In Proc. 6th International Workshop on Inductive Logic Programming (ILP-96), pages 97–118 of Stockholm University Dept. of Comuputer and System Sciences Tech. report version of proceedings, August, 1996. Springer-Verlang *Lecture Notes in Computer Science* volume version of proceedings is forthcoming.

Learning Logic Programs with Random Classification Noise

Tamás Horváth[1]*, Robert H. Sloan[2]**, and György Turán[3]***

[1] Dept. of Applied Informatics, József A. University, Szeged 6720, Hungary
[2] Dept. of EE & Computer Science, University of Illinois at Chicago
851 S. Morgan St. Rm 1120, Chicago, IL 60607-7053, USA
[3] Dept. of Mathematics, Stat., & Computer Science, University of Illinois at Chicago,
Research Group on Artificial Intelligence, Hungarian Academy of Sciences

Abstract. We consider the learnability of classes of logic programs in the presence of noise, assuming that the label of each example is reversed with a fixed probability. We review the polynomial *PAC* learnability of nonrecursive, determinate, constant-depth Horn clauses in the presence of such noise. This result is extended to an analogous class of recursive logic programs that consist of a recursive clause, a base case clause, and ground background knowledge. Also, we show that arbitrary nonrecursive Horn clauses with forest background knowledge remain polynomially *PAC* learnable in the presence of noise. We point out that the sample size can be decreased by using dependencies among the literals.

1 Introduction

Learnability in the presence of noise is a major issue in machine learning. Its importance in inductive logic programming (ILP) is illustrated by the great emphasis given to noise handling mechanisms in Lavrač and Džeroski's book [16], and by numerous papers (e.g., [15, 17, 22]). Noise-tolerant heuristics often depend on probabilities estimated from the training set. Such heuristics usually work because a moderate amount of noise typically alters these estimates only slightly.

Computational learning theory has also long recognized that noise is of central importance for learnability. Several formal models have been developed based on different assumptions about the noise occurring in the data. In this paper, we consider the *random classification* noise model [1]: each labeled example has its label reversed with the same fixed probability. Kearns's *statistical query* method [12, 14] gives a general approach for the construction of *provably*

* Partially supported by OTKA grant T-14228 and Phare TDQM grant 9305-02/1022 (ILP2/HUN). Email: `thorvath@sol.cc.u-szeged.hu`.

** Partially supported by NSF grant CCR-9314258. Email: `sloan@eecs.uic.edu`.

*** Partially supported by NSF grant CCR-9208170, OTKA grant T-14228 and Phare TDQM grant 9305-02/1022 (ILP2/HUN). Email: `U11557@uicvm.uic.edu`.

noise-tolerant learning algorithms in the random classification model. The statistical query method is based on estimating probabilities and thus it may be viewed as a formalization of the heuristics referred to above.

Most Boolean concept classes that are known to be *PAC*-learnable can be shown to be *PAC*-learnable with random classification noise using the statistical query method. Also, several predicate-logic learning problems have been shown to be *PAC*-learnable in the noise-free model by using a reduction to a learning problem in propositional logic. Results of these two types can be combined to prove the *PAC*-learnability of predicate-logic learning problems in the presence of random classification noise. (Most positive results about the learnability of logic programs apply to nonrecursive Horn clauses with ground background knowledge. The usual approaches to these problems, such as least general generalizations, assume that the data are noiseless and thus cannot be used in the present context.)

Gennaro [8] obtained the first theoretical results on learnability in predicate logic in the presence of noise, using the basic argument outlined above. His results apply to nonrecursive Horn clauses of constant locality. Džeroski [6], besides proving related results, gave positive results in the framework of *nonmonotonic ILP*. Both discussed the learnability of restricted *multiple clause* nonrecursive logic programs with noise [6, 8], although mostly not in the *PAC* setting studied in this paper, where we assume that the hypothesis class and the concept class are the same, and permit arbitrary probability distributions on the examples.

In this paper we provide further positive results about the learnability of restricted classes of logic programs in the random classification noise model. We consider single *unrestricted* nonrecursive clauses with restricted background knowledge, and restricted recursive clauses.

