Probably approximately correct learning of Horn envelopes from queries

Daniel Borchmann, Tom Hanika[a,*], Sergei Obiedkov[b]

[a] *Knowledge & Data Engineering Group, University of Kassel, Germany*
[b] *National Research University Higher School of Economics, Moscow, Russia*

Abstract

We propose an algorithm for learning the Horn envelope of an arbitrary domain using an expert, or an oracle, capable of answering certain types of queries about this domain. Attribute exploration from formal concept analysis is a procedure that solves this problem, but the number of queries it may ask is exponential in the size of the resulting Horn formula in the worst case. We recall a well-known polynomial-time algorithm for learning Horn formulas with membership and equivalence queries and modify it to obtain a polynomial-time probably approximately correct algorithm for learning the Horn envelope of an arbitrary domain.

Keywords: PAC learning, attribute exploration, FCA, formal concept
2010 MSC: 68T27, 06B99

1. Introduction

The learnability of concepts from oracle queries has received significant attention in learning theory. The most common types of oracles investigated in the literature are membership and equivalence oracles, and for these types of oracles various results have been obtained showing learnability in polynomial time. One of the most prominent examples is the fact that Horn formulas can be learnt in polynomial time with access to membership and equivalence oracles [1].

In the realm of formal concept analysis [2], a different learning method has been established almost simultaneously with the standard query learning setting. The theory of formal concept analysis emerged as a subfield of mathematical order theory, more precisely of lattice theory, and it studies lattices as *hierarchies of concepts*. Since its emergence in the early 1980s, it has evolved into a rich theory with a wide range of applications. An important technique of formal concept analysis is the *attribute exploration* algorithm. This algorithm aims at

*Corresponding author
Email addresses: daniel@algebra20.de (Daniel Borchmann), tom.hanika@cs.uni-kassel.de (Tom Hanika), sergei.obj@gmail.com (Sergei Obiedkov)

learning a Horn representation, also called a *Horn envelope*, of the knowledge of a *domain expert*. A Horn envelope of a theory is a Horn formula whose set of models includes all the models of the theory and is as specific as possible [3]. Here, a domain expert is an oracle that is able to answer questions of the form "Does A imply B in your domain?", where A and B are conjunctions of atomic propositions. If $A \to B$ is indeed true, the expert confirms this implication. Otherwise, the expert gives a *counterexample*, i.e., a model C of the domain containing A but not B.

A large number of variants of the classical attribute exploration algorithm have been investigated, and a wide range of applications have been proposed and examined [4]. In particular, it turned out that the notion of a domain expert is well suited for practical applications. However, in the worst case, attribute exploration requires exponential time in the number of propositional variables and the size of the resulting Horn formula. This is because it enumerates all the models of the domain as a byproduct, and their number may be exponential in the size of the Horn formula. On the other hand, an *exact* computation of the Horn envelope of real-world domains is rarely useful in practice, as special cases may lead to artificial Horn formulas.

The problem of exponentially many queries does not exist in the case of using membership and equivalence queries [1], but in this algorithm the queries are asked with respect to the Horn envelope rather than with respect to the actual domain we want to explore. Therefore, in our setting, this algorithm is applicable only to Horn domains (for which the Horn envelope is the same as the domain theory). But even in this case, equivalence queries may be hard to answer because they require an oracle to provide a negative counterexample, a description of something that does not exist in the domain.

In this work we want to bring together the best of both approaches: we want to devise a learning algorithm that requires only polynomial time in the size of the output and issues only polynomially many queries to a domain expert. To this end, we propose a *probably approximately correct* (PAC) version of attribute exploration that computes an approximation of the Horn envelope of the domain theory using queries about the validity of Horn formulas, just as in classical attribute exploration. We investigate two notions of approximation of Horn envelopes: one is based on the agreement of a large fraction of models, akin to the one used by [5]. A second, novel, and stronger notion called *ε-strong Horn approximation* is based on the requirement of the involved closure operators to coincide on a large fraction of subsets. The latter makes it possible to avoid some very weak approximations, as we shall discuss later.

We state the problem precisely in Section 2. We then recall the algorithm from [1] in Section 3. It serves the basis for our PAC algorithms presented in Section 4. The basic version does not need counterexamples: it only needs the oracle to confirm or reject proposed Horn clauses. Taking counterexamples into account makes it possible to reduce the number of queries. We show the effectiveness of our approach by means of example with real-world data in Section 5.

2. Preliminaries

A *Horn clause* over a set of propositional variables Φ is a disjunction of variables from Φ and their negations (i.e., *literals*) containing at most one unnegated variable (*positive literal*). The negated variables form the *body* of the Horn clause, whereas the unnegated variable is called the *head* of the clause. A *definite Horn clause* contains exactly one positive literal. A *Horn sentence* or *Horn formula* is a conjunction of Horn clauses. A Horn sentence consisting of definite Horn clauses with the same body can equivalently be represented by an *implication* $p_1 \wedge \cdots \wedge p_n \to q_1 \wedge \cdots \wedge q_m$, where $p_i, q_i \in \Phi$. If one of the clauses sharing the body is not definite, i.e., if it contains no positive literal, the corresponding sentence can be represented by an implication $p_1 \wedge \cdots \wedge p_n \to \bot$, where $\bot \notin \Phi$ is the propositional constant falsum.

We will predominantly use set notation for representing Horn clauses and sentences. In particular, we will use notation $A \to B$, where $A, B \subseteq \Phi$, to represent the implication

$$\bigwedge_{p \in A} p \to \bigwedge_{q \in B} q.$$

Here, A will be referred to as the *premise* and B as the *conclusion* of the implication $A \to B$. Abusing notation, we identify $\bot$ with the set $\Phi \cup \{\bot\}$, which implies, e.g., that $A \subseteq \bot$ and, consequently, $A \cap \bot = A$ for any $A \subseteq \Phi$. A Horn sentence $\mathcal{H}$ will be regarded as a set of implications, and $|\mathcal{H}|$ will stand for the number of implications in $\mathcal{H}$.

A *variable assignment* V is a function that maps every propositional variable in Φ to 1 (true) or 0 (false). Again, we will often identify a variable assignment with the set of variables that it maps to 1. An assignment V is a *model* of a Horn clause h (notation $V \models h$) if h evaluates to 1 under the assignment V (with the standard semantics of logical connectives). V is a model of a Horn sentence $\mathcal{H}$ (notation: $V \models \mathcal{H}$) if it is a model of every clause it contains. As a special case, it is easy to see that V is a model of an implication $A \to B$ if $A \not\subseteq V$ or $B \subseteq V$. We denote by $\operatorname{Mod} \mathcal{H}$ the set of all models of $\mathcal{H}$.

Two Horn sentences are *equivalent* if they have exactly the same sets of models. A Horn sentence $\mathcal{H}_1$ *entails* a Horn sentence $\mathcal{H}_2$ if every model of $\mathcal{H}_1$ is a model of $\mathcal{H}_2$ (notation: $\mathcal{H}_1 \models \mathcal{H}_2$). It is well-known that the set of models of a Horn sentence is closed under intersection. This makes it possible to define $\mathcal{H}(V)$ as the unique minimal model of $\mathcal{H}$ in which 1 is assigned to all variables in V and as $\Phi \cup \{\bot\}$ if no model containing V exists. It is not difficult to see that $\mathcal{H}(\cdot)$ is the *closure operator* (i.e., it is monotone, extensive, and idempotent) corresponding to the closure system that consists of models of $\mathcal{H}$ and, if Φ is not a model, of $\Phi \cup \{\bot\}$. Of course, $\mathcal{H}(V) = V$ precisely for the models of $\mathcal{H}$; we will sometimes refer to these models as sets *closed* with respect to $\mathcal{H}(\cdot)$. Obviously, if $\mathcal{H}_1$ is equivalent to $\mathcal{H}_2$, then $\mathcal{H}_1(V) = \mathcal{H}_2(V)$ for all $V \subseteq \Phi$.

Furthermore, a set of variable assignments is a set of models of a Horn sentence if and only if it is closed under intersection. We will denote the closure of a set $\mathfrak{V}$ of variable assignments under intersection by $\hat{\mathfrak{V}}$. We call a Horn

sentence $\mathcal{H}$ a *Horn envelope* for a set of assignments $\mathfrak{V}$ if $\hat{\mathfrak{V}}$ is precisely the set of models of $\mathcal{H}$; note that, in this case, $\hat{\mathfrak{V}} = \{V \subseteq \Phi \mid V = \mathcal{H}(V)\}$.

