding clause in $H_{\mathcal{S}}$ contains a subset of the atoms of C (no substitution needs to be applied) and therefore $H_{\mathcal{S}} \models C$. On the other hand consider $C \notin \mathcal{S}$. It is clear that clauses in $H_{\mathcal{S}}$ with other associated terms cannot be used to imply C. For the clause with the same associated term only the empty substitution can be used due to the atom $L(\tau')$. However in this case the "omitted Q" atom in C is present in the clause in $H_{\mathcal{S}}$ and the clause cannot be subsumed. □

The term structure in the interpretations in Lemmas 12 and 13 is more complex and we cannot use the extensions directly in clause bodies. However a related construction yields the same bounds.

LEMMA 17. *There exists a set of t clauses that can be shattered using Horn expressions bounded by NClauses $= 1$, NTerms $\leq 2t$, NLiterals $= 1$, NVariables $\leq t$, Depth $= \log t$, Arity $= 2$, NFunctions $= 3$ and NPredicates $= 1$.*

Proof. For each $1 \leq i \leq t$, let $\hat{t}_i$ be a term of depth $\log t$ represented by a binary tree of t leaves with binary function symbol f. Each $\hat{t}_i$ has as the i-th leaf a constant a and in all other leaves a constant b.

Let P be a unary predicate symbol. The set of clauses to be shattered is $\mathcal{C}l = \{C_i \mid 1 \leq i \leq t\}$, where C_i is the single literal $P(\hat{t}_i)$. Clearly, $|\mathcal{C}l| = t$.

Given a subset $\mathcal{S} \subseteq \mathcal{C}l$, let $TERM_{\mathcal{S}}$ be the term represented by a balanced binary tree of depth $\log t$ with internal nodes labeled by a function symbol f and with the constant b in a leaf i if and only if $C_i \in \mathcal{S}$. All other leaves are labeled with distinct variables, namely, a leaf in position j s.t. $C_j \notin \mathcal{S}$ contains a variable x_j. $H_{\mathcal{S}}$ is defined as the single clause with just one literal:

$$H_{\mathcal{S}} = P(TERM_{\mathcal{S}}).$$

Now we prove that $C_i \in \mathcal{S}$ iff $H_{\mathcal{S}} \not\models C_i$, or equivalently, that $C_i \in \mathcal{S}$ iff $P(TERM_{\mathcal{S}}) \not\models P(\hat{t}_i)$. Fix any C_i. By construction the i-th leaf of $\hat{t}_i$ contains the constant a. If $C_i \in \mathcal{S}$, then the i-th leaf of $TERM_{\mathcal{S}}$ contains the constant b and subsumption is not possible. Therefore, $P(TERM_{\mathcal{S}}) \not\models P(\hat{t}_i)$. If $C_i \notin \mathcal{S}$, then $TERM_{\mathcal{S}}$ contains a variable x_i in the i-th leaf. The substitution $\theta = \{x_i \mapsto a\} \cup \{x_j \mapsto b \mid 1 \leq j \leq t \text{ and } j \neq i\}$ is such that $P(TERM_{\mathcal{S}})\theta = P(\hat{t}_i)$ so that $P(TERM_{\mathcal{S}}) \models P(\hat{t}_i)$. □

The next lemma extends this construction to include a dependence on c:

LEMMA 18. *There exists a set of ct clauses that can be shattered using range-restricted and constrained Horn expressions bounded by NClauses* $\leq c$*, NTerms* $= \Theta(t + \log c)$*, NLiterals* $= 2$*, NVariables* $\leq t$*, Depth* $= O(\log t + \log c)$*, Arity* $= 2$*, NFunctions* ≤ 4 *and NPredicates* $= 2$*.*

Proof. We extend the construction in the previous lemma. First create a set of c distinct terms $\hat{\mathcal{T}}_c$ as in Lemma 9. It is safe to reuse the same binary function symbol f and the constants a and b; hence a single extra constant is needed to mimic the construction from Lemma 9 of $\hat{\mathcal{T}}_c$.

Let P, R be binary predicate symbols. The new set of clauses is

$$\mathcal{C}l = \left\{ P(t_i, \hat{c}) \rightarrow R(t_i, \hat{c}) \;\middle|\; 1 \leq i \leq t \text{ and } \hat{c} \in \hat{\mathcal{T}}_c \right\}.$$

Clearly, $|\mathcal{C}l| = |\{1, .., t\}| \times \left|\hat{\mathcal{T}}_c\right| = tc$.

Given a subset $\mathcal{S} \subseteq \mathcal{C}l$, let $H_{\mathcal{S}}$ be

$$H_{\mathcal{S}} = \left\{ P(\mathit{TERM}_{\mathcal{S}_{\hat{c}}}, \hat{c}) \rightarrow R(\mathit{TERM}_{\mathcal{S}_{\hat{c}}}, \hat{c}) \;\middle|\; \hat{c} \in \hat{\mathcal{T}}_c \right\},$$

where $\mathcal{S}_{\hat{c}}$ is the subset of $\mathcal{S}$ of clauses that are associated to the term $\hat{c}$. Notice that $H_{\mathcal{S}}$ is both range-restricted and constrained.