In Sects. 3 and 4 we present the statistical query model and the standard algorithm for learning Boolean conjunctions with statistical queries. Section 5 discusses nonrecursive determinate constant-depth Horn clauses (defined by Muggleton and Feng [19]). This class was shown to be *PAC*-learnable by Džeroski, Muggleton, and Russell [7]. We review the algorithm to learn this class with random classification noise [6, 8], using the framework and the description of the noise-free learning algorithm given by Cohen [3]. The noisy learning algorithm proceeds by building a large set of literals that contains all possible literals that might occur in the body of the target clause, and then applying the noisy conjunction learning algorithm to find the body of the hypothesis clause.

In Sect. 6 we consider *nonrecursive Horn clauses with forest background knowledge*. This is a class of unrestricted clauses with a single binary background predicate that was proven to be *PAC*-learnable by Horváth and Turán [11]. Although these clauses are not necessarily determinate, one can still view the body of such a clause as a conjunction of literals (corresponding to *new* predicates) depending only on the variables occurring in the head. An example of such a predicate is “the height of element x in the tree is at least 3”. When this predicate is translated back to the original background predicate, new *nondeterminate* literals will occur that cannot be evaluated directly for a given example.

Thus the probability estimates in the learning algorithm will refer to the new predicates.

In several logic learning problems, the number of literals is quite large in terms of the learning parameters. Thus, for the sake of efficiency, it is desirable to find learning algorithms that use a smaller sample size than the one obtained by a direct reduction to learning Boolean conjunctions. The case of forest background knowledge shows that in some cases this is indeed possible. We make use of the fact that there are *dependencies* between the different literals. For example, "x has height at least 3" implies "x has height at least 2", and "x and y have distance 2" excludes "x and y have distance 3". These dependencies can be used to show that the probabilities needed may be estimated with a *larger tolerance* than in the general case. As the sample size depends *quadratically* on the inverse of the smallest tolerance, this results in a significant decrease in the sample size.

In Sect. 7 we consider the learnability of *recursive* logic programs in the presence of random classification noise. Cohen's results [3, 4] establish a sharp boundary for those classes of recursive logic programs that are *PAC*-learnable. In particular, he shows that a restricted class of logic programs consisting of a determinate recursive clause, a nonrecursive clause, and ground background knowledge is *PAC*-learnable. The details of these restrictions are given in Sects. 2.2 and 7.

We show that this class also remains *PAC*-learnable with random classification noise. We note that although the learnability of recursive logic programs in the presence of noise is a topic of interest in itself, from the practical point of view the nonrecursive case appears to be more important. The reason for this is that target concepts for application domains with noisy data appear to be nonrecursive (see e.g., Cohen [3]). The learning algorithm presented in Sect. 7 selects the literals to be included in the body of the recursive target clause using statistical queries, similarly to the nonrecursive case. In this case, however, the queries become considerably more complicated as they have to take into account the behavior of the literal on the recursive calls, similarly to Cohen's original algorithm.

Section 8 contains a review of additional noise models from computational learning theory, and some possible directions for further research. In Appendix A, we give some mathematical details for the statistical query method.

2 Preliminaries

Let P be the *target* predicate of arity m, and $R_1, \ldots, R_r$ be the *background* predicates of arities $m_1, \ldots, m_r$. The *constants* are denoted by $a_1, \ldots, a_n$.

An *atomic formula* or *atom* is of the form $P(t_1, \ldots, t_m)$ or $R_i(t_1, \ldots, t_{m_i})$, where the t's are *terms* (i.e., variables or constants). An atom is called a *P-atom* (or an *R_i-atom*) if its predicate symbol is P (resp. R_i). An atom is *ground* if it contains no variables. Ground atoms are also referred to as *facts*.

The *background knowledge* $\mathcal{B}$ used in the learning process is assumed to consist of ground atoms of the background predicates.

In this setting, a *concept* will be a set of ground atoms of the target predicate. A *concept class* is a set of concepts. *Positive* and *negative examples* are ground atoms of the target predicate P, labeled according to their classification by a $+$ or $-$.

2.1 Nonrecursive Logic Programs and Determinateness

A *definite, nonrecursive Horn clause* is of the form

$$A \longleftarrow B_1, \ldots, B_l \ ,$$

where A is a P-atom and each B_i is a background predicate atom.