A set of variable assignments may have several equivalent envelopes. Of special interest are envelopes that are minimal in the number of implications. One particular minimal envelope is known from formal concept analysis [2] under the name of the Duquenne–Guigues or canonical basis of implications [6], which we define next. A variable assignment V is called *pseudo-closed* with respect to a closure operator $\mathcal{H}(\cdot)$ if

1. $V \neq \mathcal{H}(V)$;
2. $\mathcal{H}(W) \subsetneq V$ for every pseudo-closed $W \subsetneq V$.

Note that, according to this definition, every variable assignment minimal among those that are not closed is pseudo-closed.

The *Duquenne–Guigues basis* or *canonical basis* of a Horn sentence $\mathcal{H}$ is the following Horn sentence:

$$\bigwedge \{P \to \mathcal{H}(P) \mid P \text{ is pseudo-closed with respect to } \mathcal{H}(\cdot)\}. \tag{1}$$

If $\mathcal{H}$ is a Horn envelope of $\mathfrak{V}$, we also say that (1) is the Duquenne–Guigues basis of $\mathfrak{V}$.

The problem of learning Horn envelopes frequently occurs in various settings, in particular, in data analysis, where Horn sentences are often used to summarize interdependencies between attributes in data. In this context, the data is given by a set $\mathfrak{V}$ of variable assignments and the task is to find its Horn envelope, i.e., a basis of implications valid in the data. However, the size of the Horn envelope $\hat{\mathcal{H}}$ of $\mathfrak{V}$ can be exponential in the size of $\mathfrak{V}$ [5]. From the computational perspective, one could hope for an algorithm that runs in polynomial total time [7], that is, an algorithm polynomial in the size of input and output, i.e., in $|\Phi|$, $|\mathfrak{V}|$, and $|\hat{\mathcal{H}}|$, but no such algorithm is known yet [8]. For this reason, it may be useful to compute Horn envelopes approximately.

Let $\hat{\mathcal{H}}$ be a Horn envelope of $\mathfrak{V}$, i.e., $\operatorname{Mod} \hat{\mathcal{H}} = \hat{\mathfrak{V}}$. We call a Horn sentence $\mathcal{H}$ an *ε-Horn approximation* of $\mathfrak{V}$ if

$$\frac{|\operatorname{Mod} \mathcal{H} \mathbin{\triangle} \operatorname{Mod} \hat{\mathcal{H}}|}{2^{|\Phi|}} \leq \varepsilon, \tag{2}$$

where $A \mathbin{\triangle} B$ is the symmetric difference between sets A and B. This is the notion of approximation used in [5], where a probabilistic algorithm to compute such an approximation from a set of models in total polynomial time is presented. However, this notion of approximation may be too weak for practical purposes: achieving an ε-Horn approximation of $\mathfrak{V}$ is very easy if $\hat{\mathfrak{V}}$ is small relative to $2^{|\Phi|}$, which is often the case. Since many real-world datasets are sparse, the size of $\hat{\mathfrak{V}}$ is often exponentially smaller than $2^{|\Phi|}$. Then setting $\mathcal{H} = \{\varnothing \to \bot\}$ results in $\operatorname{Mod} \mathcal{H} = \varnothing$, and the error

$$\frac{|\operatorname{Mod} \mathcal{H} \mathbin{\triangle} \operatorname{Mod} \hat{\mathcal{H}}|}{2^{|\Phi|}} = \frac{|\operatorname{Mod} \hat{\mathcal{H}}|}{2^{|\Phi|}}$$

is exponentially small.

Therefore, we will also use a stronger notion of approximation introduced in [9]. We call $\mathcal{H}$ an *ε-strong Horn approximation* of $\mathfrak{V}$ if

$$\frac{|\{V \subseteq \Phi \mid \mathcal{H}(V) \neq \hat{\mathcal{H}}(V)\}|}{2^{|\Phi|}} \leq \varepsilon, \tag{3}$$

where $\hat{\mathcal{H}}$ is a Horn envelope of $\mathfrak{V}$. It is easy to see that an ε-strong Horn approximation of $\mathfrak{V}$ is always an ε-Horn approximation of $\mathfrak{V}$, but the reverse is not true.

Example 1. Consider $\Phi = \{a, b, c, d\}$ and $\mathfrak{V} = \{\{a, c, d\}, \{a, c\}, \{b, c\}, \{b, d\}\}$. The Horn envelope of $\mathfrak{V}$ is given by

$$\hat{\mathcal{H}} = (c \wedge d \to a) \wedge (a \to c) \wedge (a \wedge b \wedge c \to \bot).$$

In addition to the four sets from $\mathfrak{V}$, $\operatorname{Mod} \hat{\mathcal{H}}$ includes four other sets: $\varnothing$, $\{b\}$, $\{c\}$, and $\{d\}$. The Horn sentence

$$\mathcal{H} = (c \wedge d \to a) \wedge (a \to \bot)$$

is a 0.125-Horn approximation of $\mathfrak{V}$, since it disagrees with $\hat{\mathcal{H}}$ on exactly two out of sixteen subsets of Φ: $\{a, c\}$ and $\{a, c, d\}$ are models of $\hat{\mathcal{H}}$, but not of $\mathcal{H}$.

However, $\mathcal{H}$ is not even a 0.3-strong Horn approximation, because $\mathcal{H}(V)$ differs from $\hat{\mathcal{H}}(V)$ for five out of sixteen subsets V of Φ:

$$\begin{aligned}
\mathcal{H}(\{a\}) = \bot &\neq \{a, c\} = \hat{\mathcal{H}}(\{a\}) \\
\mathcal{H}(\{a, c\}) = \bot &\neq \{a, c\} = \hat{\mathcal{H}}(\{a, c\}) \\
\mathcal{H}(\{a, d\}) = \bot &\neq \{a, c, d\} = \hat{\mathcal{H}}(\{a, d\}) \\
\mathcal{H}(\{c, d\}) = \bot &\neq \{a, c, d\} = \hat{\mathcal{H}}(\{c, d\}) \\
\mathcal{H}(\{a, c, d\}) = \bot &\neq \{a, c, d\} = \hat{\mathcal{H}}(\{a, c, d\})
\end{aligned}$$

3. Learning Horn Sentences with Equivalence and Membership Queries

In this paper, we consider the problem of learning Horn approximations via queries. In the query learning framework, rather than learning from a training dataset, the learning algorithm has access to an oracle (or an expert), which it can address with certain predefined types of questions [10]. Probably, the most typical are equivalence and membership queries. In a *membership query*, the learner asks whether a certain instance is an example of the concept being learned. For the problem of learning Horn sentences, the membership query allows the learning algorithm to find out whether a particular variable assignment is a model of the target Horn sentence. An *equivalence query* is parameterized with a hypothesis describing the concept being learned. If the hypothesis matches the concept, the answer is positive and learning may be terminated. Otherwise, the oracle must provide a counterexample covered by the hypothesis, but not by the target concept (*negative counterexample*), or vice

versa (*positive counterexample*). In our case, the target concept and hypotheses are Horn sentences and a counterexample is a variable assignment satisfying exactly one of these two sentences.

An algorithm for learning Horn sentences with equivalence and membership queries is described in [1], where it is proved that it requires time polynomial in the number of variables, n, and the number of clauses, m, of the target Horn sentence; $O(mn)$ equivalence queries and $O(m^2n)$ membership queries are made in the process. In the version of the algorithm we present here, the algorithm maintains a hypothesis $\mathcal{H}$ as a list of implications of the form $A \to B$, where $A \subseteq B \subseteq \Phi \cup \{\bot\}$. The algorithm starts with the empty hypothesis, which is compatible with every possible assignment, and proceeds until a positive answer is obtained from the equivalence query. If a negative counterexample X is received instead, the algorithm uses membership queries to find the first implication $A \to B$ in the current hypothesis $\mathcal{H}$ such that $A \cap X \neq A$ is not a model of the target Horn sentence. If such an implication is found, the implication $A \to B$ is replaced by $A \cap X \to B$, which ensures that X is no longer a model of $\mathcal{H}$. When a positive counterexample X is obtained from an equivalence query, every implication $A \to B$ of which X is not a model is replaced by $A \to B \cap X$ (recall that we identify $\bot$ with $\Phi \cup \{\bot\}$). We give pseudocode in Algorithm 1 and refer the reader to [1] for further details.