Let $C_{i,\hat{c}}$ be the clause in $\mathcal{C}l$ that contains the terms t_i and $\hat{c}$. We next show that $C_{i,\hat{c}} \in \mathcal{S}$ iff $H_{\mathcal{S}} \not\models C_{i,\hat{c}}$. Notice that if $\hat{c} \neq c'$ then $P(\mathit{TERM}_{\mathcal{S}_{c'}}, c') \rightarrow R(\mathit{TERM}_{\mathcal{S}_{c'}}, c') \not\models C_{i,\hat{c}}$. Hence, $H_{\mathcal{S}} \models C_{i,\hat{c}}$ iff $P(\mathit{TERM}_{\mathcal{S}_{\hat{c}}}, \hat{c}) \rightarrow R(\mathit{TERM}_{\mathcal{S}_{\hat{c}}}, \hat{c}) \models C_{i,\hat{c}}$. Finally, to prove that $C_{i,\hat{c}} \in \mathcal{S}$ iff $P(\mathit{TERM}_{\mathcal{S}_{\hat{c}}}, \hat{c}) \rightarrow R(\mathit{TERM}_{\mathcal{S}_{\hat{c}}}, \hat{c}) \not\models C_{i,\hat{c}}$ it is sufficient to observe that $C_{i,\hat{c}} \in \mathcal{S}$ iff $C_{i,\hat{c}} \in \mathcal{S}_{\hat{c}}$, so that a similar argument as in Lemma 17 applies. □

Combining Lemmas 16 and 18 we conclude:

THEOREM 19. *Let $\mathcal{S}$ be a signature with at least 9 function symbols and 4 predicates of arity at least 2. The VC dimension of the class of range-restricted and constrained first order Horn expressions over $\mathcal{S}$ with at most c clauses, each using up to l literals and $t + \log c$ terms in the framework of learning from entailment is $\Omega(cl + ct)$.*

COROLLARY 20. *The VC dimension of the class of range-restricted and constrained expressions in $\mathcal{H}^{\leq c,t,l}$ for learning from entailment is $\tilde{\Theta}(cl + ct)$.*

Now applying the lower bound given by (Maass and Turán, 1992) we can conclude:

COROLLARY 21. *Any algorithm that exactly learns the class of range-restricted and constrained expressions in $\mathcal{H}^{\leq c,t,l}$ for either learning from interpretations or learning from entailment must make $\Omega(cl + ct)$ membership and equivalence queries.*

7. Conclusions and future work

The paper studies different complexity parameters for first order learnability. The results show that the standard notion of size is not polynomially related to parameters that are commonly used in the literature, identify an alternative notion of size that can be captured, and characterize the VC-dimension showing that the new size and parameters are indeed crucial for learnability. This gives a uniform treatment to different ways of quantifying the complexity and puts previous work in context so that lower bounds can be interpreted appropriately.

The results are also useful in clarifying the complexity of recent algorithms on learning Horn expressions with equivalence and membership queries. The case of Horn definitions (with a single head) (Reddy and Tadepalli, 1997) is indeed polynomial in $c + l + t$. Other results are either not polynomial (Arias and Khardon, 2002) or rely on syntax based oracles (Arimura, 1997, Reddy and Tadepalli, 1998) and (Krishna Rao and Sattar, 1998). Our results in Arias and Khardon (2002) show that constrained expressions as well as range-restricted expressions are learnable with complexity polynomial in $c+t^v+t^a$, where v is the number of variables per clause and a is the maximum arity of predicates and function symbols (we simplify here by ignoring some of the parameters). Note that t^a essentially bounds l but may in fact be much larger than l. This issue seems to arise in any context where multiple consequents are possible and identifying these may require looking at the t^a possibilities. More importantly, it is not known whether the exponential dependence on v is necessary and this remains the main discrepancy between known lower and upper bounds. As pointed out above, VC based bounds cannot resolve this question since they are limited by expression size. The notion of certificate size of concept classes, developed by Hellerstein et al., Hegedűs (1996, 1995) gives both lower and upper bounds for query complexity and thus may provide tools to do so. Characterizing the certificate complexity of first order classes is an interesting direction for future work. Preliminary results solving some cases in propositional logic are reported in (Arias et al., 2003).

References

Angluin, D.: 1988, 'Queries and Concept Learning'. *Machine Learning* **2**(4), 319–342.

Arias, M. and R. Khardon: 2002, 'Learning Closed Horn Expressions'. *Information and Computation* **178**, 214–240.