Given a background knowledge $\mathcal{B}$, a definite, nonrecursive Horn clause C represents the concept $C_{\mathcal{B}}$ consisting of all the ground atoms of the target predicate that are implied by the logic program $(C, \mathcal{B})$.

Now we turn to determinate clauses. In this case the atoms in the body are assumed to be *ordered*. Consider a clause $A \longleftarrow B_1, \ldots, B_l$. The *input variables* of an atom B_i are those variables which occur in at least one of the atoms preceding B_i. Its other variables are called *output variables*.

An atom B_i in the body is *determinate* if the following holds. Let σ be a substitution that maps A to some P-fact and maps the atoms $B_1, \ldots, B_{i-1}$ to facts in $\mathcal{B}$. Then there is *at most one* binding of the output variables in B_i that maps B_i to some fact in $\mathcal{B}$. A clause is determinate (with respect to the background knowledge $\mathcal{B}$) if all atoms in its body are determinate.

Determinate clauses are often assumed to have constant depth. The *depth* of a variable is defined as follows. Variables in the head have depth 0. Assume that we determined the depth of all variables in $B_1, \ldots, B_{i-1}$, and let x be an output variable of B_i. Thus, B_i is the first atom where x occurs. Let d be the maximum depth of the input variables of B_i. Then the depth of x is $d + 1$.

It is a hard problem to decide if a clause is determinate with respect to a background knowledge. The problem in general can be shown to be co-NP-complete. Cohen introduced natural additional conditions that lead to a nice syntactic class of determinate clauses [3].

The *mode* of an atom is a partition of its variables into *input* and *output variables*. For example, if $\text{Father}(x, y)$ is a binary predicate corresponding to "the father of x is y", then $[\text{Father}, +, -]$ is a mode. A *mode constraint* M is a set of modes. Each predicate must have a unique mode. A clause *satisfies* a mode constraint, if for each of its atoms, the division of its variables into input and output variables is consistent with a mode in the mode constraint. We assume that the predicate Equal with mode $[\text{Equal}, +, +]$ is part of $\mathcal{B}$ and M. Our results remain valid *without* the assumptions about the unique mode and the Equal predicate [3].

Now, a mode for a given predicate R is *determinate with respect to a background knowledge* $\mathcal{B}$ if the R-facts in $\mathcal{B}$ form a (partial) function mapping the input variables to the output variables. For example, the mode $[\text{Father}, +, -]$ is determinate with respect to databases representing family relationships. A mode constraint is determinate with respect to $\mathcal{B}$ if all its modes are determinate with respect to $\mathcal{B}$. We note that for every fixed set of background predicates it can be checked efficiently whether a given mode constraint is determinate with respect to a given background knowledge.

Then, it is clear from the definitions that any clause satisfying a determinate mode constraint with respect to $\mathcal{B}$ is determinate with respect to $\mathcal{B}$.

Let M be a determinate mode constraint with respect to the background knowledge $\mathcal{B}$. We denote by $\mathcal{C}_{\mathcal{B},d,M}$ the class of concepts with background knowledge $\mathcal{B}$ that are represented by Horn clauses of depth at most d satisfying M.

The parameters measuring the size of a learning problem are m, the *arity of the target predicate*, and n, the *number of constants* in $\mathcal{B}$. We assume throughout that the arity of every background predicate is bounded by a constant.

2.2 Recursive Logic Programs

Now we describe a modification of the framework above that is suitable for determinate *recursive* clauses, due to Cohen [3].

The basic objects are assumed to come from an infinite set, such as $\{0,1\}^*$ or $\mathbb{N}^*$, replacing the constants $a_1, \ldots, a_n$. The *size* of a string or a sequence is its length and the size of a ground atom is the sum of the sizes of its arguments.

A *recursive* clause differs from a nonrecursive clause in that its body may contain P-atoms. These are called the *recursive literals* of the body. We will restrict ourselves to closed linear recursive clauses. A recursive clause is *linear* if its body contains exactly one recursive literal, and it is *closed* if its recursive literals have no output variables.

Examples are now labeled *extended instances* consisting of a labeled ground P-atom and a set of ground atoms of the background predicates, forming a *description* of the ground P-atom. For instance, a positive example of the target concept *append* is

Labeled fact: (append($l12, l3, l123$), $+$)
Description: {components($l12, 1, l2$), components($l2, 2$, nil),
components($l123, 1, l23$), components($l23, 2, l3$), components($l3, 3$, nil)}

The size of an extended instance is the sum of the sizes of its ground atoms.