Algorithm 1 HORN1($equivalence(\cdot), member(\cdot)$)

Input: An equivalence and a membership oracles for a Horn sentence $\mathcal{H}_*$.
Output: The Duquenne–Guigues basis of $\mathcal{H}_*$ (as a set of implications).

```
1:  H := []                                                    {empty list}
2:  while equivalent(H) returns a counterexample X do
3:      if X ⊨ H then                                  {negative counterexample}
4:          found := false
5:          for all A → B ∈ H do
6:              C := A ∩ X
7:              if A ≠ C and not member(C) then
8:                  replace A → B by C → B in H
9:                  found := true
10:                 exit for
11:         if not found then
12:             add X → ⊥ to the end of H
13:     else                                           {positive counterexample}
14:         for all A → B ∈ H such that X ⊭ A → B do
15:             replace A → B
16:             by A → B ∩ X in H    {If B = ⊥, assume that B = Φ ∪ {⊥}}
```

Table 1 shows one possible execution of Algorithm 1 when learning the formula $(c \wedge d \to a) \wedge (a \to c) \wedge (a \wedge b \wedge c \to \bot)$.

In [11], it is shown that Algorithm 1 always produces the Duquenne–Guigues basis of the target Horn sentence no matter what examples are received from

$\mathcal{H}$	X	type
[]	$\{c, d\}$	negative
$[\{c, d\} \to \bot]$	$\{a, c, d\}$	positive
$[\{c, d\} \to \{a\}]$	$\{a, b\}$	negative
$[\{c, d\} \to \{a\}, \quad \{a, b\} \to \bot]$	$\{a\}$	negative
$[\{c, d\} \to \{a\}, \quad \{a\} \to \bot]$	$\{a, c\}$	positive
$[\{c, d\} \to \{a\}, \quad \{a\} \to \{c\}]$	$\{a, b, c\}$	negative
$[\{c, d\} \to \{a\}, \quad \{a\} \to \{c\}, \quad \{a, b, c\} \to \bot]$		

Table 1: A possible HORN1 learning protocol for $\mathcal{H}_* = (c \wedge d \to a) \wedge (a \to c) \wedge (a \wedge b \wedge c \to \bot)$

the equivalence queries.

However, this algorithm has limitations in terms of applications we have in mind. In what situations query-based learning can be useful? One scenario is when there is not enough data about the domain under consideration, but there are domain experts willing to share their knowledge about the domain. We can use queries to extract information from them. Another scenario is when there is a huge amount of data, more than can be handled by standard algorithms for mining dependencies, and this data is organized in a distributed database or is spread over the Internet; however, there are mechanisms for efficiently querying the data. Query-based learning may also be useful if we work with a mathematical domain, one with an infinite number of objects, and there are procedures that can automatically prove theorems about the domain or generate counterexamples from this domain to our hypotheses; such procedures can be used as oracles, and we only need to ask them the right questions.

Unfortunately, it is not easy to use Algorithm 1 to learn valid implications in such situations. One problem is that the algorithm needs negative counterexamples. These counterexamples are not part of the domain, they are propositional combinations that never occur. It is unreasonable to expect from a human expert to be able to easily produce such combinations. A computer program can search a database or the Internet for a positive counterexample to a hypothesis, but it is more difficult to find something that does not exist. It may not always be easy to construct a mathematical object that violates a certain conjecture, but it seems much more difficult to construct a description of a non-existing object that satisfies the conjecture.

There is a more fundamental problem with applying Algorithm 1 in our setting: the oracles in Algorithm 1 must answer queries relative to the Horn formula being learnt. In our case, we work with an arbitrary domain and want to compute its Horn envelope; we assume that the oracle answers queries relative to the domain and not to its Horn envelope. If our domain is not Horn, i.e., its set of models $\mathfrak{V}$ is not closed under intersection, then the set $\hat{\mathfrak{V}}$ of models of its Horn envelope is different from $\mathfrak{V}$. Therefore, we will not receive a positive answer to an equivalence query even if we compute the envelope precisely; instead, we

will obtain a negative counterexample from $\hat{\mathfrak{V}} \setminus \mathfrak{V}$. A similar problem occurs with membership queries: to be able to use Algorithm 1, we need the oracle to answer membership queries relative to $\hat{\mathfrak{V}}$, rather than to $\mathfrak{V}$.

4. Learning Horn Envelopes of Arbitrary Domains

A solution is offered by formal concept analysis in the form of a procedure called *attribute exploration* [2, 4]. Instead of membership and equivalence queries, it uses what we will call *implication queries*, i.e., queries of the form "Is it true that $\mathfrak{V} \models A \to B$?" for $A, B \subseteq \Phi \cup \{\bot\}$. The oracle, or domain expert, answers positively in case the entailment holds or provides a *positive counterexample* $X \in \mathfrak{V}$ such that $X \not\models A \to B$. In terms of [10], implication queries are a special case of *superset queries*: asking whether $\mathfrak{V} \models A \to B$ amounts to asking whether the set of models of $A \to B$ is a superset of $\mathfrak{V}$.

The algorithm only asks about the validity of implications that do not follow from those already confirmed by the expert and that do not contradict examples provided by the expert. Upon termination of the algorithm, the set of confirmed implications is the canonical basis of $\mathfrak{V}$. Moreover, the set $\mathfrak{V}'$ of all models returned by the expert can be considerably smaller than $\mathfrak{V}$, but it has the same Horn envelope $\hat{\mathcal{H}}$. The downside is that the number of queries may be exponential in $\hat{\mathcal{H}}$, since $\mathfrak{V}'$ must contain all models of $\hat{\mathcal{H}}$ that cannot be represented as the intersection of other models of $\hat{\mathcal{H}}$; these are called *characteristic* models of $\hat{\mathcal{H}}$, and their number can be exponential in $|\hat{\mathcal{H}}|$ [5]. Also, while deciding what queries must be posed, the algorithm implicitly enumerates all models in $\mathfrak{V}$. Because of this, the time between two queries to the domain expert can be exponential in the number of variables $|\Phi|$.

In the following, we present a modification of Algorithm 1 that simulates membership queries relative to $\hat{\mathfrak{V}}$ by implication queries relative to $\mathfrak{V}$, the same queries as those used in attribute exploration. It also replaces equivalence queries by a call to a stochastic procedure, which makes it possible to compute an ε-Horn approximation of $\mathfrak{V}$ with the desired probability δ. We will then modify this algorithm so that it produces an ε-strong Horn approximation of $\mathfrak{V}$. The resulting algorithms can be considered as PAC versions of attribute exploration.

4.1. Simulating Membership Queries

Let $\hat{\mathcal{H}}$ be a Horn envelope of a set $\mathfrak{V} \subseteq 2^\Phi$. For computing $\hat{\mathcal{H}}$, we need the membership query be answered relative to $\hat{\mathfrak{V}}$. Such a query can be simulated by several implication queries relative to $\mathfrak{V}$. One well-known (see, e.g., [12]) method to do this is presented in Theorem 1.

Theorem 1. *Let Φ be a set of variables, $A \subsetneq \Phi$, and $\mathfrak{V} \subseteq 2^\Phi$ be an arbitrary set of variable assignments. Then $A \in \hat{\mathfrak{V}}$ if and only if $\mathfrak{V} \models A \to \{a\}$ for no $a \in \Phi \setminus A$.*

Proof. If $\mathfrak{V} \models A \to \{a\}$ for some $a \in \Phi \setminus A$, then every assignment from $\mathfrak{V}$ that includes A as a subset must contain a. But then, since $a \notin A$, the set A is not in $\mathfrak{V}$ and it cannot be an intersection of assignments from $\mathfrak{V}$; i.e., $A \notin \hat{\mathfrak{V}}$.