Arias, M., R. Khardon, and R. A. Servedio: 2003, 'Polynomial Certificates for Propositional Classes'. In: *Proceedings of the Conference on Computational Learning Theory*. pp. 537–551, Springer-Verlag.

Arimura, H.: 1997, 'Learning Acyclic First-order Horn Sentences from Entailment'. In: *Proceedings of the International Conference on Algorithmic Learning Theory*. Springer-Verlag.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth: 1989, 'Learnability and the Vapnik-Chervonenkis Dimension'. *Journal of the ACM* **36**(4), 929–965.

Cohen, W.: 1995, 'PAC-learning recursive logic programs: Efficient algorithms'. *Journal of Artificial Intelligence Research* **2**.

De Raedt, L.: 1997, 'Logical Settings for Concept Learning'. *Artificial Intelligence* **95**(1), 187–201.

De Raedt, L. and S. Džeroski: 1994, 'First order jk-clausal theories are PAC-learnable'. *Artificial Intelligence* **70**.

Džeroski, S., S. Muggleton, and S. Russell: 1992, 'PAC-Learnability of Determinate Logic Programs'. In: D. Haussler (ed.): *Proceedings of the Conference on Computational Learning Theory*. Pittsburgh, PA, pp. 128–135, ACM Press.

Ehrenfeucht, A., D. Haussler, M. Kearns, and L. Valiant: 1989, 'A General Lower Bound on the Number of Examples Needed for Learning'. *Information and Computation* **82**(3).

Frazier, M. and L. Pitt: 1993, 'Learning from Entailment: An Application to propositional Horn sentences'. In: *Proceedings of the International Conference on Machine Learning*. Amherst, MA, pp. 120–127, Morgan Kaufmann.

Gottlob, G.: 1987, 'Subsumption and implication'. *Information Processing Letters* **24**(2), 109–111.

Grohe, M. and G. Turán: 2002, 'Learnability and Definability in Trees and Similar Structures'. In: *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. pp. 645–658, Springer-Verlag. LNCS 2285.

Hegedűs, T.: 1995, 'On Generalized Teaching Dimensions and the query complexity of learning'. In: *Proceedings of the Conference on Computational Learning Theory*. New York, NY, USA, pp. 108–117, ACM Press.

Hellerstein, L., K. Pillaipakkamnatt, V. Raghavan, and D. Wilkins: 1996, 'How Many Queries are Needed to Learn?'. *Journal of the ACM* **43**(5), 840–862.

Horváth, T. and G. Turán: 2001, 'Learning logic programs with structured background knowledge'. *Artificial Intelligence* **128**(1-2), 31–97.

Kearns, M. and U. Vazirani: 1994, *An Introduction to Computational Learning Theory*. Cambridge: MIT Press.

Khardon, R.: 1999a, 'Learning Function Free Horn Expressions'. *Machine Learning* **37**.

Khardon, R.: 1999b, 'Learning Range Restricted Horn Expressions'. In: *Proceedings of the Fourth European Conference on Computational Learning Theory*. pp. 111–125, Springer-Verlag.

Kietz, J. U. and S. Džeroski: 1994, 'Inductive logic programming and learnability'. *SIGART Bulletin* **5**(1), 22–32.

Krishna Rao, M. and A. Sattar: 1998, 'Learning from Entailment of Logic Programs with Local Variables'. In: *Proceedings of the International Conference on Algorithmic Learning Theory*. Springer-Verlag.

Lloyd, J. W.: 1987, *Foundations of logic programming; (2nd extended ed.)*. Springer-Verlag.

Maass, W. and G. Turán: 1992, 'Lower bound methods and separation results for online learning models'. *Machine Learning* **9**, 107–145.

Maass, W. and G. Turán: 1995, 'On learnability and predicate logic (Extended Abstract)'. In: *Proceedings of the 4th Bar-Ilan Symposium on Foundations of AI (BISFAI)*.

Muggleton, S. and L. De Raedt: 1994, 'Inductive Logic Programming: Theory and Methods'. *The Journal of Logic Programming* **19 & 20**, 629–680.

Muggleton, S. and C. Feng: 1992, 'Efficient Induction of Logic Programs'. In: S. Muggleton (ed.): *Inductive Logic Programming*. Academic Press, pp. 281–298.

Reddy, C. and P. Tadepalli: 1997, 'Learning Horn Definitions with Equivalence and Membership Queries'. In: *International Workshop on Inductive Logic Programming*. Prague, Czech Republic, pp. 243–255, Springer-Verlag.

Reddy, C. and P. Tadepalli: 1998, 'Learning First Order Acyclic Horn Programs from Entailment'. In: *International Conference on Inductive Logic Programming*. Madison, WI, pp. 23–37, Springer-Verlag.

Vapnik, V. and A. Chervonenkis: 1971, 'On the Uniform Convergence of Relative Frequencies of Events to their probabilities'. *Theory of Probability and its applications* **XVI**(2), 264–280.