One reason for this formalism is to allow us to discuss recursive structures, such as lists, without using function symbols. Also, this modification provides a framework for learning predicates of *constant arity* from an *exponential-size concept class* over an exponential-size or infinite domain. The arity of the target predicate and the arities of the background predicates are assumed to be fixed. (We know how to *PAC* learn recursive clauses only when the arity of the target predicate is constant.) The relevant parameters for the learning problem are the *bound on the size of the examples* and the *sizes of the background knowledge* and the *mode declaration*. Note that only part of the background knowledge is given explicitly in $\mathcal{B}$. Further facts from the background knowledge are given in the descriptions of the extended examples. Determinateness is required for each example with respect to $\mathcal{B}$ and its description.

2.3 *PAC* Learning and Random Classification Noise

Now we give a brief definition of *PAC*-learnability. Our description is informal; more details are given, for example, in Kearns and Vazirani [14].

It is assumed that there is a probability distribution on the set of all ground P-atoms. A learning algorithm first draws a sequence of random examples and then it outputs a *hypothesis* from the target class. The *error* of a hypothesis is the probability of the set of misclassified instances.

A family of learning problems is *polynomially PAC-learnable* if there is a learning algorithm with inputs ϵ, δ, and the background knowledge, and access to examples, such that the following hold. First, there is a polynomial in $1/\epsilon$, $1/\delta$, and the other relevant learning parameters that bounds the time of the learning algorithm. Second, for every learning problem that belongs to the family, for every $\epsilon, \delta > 0$, for every target concept and for every probability distribution on the domain, Pr(error of the hypothesis output by the algorithm $> \epsilon) \leq \delta$. (In the introduction, the term "polynomially," which refers to the running time, was omitted for simplicity.)

Formally, one speaks of the training data coming from an oracle EXAMPLES, which draws ground P-atoms from a fixed but unknown probability distribution, and returns the atoms together with the proper label. To study *random classification noise with rate* $\eta < 1/2$, we define a noisy oracle, EXAMPLES$^\eta$, which gets labeled examples from EXAMPLES, and for each example reverses the label independently with probability η. We say that a family of learning problems is *polynomially PAC-learnable with random classification noise* if the definition of polynomial *PAC*-learnability is met with the EXAMPLES oracle replaced by the noisy oracle EXAMPLES$^\eta$. Notice that formally the performance of the learning algorithm is measured against the correct labels; the noise occurs only in the training set. In fact, a hypothesis that more closely matches the true labels will also more closely match the labels given by the noisy oracle.

3 *PAC* Learning with Statistical Queries

In this section, we present a method for *PAC* learning a concept class from data with random classification noise. Kearns's *statistical query method* [12, 14] allows one to learn most, although not all, *PAC*-learnable concept classes with noisy data of this type.

Here we will present an overview of the method. In Appendix A we will prove some of the specific bounds on sample size. Our presentation generally follows Kearns and Vazirani's textbook [14], although we provide explicit, and slightly different, bounds on sample size instead of big-O bounds. Another proof of Theorem 1, below, is given by Aslam and Decatur [2].

Let us assume that we are trying to learn target concept C on instance space X, with distribution D. A *statistical query learning algorithm* has access to the STAT oracle instead of the usual EXAMPLES oracle. The STAT oracle will tell us statistical properties of labeled examples, instead of giving us a sample of labeled examples.

More precisely, a query (χ, τ) to the STAT oracle asks for an approximation to $\Pr_D(\chi(x, C(x)) = 1)$ with additive *tolerance* τ. Here $\chi : X \times \{0,1\} \to \{0,1\}$ is a $\{0,1\}$-function of labeled examples that can be computed in polynomial time. Thus, if $X = \{0,1\}^n$ we can ask the oracle questions such as, "What fraction of examples have the second bit off and a label +?" We get back an answer that is guaranteed to be accurate to within τ.