Conversely, if $\mathfrak{V} \models A \to \{a\}$ for no $a \in \Phi \setminus A$, then, for every $a \in \Phi \setminus A$, there is $B \in \mathfrak{V}$ such that $A \subseteq B$, but $a \notin B$. Hence, A is the intersection of all $B \in \mathfrak{V}$ such that $A \subseteq B$; i.e., $A \in \hat{\mathfrak{V}}$. □

Theorem 1 makes it possible to check membership in $\hat{\mathfrak{V}}$ using at most $|\Phi|$ implication queries for every proper subset of Φ. To check if $A \in \hat{\mathfrak{V}}$ for $A = \Phi$, one query $A \to \bot$? is sufficient. Of course, for any subset A of Φ, a positive answer to such a query means that $A \notin \hat{\mathfrak{V}}$. This reasoning leads to Algorithm 2.

Algorithm 2 IsMember(A, $is\_valid(\cdot)$)

Input: A set $A \subseteq \Phi$ and an implication oracle $is\_valid(\cdot)$ for some $\mathfrak{V} \subseteq 2^{\Phi}$.
Output: **true** if $A \in \hat{\mathfrak{V}}$ and **false** otherwise.

```
1: if is_valid(A → ⊥) then
2:     return false
3: for all a ∈ Φ \ A do
4:     if is_valid(A → {a}) then
5:         return false
6: return true
```

Note that, in this simulation, we do not use counterexamples provided by the implication oracle. Following [10], we will call implication queries that do not return counterexamples *restricted*. Thus, a membership query relative to $\hat{\mathfrak{V}}$ can be simulated by a linear (in $|\Phi|$) number of restricted implication queries relative to $\mathfrak{V}$. Since essentially all the algorithm does is posing queries and every next query can be obtained from the previous one in constant time, it is obvious that the time complexity of Algorithm 2 is $O(|\Phi|)$ (of course, not including the time the oracle might need to answer the queries).

4.2. Simulating Equivalence Queries

We replace every equivalence query by sampling a number of variable assignments and checking whether any of them is a positive or negative counterexample. This technique, proposed in [10], makes it possible to obtain a polynomial-time PAC algorithm from a polynomial-time exact learning algorithm that uses equivalence queries. A similar strategy is used in [5] to obtain a PAC algorithm computing an ε-Horn approximation of an explicitly given set of models. In our case, the difference is that we use this technique to transform an exact algorithm for learning a Horn theory with the membership oracle w.r.t. this theory into an algorithm for learning the Horn envelope of an arbitrary theory with the implication oracle w.r.t. this arbitrary theory.

In our algorithm, we sample $\left\lceil \frac{1}{\varepsilon} \cdot \left(i + \ln \frac{1}{\delta}\right) \right\rceil$ variable assignments to simulate the ith equivalence query asked by the algorithm. For each generated assignment X, we check if X satisfies our hypothesis $\mathcal{H}$ and, using Algorithm 2, if $X \in \hat{\mathfrak{V}}$. If the answers to these questions are different, then X is a counterexample to $\mathcal{H}$. If none of the generated assignments is a counterexample, the algorithm concludes that $\mathcal{H}$ is an ε-approximation of $\mathfrak{V}$. We present the sampling procedure

in Algorithm 3 and the procedure that computes an ε-Horn approximation in Algorithm 4.

Algorithm 3 IsApproximatelyEquivalent($\mathcal{H}$, *is_valid*$(\cdot)$, ε, δ, i)

Input: A set $\mathcal{H}$ of implications over a set Φ of propositional variables, an implication oracle *is_valid*$(\cdot)$ for some $\mathfrak{V} \subseteq 2^\Phi$, $0 < \varepsilon \leq 1$, $0 < \delta \leq 1$, and $i \in \mathbb{N}$.

Output: A counterexample to $\mathcal{H}$ relative to $\hat{\mathfrak{V}}$ if found; **true**, otherwise.

```
for j := 1 to ⌈(1/ε) · (i + ln(1/δ))⌉ do
    generate X ⊆ M uniformly at random
    if (X ⊨ H) ≢ IsMember(X, is_valid(·)) then
        return X
return true
```

Theorem 2. *Let $\mathfrak{V} \subseteq 2^\Phi$ be an arbitrary set of variable assignments and $\hat{\mathcal{H}}$ be its Horn envelope. Given a (restricted) implication oracle for $\mathfrak{V}$, $0 < \varepsilon \leq 1$, and $0 < \delta \leq 1$ as input, Algorithm 4 computes an implication set $\mathcal{H}$ that, with probability at least $1 - \delta$, is an ε-Horn approximation of $\mathfrak{V}$. This algorithm runs in time polynomial in $|\Phi|$, $|\hat{\mathcal{H}}|$, $1/\varepsilon$, and $1/\delta$.*

Proof. As shown in [1], Algorithm 1 requires a number of counterexamples polynomial in $|\Phi|$ and $|\hat{\mathcal{H}}|$ no matter what counterexamples it receives. Suppose that this number is at most k. Since the only difference between Algorithm 1 and Algorithm 4 is how queries get answered, the upper bound k on the number of counterexamples will work for Algorithm 4, too. We will make sure that the probability of failing to find a counterexample for the ith equivalence query using Algorithm 3 is at most $\delta_i = \delta/2^i$. Then the probability of failing to find a counterexample for any of at most k equivalence queries is bounded above by

$$\frac{\delta}{2}+\left(1-\frac{\delta}{2}\right)\left(\frac{\delta}{4}+\left(1-\frac{\delta}{4}\right)\left(\frac{\delta}{8}+\left(1-\frac{\delta}{8}\right)\left(\dots\left(\frac{\delta}{2^{k-1}}+\left(1-\frac{\delta}{2^{k-1}}\right)\frac{\delta}{2^k}\right)\dots\right)\right)\right) \leq$$

$$\leq \frac{\delta}{2}+\frac{\delta}{4}+\frac{\delta}{8}+\dots+\frac{\delta}{2^k} < \delta.$$

Let us assume that, at some point of the algorithm,

$$\frac{|\operatorname{Mod}\mathcal{H} \mathbin{\triangle} \operatorname{Mod}\hat{\mathcal{H}}|}{2^{|\Phi|}} > \varepsilon.$$

If this is not the case, then $\mathcal{H}$ is already an ε-approximation of $\mathfrak{V}$, and it is safe to terminate the algorithm. Under this assumption, if we choose X randomly, we have $X \in \operatorname{Mod}\mathcal{H} \mathbin{\triangle} \operatorname{Mod}\hat{\mathcal{H}}$ with probability of at least ε. We check if this is the case with the help of Algorithm 2. If $X \in \operatorname{Mod}\mathcal{H} \mathbin{\triangle} \operatorname{Mod}\hat{\mathcal{H}}$, we use it as a counterexample to the equivalence query and proceed as in Algorithm 1. Otherwise, we generate another X. We make at most l attempts at generating X; if we do not obtain a counterexample, we output $\mathcal{H}$ and terminate.

Algorithm 4 HornApproximation($is\_valid(\cdot)$, ε, δ)

Input: An implication oracle $is\_valid(\cdot)$ for some $\mathfrak{V} \subseteq 2^{\Phi}$, $0 < \varepsilon \leq 1$, and $0 < \delta \leq 1$.

Output: A set of implications $\mathcal{H}$ that, with probability at least $1 - \delta$, is an ε-Horn approximation of $\mathfrak{V}$.

```
1:  H := []
2:  i := 1
3:  while IsApproximatelyEquivalent(H, is_valid(·), ε, δ, i) returns a
    counterexample X do
4:      if X ⊨ H then                                  {negative counterexample}
5:          found := false
6:          for all A → B ∈ H do
7:              C := A ∩ X
8:              if A ≠ C and not IsMember(C) then
9:                  replace A → B by C → B in H
10:                 found := true
11:                 exit for
12:         if not found then
13:             add X → ⊥ to the end of H
14:     else                                           {positive counterexample}
15:         for all A → B ∈ H such that X ⊭ A → B do
16:             replace A → B
17:             by A → B ∩ X in H   {If B = ⊥, assume that B = Φ ∪ {⊥}}
18:     i := i + 1
```

The probability that we fail to find a counterexample in l trials is smaller than δ_i if

$$l > \frac{1}{\varepsilon} \cdot \ln \frac{1}{\delta_i}. \tag{4}$$

Indeed, the probability of failure is less than $(1-\varepsilon)^l$. For this to be less than δ_i, we need

$$l > \log_{1-\varepsilon} \delta_i = \frac{\ln \delta_i}{\ln(1-\varepsilon)} = \frac{\ln(1/\delta_i)}{-\ln(1-\varepsilon)}.$$

Since $-\ln(1-\varepsilon) > \varepsilon$, it suffices to choose any l satisfying (4) to make the probability of failure less than δ_i. In particular, we can set

$$l = \left\lceil \frac{1}{\varepsilon} \cdot \ln \frac{1}{\delta_i} \right\rceil = \left\lceil \frac{1}{\varepsilon} \cdot \ln \frac{2^i}{\delta} \right\rceil \leq \left\lceil \frac{1}{\varepsilon} \cdot \left(i + \ln \frac{1}{\delta} \right) \right\rceil \leq \left\lceil \frac{1}{\varepsilon} \cdot \left(\mathit{poly}(|\Phi|, |\hat{\mathcal{H}}|) + \ln \frac{1}{\delta} \right) \right\rceil,$$

where $\mathit{poly}(|\Phi|, |\hat{\mathcal{H}}|)$ is some polynomial that upper-bounds k, the number of counterexamples required by Algorithm 1 for the target Horn sentence $\hat{\mathcal{H}}$.

To sum up, Algorithm 1 runs in time polynomial in $|\Phi|$ and the number of implications in the target Horn sentence $\hat{\mathcal{H}}$. We simulate this algorithm, but replace each equivalence query by a number of attempts polynomial in $|\Phi|$, $|\hat{\mathcal{H}}|$, $1/\varepsilon$, and $1/\delta$ at generating a counterexample to the current hypothesis $\mathcal{H}$. Each such attempt requires time $\mathit{poly}(\Phi, |\hat{\mathcal{H}}|)$, in particular, since the algorithm guarantees that $|\mathcal{H}| \leq |\hat{\mathcal{H}}|$. Therefore, our simulation runs in time polynomial in $|\Phi|$, $|\hat{\mathcal{H}}|$, $1/\varepsilon$, and $1/\delta$ and, as argued above, produces an ε-Horn approximation of $\mathfrak{V}$ with probability at least $1-\delta$. □

This result, in another form, is known from literature [1]. We include it, on the one hand, to show that it also holds with the implication oracle and, on the other hand, in order to present a comprehensive exposition.

4.3. Strong Approximations

The algorithm we have just presented can be modified to compute ε-strong Horn approximations. We only need to modify the way counterexamples are generated by the IsApproximatelyEquivalent procedure.

If

$$\frac{|\{V \subseteq \Phi \mid \mathcal{H}(V) \neq \hat{\mathcal{H}}(V)\}|}{2^{|\Phi|}} > \varepsilon, \tag{5}$$

then, by generating X uniformly at random, we obtain X such that $\mathcal{H}(X) \neq \hat{\mathcal{H}}(X)$ with probability at least ε. Suppose that we have generated such an X. The problem is that this X is not necessarily a counterexample in the sense required by the algorithm, because it may happen that it belongs neither to $\operatorname{Mod} \mathcal{H}$ nor to $\hat{\mathfrak{V}}$. It turns out that we can use X to manufacture a counterexample in time polynomial in $|\Phi|$.

Theorem 3. *Let $\hat{\mathcal{H}}$ be the Horn envelope of $\mathfrak{V} \subseteq 2^\Phi$ and $\mathcal{H}$ be a Horn formula over Φ. Then $\mathcal{H}(X) = \hat{\mathcal{H}}(X)$ if and only if $\mathcal{H}(X) \in \hat{\mathfrak{V}} \cup \{\bot\}$ and $\mathfrak{V} \models X \to \mathcal{H}(X)$.*

Proof. Suppose that $\mathcal{H}(X) = \hat{\mathcal{H}}(X) \neq \bot$. Then $\hat{\mathcal{H}}(X) \in \hat{\mathfrak{V}}$ and $\mathfrak{V} \models X \to \hat{\mathcal{H}}(X)$, and we also have $\mathcal{H}(X) \in \hat{\mathfrak{V}}$ and $\mathfrak{V} \models X \to \mathcal{H}(X)$. If, on the other hand, $\mathcal{H}(X) = \hat{\mathcal{H}}(X) = \bot$, then X is a subset of no model in $\mathfrak{V}$ and $\mathfrak{V} \models X \to \bot$.

Conversely, if $\mathfrak{V} \models X \to \mathcal{H}(X)$, then $\mathcal{H}(X) \subseteq \hat{\mathcal{H}}(X)$; and, if $\mathcal{H}(X) \in \hat{\mathfrak{V}}$, then $\hat{\mathcal{H}}(X)$, the minimal superset of X from $\hat{\mathfrak{V}}$, must be a subset of $\mathcal{H}(X)$, i.e., $\hat{\mathcal{H}}(X) \subseteq \mathcal{H}(X)$. The latter must also hold if $\mathcal{H}(X) = \bot$. □

To obtain a counterexample from a randomly generated X, we first compute $\mathcal{H}(X)$ and query the oracle to verify the implication $X \to \mathcal{H}(X)$. If the implication is invalid, the oracle will return a positive counterexample C. Otherwise, we check if $\mathcal{H}(X) \in \hat{\mathfrak{V}}$ using the IsMember procedure. If the outcome is negative, then $\mathcal{H}(X)$ is a negative counterexample (provided $\mathcal{H}(X) \neq \bot$); else, from Theorem 3, we know that $\mathcal{H}(X) = \hat{\mathcal{H}}(X)$, and we generate another X unless we have reached the maximum number of iterations. Algorithm 5 gives the pseudocode.

Thus, given (5), the probability of finding a counterexample at one iteration of Algorithm 5 is greater than ε, and the same reasoning as in Section 4.2 leads to the following theorem.

Theorem 4. *Let $\mathfrak{V} \subseteq 2^\Phi$ be an arbitrary set of variable assignments and $\hat{\mathcal{H}}$ be its Horn envelope. Given an implication oracle for $\mathfrak{V}$, $0 < \varepsilon \leq 1$, and $0 < \delta \leq 1$ as input and using Algorithm 5 as the* IsApproximatelyEquivalent *procedure, Algorithm 4 computes an implication set $\mathcal{H}$ that, with probability at least $1 - \delta$, is an ε-strong Horn approximation of $\mathfrak{V}$. This algorithm runs in time polynomial in $|\Phi|$, $|\hat{\mathcal{H}}|$, $1/\varepsilon$, and $1/\delta$.*

Algorithm 5 IsStronglyApproximatelyEquivalent($\mathcal{H}$, *is_valid*$(\cdot)$, ε, δ, i)

Input: A Horn formula $\mathcal{H}$ over a set of propositional variables Φ, an implication oracle *is_valid*$(\cdot)$ for some $\mathfrak{V} \subseteq 2^\Phi$, $0 < \varepsilon \leq 1$, $0 < \delta \leq 1$, and $i \in \mathbb{N}$.

Output: A counterexample to $\mathcal{H}$ with respect to $\hat{\mathfrak{V}}$ if found; **true**, otherwise.

for $j := 1$ **to** $\left\lceil \frac{1}{\varepsilon} \cdot \left(i + \ln \frac{1}{\delta}\right) \right\rceil$ **do**
  generate $X \subseteq M$ uniformly at random
  $Y := \mathcal{H}(X)$
  if $Y \neq X$ **and** *is_valid*$(X \to Y)$ returns a counterexample C **then**
    return C {C is a positive counterexample}
  if $Y \neq \bot$ **and not** IsMember(Y, *is_valid*$(\cdot)$) **then**
    return Y {Y is a negative counterexample}
return true

Example 2. As in Example 1, consider learning the Horn envelope of $\mathfrak{V} = \{\{a, c, d\}, \{a, c\}, \{b, c\}, \{b, d\}\}$. Fix $\varepsilon = 0.25$ and $\delta = e^{-1} \approx 0.37$. Suppose that the learning process happens to follow the protocol in Table 1, and after five iterations of the **while** loop of Algorithm 4, we obtain

$$\mathcal{H} = (c \wedge d \to a) \wedge (a \to \bot).$$

As discussed in Example 1, our goal is achieved at this point if we aim at a 0.25-Horn approximation, but it is not achieved if we aim at a 0.25-strong Horn approximation. In any case, Algorithm 4 continues with $i = 6$, and it has $1/0.25 \cdot (6 + \ln e) = 28$ chances to generate a counterexample when simulating the equivalence oracle using Algorithm 3 or 5.[1]

Suppose that the first set X generated by Algorithm 3 is $\{d\}$. In this case, we have $X \models \mathcal{H}$ and IsMember(X, $is\_valid(\cdot)$) returns **true**. Therefore, the algorithm generates another X, for example, $X = \{a, b, c, d\}$. Now, $X \not\models \mathcal{H}$ because of $\{a\} \to \bot \in \mathcal{H}$ and IsMember(X, $is\_valid(\cdot)$) returns **false**, since $\mathfrak{V} \models \{a, b, c, d\} \to \bot$. After a few more iterations, we could get $X = \{a, d\}$, which wouldn't result in a counterexample either, because $\{a, d\} \not\models \{a\} \to \bot$ and $\{a, d\} \to \{c\}$ is a valid implication forcing IsMember($\{a, d\}$, $is\_valid(\cdot)$) to return **false**. As shown in Example 1, there are only two sets that could qualify as counterexamples: $\{a, c\}$ and $\{a, c, d\}$. Table 1 shows how learning would proceed if we manage to generate $\{a, c\}$.