A family of learning problems is *polynomially learnable from statistical queries* if there is a learning algorithm L running in time polynomial in $1/\epsilon$ and the learning parameters, such that all of L's queries have tolerance bounded by the inverse of a polynomial of $1/\epsilon$ and the learning parameters, and L outputs a hypothesis with error at most ϵ. The main result is the following.

Theorem 1 (Kearns). *Any concept class that is polynomially learnable from statistical queries can be polynomially* PAC *learned with random classification noise.*

We give an outline of the proof. We show how to simulate the answer to a statistical query (χ, τ) with noisy examples. The key idea is to partition the instance space X into X_1 and X_2, where X_1 is the subset of X for which $\chi(x, 0) \neq \chi(x, 1)$. That is, X_1 is the part of X where the answers to query χ depend on the label of the example, and X_2 is the part where the answers do not depend on the label. Notice that for any example we can determine whether it is in X_1 or X_2, because we can evaluate χ.

Let p_1 and p_2 be the total probability weight that D assigns to X_1 and X_2, respectively, and let D_1 and D_2 be the probability measures induced on X_1 and X_2, respectively, by D. (Thus $D_1 = D/p_1$ and $D_2 = D/p_2$. We will see in (1) and (2) that we never use D_2 directly, and we do not use D_1 when p_1 is 0.)

We wish to estimate $\Pr_D(\chi(x, C(x)) = 1)$; we will abbreviate this quantity by $\Pr_D(\chi = 1)$. Now

$$\Pr_D(\chi = 1) = p_1 \Pr_{D_1}(\chi = 1) + \Pr_D(\chi = 1 \wedge x \in X_2) \ . \tag{1}$$

One can estimate p_1 directly from the noisy oracle. The same holds for $\Pr_D(\chi = 1 \wedge x \in X_2)$, since we can determine whether an instance is in X_2, and if it is, then the label of the example is irrelevant for computing χ.

We also need to show how to estimate $\Pr_{D_1}(\chi = 1)$. Let us use $\Pr_{D,\eta}(E)$ to denote the probability of event E with respect to examples drawn from the random classification noise oracle with noise rate η and distribution D on the instances. We show how to calculate $\Pr_{D_1}(\chi = 1)$ given η and $\Pr_{D_1,\eta}(\chi = 1)$.

Now $\Pr_{D_1,\eta}(\chi = 1) = \eta + (1 - 2\eta) \Pr_{D_1}(\chi = 1)$. Thus we can rewrite (1) as

$$\Pr_D(\chi = 1) = p_1 \frac{\Pr_{D_1,\eta}(\chi = 1) - \eta}{1-2\eta} + \Pr_{D,\eta}(\chi = 1 \wedge x \in X_2) \ . \tag{2}$$

Notice that all the quantities except η on the right-hand side of (2) can be estimated directly from the noisy oracle.

Consider a fixed algorithm SQA that learns from statistical queries. Given the noise rate η, we could simulate Algorithm SQA, by drawing a sufficiently large sample of say, m_1 examples, and for each query asked by Algorithm SQA, using that sample to estimate the quantities on the right-hand side of Equation 2. This algorithm, which we will call Algorithm GENERATEHYPOTHESIS(η), would give the correct answer with high probability. There is some chance for error because it might be that one of the quantities that we estimated empirically was badly estimated. We note that this is the point where the confidence parameter δ enters the estimates, as δ is not a parameter of a statistical query algorithm.

Now, given an upper bound η_b on the noise rate η, we could simply run Algorithm GENERATEHYPOTHESIS on inputs $0, \Delta, 2\Delta, 3\Delta, \ldots (\eta_b/\Delta)\Delta$, for some appropriate small value of Δ. Each run would generate a hypothesis. We then draw another sample of m_2 examples, and measure the empirical error rate of each hypothesis on this sample. One can show that even with examples from the noisy oracle, the lower the actual error rate of a hypothesis, the lower the expected observed error rate. (Of course, with examples from the noisy oracle, the minimum error rate will be η, not 0.) Thus, we can use the hypothesis with the lowest error rate as our final output.

To make our algorithm complete, we need to specify the values of m_1, m_2, and Δ. We assume that we have a statistical query algorithm that draws all its queries from a finite set $\mathcal{Q}$, and that the smallest tolerance that it ever uses in any query is α. We also assume we are given an upper bound η_b on the actual noise rate. (Kearns [12] shows how to remove this assumption.) Moreover, our final goal is to *PAC* learn with the usual parameters ϵ and δ.

We use $\Delta \leq \alpha(1-2\eta_b)^2/16$ [14].[4] The other two values are

$$m_1 = \frac{32}{\alpha^2(1-2\eta_b)^2} \ln\left(\frac{12|\mathcal{Q}|}{\delta}\right) \tag{3}$$

$$m_2 = \frac{2}{\epsilon^2(1-2\eta_b)^2} \ln\left(\frac{16}{(1-2\eta_b)^2\alpha\delta}\right) \ . \tag{4}$$

The total number of samples required is $m_1 + m_2$. Justification for these values is given in Appendix A.

Note: The value α should be a lower bound for the smallest tolerance used by a statistical query algorithm that learns with an accuracy of $\epsilon/2$ (not ϵ), for reasons we sketch in Appendix A.

4 Learning Conjunctions from Noisy Data

Angluin and Laird [1] gave the first algorithm for *PAC* learning conjunctions from data with random classification noise. Their algorithm can be modified using the statistical query method to get an improved bound on the sample size.

Let us assume that our target concept C is a conjunction of some subset of a set of n given variables. (That is, negation is not allowed. It is easy to generalize

[4] Kearns and Vazirani [14] show that $\Delta = c_0\alpha'(1-2\eta)^2/2$, where α' is a parameter smaller than α which we show in Appendix A can be safely set to $\alpha/8$. It is easy to fill in the missing details and see that setting the leading constant $c_0 = 1/2$ works.

to allow negation.) For a fixed target concept, a distribution D on the instances, and a variable v, let us define $p_{01}(v)$ to be $\Pr_D$(positive instances with v false). We call variable v *harmful* if $p_{01}(v) > K$, where for now we set $K = \epsilon/n$.

Notice that if v is actually in the target concept, then $p_{01}(v) = 0$. Any conjunction that includes all variables with $p_{01} = 0$ and no harmful variables will differ from the target conjunction by at most ϵ [1]. Thus a statistical query learning algorithm for conjunctions is to ask for $p_{01}(v)$ for every variable v with tolerance $K/2$, and include all those variables where the query is answered with a value of $K/2$ or less.

This gives a sample complexity of

$$\frac{1}{\epsilon^2(1-2\eta_b)^2}\left(128n^2\ln\left(\frac{12n}{\delta}\right)+\ln\left(\frac{2n}{(1-2\eta_b)^2\epsilon\delta}\right)\right) \quad .$$

Typically, when the conjunctions we are learning are the bodies of Horn clauses, we would prefer a smaller conjunction to a larger conjunction as the output of the learning algorithm. We can sometimes get such a reduction in the size of the learned conjunction by omitting those variables that are almost always 1.

Let us call a variable *insignificant* (or *irrelevant* in the terminology of Lavrač and Džeroski [16]) if the probability that it is false in any example is less than $\epsilon/2n$. If our learning algorithm omits all insignificant variables from its output, it increases its error rate by at most $\epsilon/2$. We can discard some insignificant variables by asking for the probability that each variable is off with tolerance $\epsilon/4n$, and discarding those where the answer is at most $\epsilon/4n$. We also need to adjust the value of K to $\epsilon/2n$, so the error in our learner's hypothesis from including (nonharmful) variables not in the target concept is at most $\epsilon/2$.

This modification does not change the asymptotic sample complexity of the algorithm, but the constants do increase so the sample complexity becomes

$$\frac{1}{\epsilon^2(1-2\eta_b)^2}\left(2048n^2\ln\left(\frac{24n}{\delta}\right)+\ln\left(\frac{8n}{(1-2\eta_b)^2\epsilon\delta}\right)\right) \quad .$$

5 Nonrecursive Determinate Constant-Depth Clauses

We present a version of the algorithm to learn nonrecursive, determinate constant-depth Horn clauses with random classification noise [6, 8]. We use Cohen's approach [3] to the noise-free case as a starting point.

We assume that we are given a background knowledge $\mathcal{B}$, a depth bound d, and a mode constraint M that is determinate with respect to $\mathcal{B}$. We are trying to learn a target concept from $\mathcal{C}_{\mathcal{B},d,M}$.