What changes if we use Algorithm 5 instead? As in the case of Algorithm 3, generating $X = \{d\}$ would not lead to a counterexample, since $\mathcal{H}(\{d\}) = \{d\}$, and, therefore, $\{d\}$ is processed in the same way by both algorithms. For $X = \{a, b, c, d\}$, we have $\mathcal{H}(X) = \bot$. Since $\mathfrak{V} \models \{a, b, c, d\} \to \bot$, the algorithm will not generate a counterexample from $\{a, b, c, d\}$. However, $\mathcal{H}(\{a, d\}) = \bot$, but $\{a, d\} \to \bot$ is not a valid implication. Hence, for $X = \{a, d\}$, the call to $is\_valid(\{a, d\} \to \bot)$ in Algorithm 5 will return a positive counterexample, $\{a, c\}$ or $\{a, c, d\}$, which will be propagated to the **while** loop of Algorithm 4.

It should be clear that, in general, Algorithm 5 has a better chance of generating a counterexample than Algorithm 3 and can do it faster. In the particular case described above, only two of sixteen subsets of Φ are counterexamples, and Algorithm 3 must stumble upon one of them while randomly generating subsets of Φ. Algorithm 5, on the other hand, can produce a counterexample from any of the five subsets listed in the end of Example 1.

4.4. Variations and Optimizations

The algorithm can be modified so that its current hypothesis $\mathcal{H}$ is always such that $\mathfrak{V} \models \mathcal{H}$. To ensure this, we need to take some care when replacing and adding implications in lines 9 and 13 of Algorithm 4. In particular, instead of adding implication $X \to \bot$, we should check via an implication query whether it is valid, and, if not, add instead implication $X \to \hat{\mathcal{H}}(X)$ by computing $\hat{\mathcal{H}}(X)$, again, using implication queries. One way to do this is to query about the validity of implications of the form $X \to \{a\}$ for $a \in \Phi \setminus X$: those a for which the answer is positive belong to $\hat{\mathcal{H}}(X)$. Similarly, we can replace $A \to B$ by $C \to \hat{\mathcal{H}}(C)$ in line 9. With these modifications, our sampling procedure that replaces the equivalence oracle will return only negative counterexamples, and thus the part of Algorithm 4 dealing with positive counterexamples can be eliminated.

[1] This is, of course, a toy example: in this particular case, an exhaustive search through the sixteen subsets of $\Phi = \{a, b, c, d\}$ would be more effective.

To reduce the number of queries, we can cache counterexamples returned by the oracle. All these counterexamples are models from $\mathfrak{V}$, and, therefore, they can be used to falsify some implications without resorting to the oracle: if an implication $A \to B$ has a counterexample among the models obtained so far, a query about its validity is not necessary. Since the total number of queries submitted by the algorithm is polynomial in all the quantities we care about, so is the number of counterexamples received from the oracle, and, consequently, the memory and time overhead incurred by this modification is also polynomial.

Likewise, we can cache the implications confirmed by the oracle and use them to verify the validity of some other implications when needed. It is also worth exploring whether integrating such confirmed implications into the current hypothesis may be useful.

5. Experimental Evaluation

Our algorithms come with a theoretical guarantee on the quality of approximation or, to be more precise, on the probability of attaining the desired quality. In Section 5.1, we describe quality measures *precision* and *recall*, which are slightly different from those of (2) and (3) for which the algorithms were designed. In Section 5.3, we experimentally evaluate the quality of approximations computed by Algorithm 4 in terms of these measures.

In general, the domain expert, or the oracle, used in learning is not necessarily a human being: it may well be a knowledge base equipped with a procedure capable of answering implication queries. To easily obtain domain experts for our experiments, we make use of the following approach. Starting from a data set $\mathfrak{V}$, we simulate a domain expert for $\mathfrak{V}$ by confirming $A \to B$ if $\mathfrak{V} \models A \to B$. Otherwise, the expert returns a counterexample to $A \to B$ from the dataset. The datasets we use are described in Section 5.2.

5.1. Precision and Recall

Informally, precision measures how often the extracted implications infer only correct knowledge from a given variable assignment. Conversely, recall measures how often the knowledge inferred from a variable assignment is complete.

More formally, let Φ be a finite set, let $\mathfrak{V}$ be a set of variable assignments over Φ, $\hat{\mathcal{H}}$ be its Horn envelope, and $\mathcal{H}$ be a set of implications. Then the *precision* and *recall* of $\mathcal{H}$ with respect to $\mathfrak{V}$ are defined by

$$\operatorname{prec}_{\mathfrak{V}}(\mathcal{H}) \coloneqq \frac{|\{A \subseteq \Phi \mid \mathfrak{V} \models A \to \mathcal{H}(A)\}|}{2^{|\Phi|}},$$

$$\operatorname{recall}_{\mathfrak{V}}(\mathcal{H}) \coloneqq \frac{|\{A \subseteq \Phi \mid \mathcal{H} \models A \to \hat{\mathcal{H}}(A)\}|}{2^{|\Phi|}}.$$

One can see that precision and recall are, in a way, two sides of strong approximation as defined by (3).

Computing the exact values of precision and recall for sufficiently large sets Φ is infeasible and, for our experimental evaluation, is not necessary: a good

approximation of the values would be enough. To obtain such approximations, we sample a certain number of subsets $A \subseteq \Phi$ and count how often the corresponding condition is true. More precisely, to obtain a good approximation of $\mathrm{prec}_{\mathfrak{V}}(\mathcal{H})$ and $\mathrm{recall}_{\mathfrak{V}}(\mathcal{H})$, we randomly choose a subset $\mathcal{T} \subseteq 2^{\Phi}$ and compute

$$\mathrm{prec}^{\approx}_{\mathfrak{V}}(\mathcal{H}) \coloneqq \frac{|\{A \in \mathcal{T} \mid \mathfrak{V} \models A \to \mathcal{H}(A)\}|}{|\mathcal{T}|},$$
$$\mathrm{recall}^{\approx}_{\mathfrak{V}}(\mathcal{H}) \coloneqq \frac{|\{A \in \mathcal{T} \mid \mathcal{H} \models A \to \hat{\mathcal{H}}(A)\}|}{|\mathcal{T}|},$$

An immediate question is what size n the sample set $\mathcal{T}$ needs to have for the approximation to be a good one. Utilizing Hoeffding's inequality [13], we obtain for fixed $0 < \eta, t$ that

$$\Pr(\mathrm{prec}_{\mathfrak{V}}(\mathcal{H}) - \mathrm{prec}^{\approx}_{\mathfrak{V}}(\mathcal{H}) \geq t) < \eta,$$
$$\Pr(\mathrm{recall}_{\mathfrak{V}}(\mathcal{H}) - \mathrm{recall}^{\approx}_{\mathfrak{V}}(\mathcal{H}) \geq t) < \eta$$

for

$$n \geq \frac{1}{2t^2} \cdot \ln \frac{1}{\eta}.$$

For our experiments, we chose $\eta = 0.001$ and $t = 0.01$, resulting in $n \approx 35000$ samples.

5.2. Datasets

We utilized various datasets with various properties. All used datasets were obtained from the UCI Machine Learning Repository [14]. The particular choice for the Zoo dataset and the Breast Cancer dataset was made due to the fact that those datasets are almost Boolean, well investigated, and of moderate size, thus suiting our experiments. For comparison reasons, we also considered randomly generated datasets that were of the same size and density as the ones we use from the UCI Machine Learning Repository.

Zoo Dataset (ZD). This dataset, created by Richard Forsyth, consists of 101 animals described by 15 attributes. From these attributes, 14 are Boolean and have been used as they are. Examples include attributes *(has) feathers*, *(is) airborne*, and *(has a) backbone*. The two remaining attributes *(number of) legs* and *type* were replaced by *legs = 0*, *legs = 2*, *legs = 4*, *legs = 5*, *legs = 6*, *legs = 8*, *type = 1*, ..., *type = 6*. The models of this dataset are then the combinations of attributes occurring in it.

Breast Cancer Dataset (BC). This dataset was originally obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [15]. It consists of 699 named instances, each representing a clinical case described by nine numeric attributes such as *Uniformity of Cell Size*, *Bare Nuclei*, and *Marginal Adhesion*. Each of these attributes can have a value between one and ten, and these attributes were turned into Boolean attributes in the same way as

for the ZD dataset. Finally, one attribute classifies a clinical case as malignant or benign. The models of this dataset are again the combinations of attributes occurring in it.

Random Dataset (RD). For both the Zoo dataset and the Breast Cancer dataset, we generated ten random datasets, all with the same number of attributes, instances, as well as incidence probability. These datasets have been obtained by randomly choosing whether an instance possesses an attribute, with the same probability as for the original datasets. Note that while the process places incidences uniformly at random, the Horn envelopes of the resulting set of models do not have to be distributed uniformly, as discussed in [16].

5.3. Experimental Results

For the various datasets described above, we conducted two types of experiments. Firstly, we ran Algorithm 4 for various choices of ε and δ and computed the precision, recall, fraction of valid implications, as well as the number of computed implications. The purpose of these experiments is to investigate the quality of the approximation returned by the algorithm. Secondly, we repeated the algorithm a certain number of times and investigated the distribution of precision, recall, fraction of correct implications, and the number of implications. The purpose here is to see how much the results can vary between runs of the algorithm.

Single Runs of HornApproximation. We begin our discussion with the results for the Zoo dataset. We ran Algorithm 4 for $\varepsilon \in \{0.01, 0.1, 0.5\}$ and $\delta \in \{0.1, 0.9\}$, three times each. We chose these particular values for ε and δ so that our estimates of precision and recall differ from the true values by at most 1%, 10%, and 50% with high as well as low probability. Increasing ε further seems unreasonable for real-world applications.

We observed different outcomes for different parameter combinations, as shown in Table 2. Among the computed implications were several combining different attributes, e.g.

$$\{\mathit{airborne}, \mathit{breathes}, \mathit{venomous}\} \to \{\mathit{eggs}, \mathit{type{=}6}, \mathit{leg{=}6}, \mathit{hair}\}.$$

A complete list of implications for one run is shown in the end of this section. Our estimates of precision computed as explained in Section 5.1 were always 1 and were therefore not included in Table 2. The recall is very volatile in our experiments. Varying the ε parameter has a big impact on the size of the resulting set of implications: the smaller ε, the more implications are found. The increase in the number of learned implications when ε is decreased is expected, since, with more samples, more queries to the oracles can be stated. Indeed, choosing $\varepsilon = 1/100, 1/1000, 1/10000$ resulted in bases of sizes 24, 38, and 95, respectively. Note that the Duquenne–Guigues basis of ZD has 141 implications. On the other hand, more queries do not necessarily lead to more implications, as shown by the results in Table 2.

Name	SR_1	DP_1	BS_1	SR_2	DP_2	BS_2	SR_3	DP_3	BS_3
$ZD_{(0.01,0.1)}$	**0.91**	0.75	24	0.89	0.87	23	0.88	**0.96**	**26**
$ZD_{(0.01,0.9)}$	0.08	0.71	24	**0.90**	**0.92**	**28**	0.81	0.74	26
$ZD_{(0.1,0.1)}$	0.09	**1.00**	**17**	**0.24**	0.79	14	0.00	0.75	14
$ZD_{(0.1,0.9)}$	0.19	0.73	11	**0.75**	0.73	11	0.49	**0.73**	**15**
$ZD_{(0.5,0.1)}$	0.07	1.00	10	**0.37**	1.00	11	0.00	**1.00**	**11**
$ZD_{(0.5,0.9)}$	0.73	0.89	9	0.54	0.78	9	**0.73**	**1.00**	**11**
$BC_{(0.01,0.1)}$	1.00	0.95	39	0.99	**0.97**	38	**1.00**	0.96	**50**
$BC_{(0.01,0.9)}$	**1.00**	0.95	41	1.00	0.94	**47**	1.00	**0.98**	44
$BC_{(0.1,0.1)}$	**0.99**	**0.97**	**31**	0.93	0.96	26	0.98	0.93	29
$BC_{(0.1,0.9)}$	0.88	0.94	33	0.97	0.90	29	**0.99**	**0.97**	**35**
$BC_{(0.5,0.1)}$	0.84	1.00	22	**0.88**	**1.00**	**24**	0.67	1.00	21
$BC_{(0.5,0.9)}$	0.75	**1.00**	25	**0.91**	1.00	24	0.79	0.93	**28**

Table 2: Results for the Zoo (ZD) and Breast Cancer (BC) experiments for three independent executions of Algorithm 4 with different parameter combinations. SR = a statistical estimate of the recall, DP = the fraction of valid implications, BS = the number of computed implications.

The BC dataset has six times as many attributes as the Zoo dataset. Its Duquenne–Guigues basis consists of 10739 implications. Here an inferior recall for higher values of ε can be observed, but the precision and the fraction of correctly computed implications do not seem to be correlated with ε.

Finally, for each random dataset, we applied our algorithm and calculated the average value and the standard deviation of the size of the implication set, the fraction of correctly computed implications, and the recall. We used $\varepsilon = 0.1$ and $\delta = 0.1$. For the Zoo dataset, we obtained around 23.1 ± 3.8 implications, with a fraction of 0.84 ± 0.12 valid ones, and recall around 0.90 ± 0.05. For the Breast Cancer dataset, we obtained 24 ± 1.3 implications, the 0.94 ± 0.04 fraction of which were valid, and a recall of 0.97 ± 0.01.

The size of the set of implications dropped for the Breast Cancer dataset significantly, from about 30 to approximately 24. On the contrary, we see an increase from around 15 to 24 in the Zoo dataset. For both datasets, we can observe that the fraction of valid implications is about the same in random datasets and the Zoo and Breast Cancer datasets, respectively. However, the recall in the Breast Cancer case stays the same, whereas in the Zoo case the recall for the random dataset is considerably larger than for the original dataset. The standard deviation for both measures is considerably smaller for random datasets. We conjecture that the drop in the number of implications obtained for the Breast Cancer dataset might be attributed to the random generation process: while generating the random datasets, we did not take into account that multiple values of a numeric attribute should still exclude each other. Since Breast Cancer dataset contains many numeric attributes, this effect could be large.

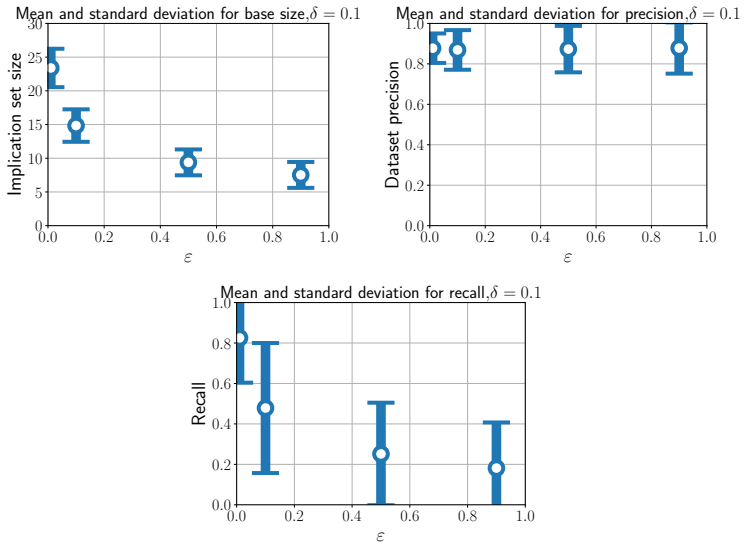

Figure 1: Stability experiment for ZD. Results with $\varepsilon \in \{0.01, 0.1, 0.5, 0.9\}$ for the size of the implication set (upper left), precision (upper right), and recall (bottom).

Repeated Runs of HornApproximation. How reliable is the computation for a particular set of parameters? Since the results in the previous section revealed a high volatility, especially for the recall measure, we wanted to check how reliable the results of the algorithm were in terms of reproducibility. For this, we applied the algorithm 1000 times to the Zoo dataset using $\varepsilon \in \{0.01, 0.1, 0.5, 0.9\}$ with $\delta = 0.1$. The results are shown in Figure 1.

For the mean of the number of implications, as well as for the mean of the recall, we observe an inverse proportionality for increasing ε. For the recall, the standard deviation is high in general and increasing with ε. In contrast, the fraction of valid implications remains stable for all considered ε with only a small increase in the standard deviation.

All plots indicate that the implications obtained by the algorithm are reliable to a certain degree with respect to multiple runs of the algorithm. The observed inverse proportionality can be explained by the number of samples drawn for a fixed ε being inverse proportional, cf. Algorithm 3.

Example Results for the Zoo Data. In Figure 2, we show the set of implications obtained by applying the PAC attribute exploration algorithm to the Zoo dataset using $\varepsilon = 0.01$ and $\delta = 0.1$. Overall, there were 24 implications, 18 of which were valid in the Zoo dataset. In this case, the approximate precision and recall were close to 1.00 and 0.92, respectively.

- $\{leg{=}5\} \to \{eggs, predator, type{=}7, aquatic\}$
- $\{tail, aquatic\} \to \{backbone\}$
- $\{hair\} \to \{breathes\}$
- $\{type{=}1\} \to \{milk, backbone, breathes\}$
- $\{type{=}3\} \to \{backbone, tail\}$
- $\{airborne\} \to \{breathes\}$
- $\{type{=}2\} \to \{eggs, feathers, catsize, leg{=}2, backbone, tail, breathes\}$ [FALSE]
- $\{type{=}4\} \to \{eggs, toothed, fins, leg{=}0, backbone, tail, aquatic\}$
- $\{milk\} \to \{type{=}1, backbone, breathes\}$
- $\{leg{=}6\} \to \{eggs, type{=}6, airborne, breathes, hair, venomous\}$ [FALSE]
- $\{domestic, catsize\} \to \{milk, predator, toothed, type{=}1, backbone, breathes, hair\}$ [FALSE]
- $\{tail, type{=}7\} \to \{predator, leg{=}8, breathes, venomous\}$
- $\{leg{=}0, breathes, hair\} \to \{milk, predator, toothed, catsize, fins, type{=}1, backbone, aquatic\}$
- $\{toothed\} \to \{backbone\}$
- $\{type{=}5\} \to \{leg{=}4, eggs, toothed, backbone, breathes, aquatic\}$
- $\{eggs, catsize, backbone\} \to \{tail, breathes\}$ [FALSE]
- $\{leg{=}2\} \to \{backbone, breathes\}$
- $\{leg{=}8\} \to \{predator, tail, type{=}7, breathes, venomous\}$ [FALSE]
- $\{leg{=}4, breathes\} \to \{backbone\}$
- $\{fins, backbone\} \to \{toothed, aquatic\}$
- $\{feathers, breathes\} \to \{type{=}2, eggs, catsize, leg{=}2, backbone, tail\}$ [FALSE]
- $\{type{=}6, backbone\} \to \bot$
- $\{leg{=}4, leg{=}2, backbone, breathes\} \to \bot$
- $\{leg{=}0, backbone, breathes\} \to \{predator, toothed\}$

Figure 2: The result of a particular run of the PAC attribute exploration with $\varepsilon = 0.01$ and $\delta = 0.1$. False implications are marked at the end by [FALSE].

6. Conclusion

In this paper, we have shown that Horn envelopes of arbitrary domains are PAC-learnable via implication queries for which the oracle must confirm that an implication $A \to B$ is valid in the domain or provide a counterexample to it. We have considered two notions of approximation of Horn envelopes, one much stronger than the other one, and described algorithms to compute both.

There are various possible next steps. One aspect is to optimize the algorithm through more effective usage of implications that the oracle confirms and counterexamples it provides. Another interesting modification of the algorithm would be to change the sampling distribution in order to reduce the number of queries or to better adapt to a domain while preserving PAC learnability. Beyond that, one may think about adapting the algorithm to learn implications that admit a certain small fraction of counterexamples (i.e., high-confident association

rules). Other possible settings include learning from error-prone experts or from multiple experts with partial or even conflicting views on the domain.

An important potential application of the algorithms presented here is completing description logic knowledge bases. This has been done with standard attribute exploration [17]; we plan to consider a similar application for its PAC versions proposed in this paper.

Finally, one may think about a *time-constraint exploration* version suitable for situations when the system has only a limited amount of time to learn implicational knowledge.

Acknowledgements

Sergei Obiedkov is supported by the Russian Science Foundation (grant 17-11-01294).

References

[1] D. Angluin, M. Frazier, L. Pitt, Learning conjunctions of Horn clauses, Machine Learning 9 (2-3) (1992) 147–164. doi:10.1007/bf00992675.
URL http://dx.doi.org/10.1007/bf00992675

[2] B. Ganter, R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer, Berlin/Heidelberg, 1999.

[3] D. J. Kavvadias, C. H. Papadimitriou, M. Sideri, On Horn envelopes and hypergraph transversals., in: K.-W. Ng, P. Raghavan, N. V. Balasubramanian, F. Y. L. Chin (Eds.), ISAAC, Vol. 762 of Lecture Notes in Computer Science, Springer, 1993, pp. 399–405.
URL http://dblp.uni-trier.de/db/conf/isaac/isaac93.html#KavvadiasPS93

[4] B. Ganter, S. Obiedkov, Conceptual Exploration, Springer, Berlin/Heidelberg, 2016.

[5] H. Kautz, M. Kearns, B. Selman, Horn approximations of empirical data, Artificial Intelligence 74 (1) (1995) 129–145. doi:10.1016/0004-3702(94)00072-9.
URL http://dx.doi.org/10.1016/0004-3702(94)00072-9

[6] J.-L. Guigues, V. Duquenne, Famille minimale d'implications informatives résultant d'un tableau de données binaires, Mathématiques et Sciences Humaines 24 (95) (1986) 5–18.

[7] D. S. Johnson, C. H. Papadimitriou, On generating all maximal independent sets, Inf. Process. Lett. 27 (3) (1988) 119–123. doi:10.1016/0020-0190(88)90065-8.
URL http://dx.doi.org/10.1016/0020-0190(88)90065-8

[8] R. Khardon, Translating between Horn representations and their characteristic models, J. Artif. Intell. Res. (JAIR) 3 (1995) 349–372.

[9] M. Babin, Models, methods, and software for dependency mining based on lattices of closed sets, Ph.D. thesis, National Research University Higher School of Economics, Moscow, in Russian (2012).

[10] D. Angluin, Queries and concept learning, Machine Learning 2 (1988) 319–342.

[11] M. Arias, J. L. Balcázar, Construction and learnability of canonical Horn formulas, Machine Learning 85 (3) (2011) 273–297.

[12] M. Arias, J. L. Balcázar, C. Tîrnăucă, Learning definite Horn formulas from closure queries, Theoretical Computer Science 658 (Part B) (2017) 346 – 356.

[13] W. Hoeffding, Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association 58 (301) (1963) pp. 13–30. URL http://www.jstor.org/stable/2282952

[14] M. Lichman, UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml

[15] O. L. Mangasarian, W. H. Wolberg, Cancer diagnosis via linear programming, SIAM News 23 (5) (1990) 1–18.

[16] D. Borchmann, T. Hanika, Some experimental results on randomly generating formal contexts., in: M. Huchard, S. Kuznetsov (Eds.), CLA, Vol. 1624 of CEUR Workshop Proceedings, CEUR-WS.org, 2016, pp. 57–69. URL http://dblp.uni-trier.de/db/conf/cla/cla2016.html#BorchmannH16

[17] F. Baader, B. Ganter, B. Sertkaya, U. Sattler, Completing description logic knowledge bases using formal concept analysis, in: M. M. Veloso (Ed.), Proceedings IJCAI'07, AAAI Press, 2007, pp. 230–235.