One can construct a clause $\mathrm{BOTTOM}^*_d(P, m, M)$ in polynomial time such that the target clause must be semantically equivalent to one of its subclauses [3].

We could almost view the body of this clause as a large conjunction, and apply the algorithm of Sect. 4, with the atoms in the clause's body playing the role of the variables. The reason we write "almost" is because any given atom in the body might be true, false, or undefined for a particular binding of the variables in the head. Thus the notions of *harmful* and *insignificant* of Sect. 4 must be modified by replacing "false" by "false or undefined".

For an ordered clause $A \longleftarrow B_1, \ldots, B_l$, we say an atom B_i (or A) *directly supports* atom B_j, if some output variable of B_i is an input variable of B_j. Atom B_i *supports* atom B_j if there is a (possibly empty) sequence of atoms $C_1, \ldots, C_s$ such that B_i directly supports C_1, each C_k directly supports C_{k+1}, for $1 \leq k \leq s-1$, and C_s directly supports B_j.

Now we could modify our statistical query conjunction learning algorithm from Sect. 4 to inquire about the atoms in order. Whenever an atom is to be deleted, also delete all atoms it supports. This will in fact work for the first algorithm we gave (the one that considers only harmfulness and not significance). We get an algorithm that *PAC* learns nonrecursive determinate constant-depth Horn clauses, with the drawback that it will always learn as large a body as possible.

However, we cannot simply apply the significance modification of the conjunction algorithm, and discard all insignificant atoms and all atoms supported by them. The difficulty is that an insignificant atom might support a significant atom.

Thus we are led to the following statistical query algorithm. Let $\mathrm{BOTTOM}_d^*(P, m, M)$ be $P(\mathbf{x}) \leftarrow L_1, \ldots, L_k$. Define the probabilities

$$
\begin{aligned}
p_i &= \Pr(P(\mathbf{x}) \text{ is positive and } L_i \text{ is false or undefined with binding} \\
&\qquad \text{induced by } \mathbf{x}) \\
q_i &= \Pr(L_i \text{ is false or undefined}) \ .
\end{aligned}
$$

Algorithm DET.

1. Compute $S := \mathrm{BOTTOM}_d^*(P, m, M)$. Put $T = \epsilon/4k$.
2. For each literal L_i in order, obtain an estimate $\hat{p}_i$ of p_i with tolerance T. If $\hat{p}_i \leq T$ then retain L_i in S. Otherwise, delete L_i and all atoms supported by L_i from S.[5]
3. For each remaining literal L_j in S, obtain an estimate $\hat{q}_j$ of q_j with tolerance T.
   If $q_j \leq T$, then label L_j insignificant.
4. Delete from S all insignificant literals that do not support any significant literals.

Theorem 2. *Algorithm* DET *is a polynomial statistical query algorithm for the concept class $\mathcal{C}_{\mathcal{B},d,M}$. Thus, the concept class $\mathcal{C}_{\mathcal{B},d,M}$ is polynomially PAC learnable with random classification noise with sample size*

$$
O\left(\frac{1}{\epsilon^2(1-2\eta_b)^2}\left((2m|M|)^{2a^{d+1}} \ln\left(\frac{(2m|M|)^{a^{d+1}}}{\delta}\right) + \ln\left(\frac{(2m|M|)^{a^{d+1}}}{(1-2\eta_b)^2\epsilon\delta}\right)\right)\right) ,
$$

[5] Notice that some literals may be deleted before we can calculate $\hat{p}_i$ for them.

where $|M|$ is the number of modes in M, and constant a is the maximum arity of any predicate in $\mathcal{B}$.

Proof. The theorem follows from the previous discussion and the following lemma. Let the target concept be C and let H be the concept represented by the output of Algorithm DET.

Lemma 3. $\Pr(C \triangle H) \leq \epsilon$.

Proof of lemma. Suppose $C(\mathbf{a})$ is 0 but $H(\mathbf{a})$ is 1. Then there is a literal L_j in the body of the target clause that does not belong to the body of the hypothesis clause. It must be that $p_j = 0$, so $\hat{p}_j \leq T$. Similarly, $\hat{p}_\ell \leq T$ for all literals L_ℓ supporting L_j. Thus we could not have removed L_j in Step 2. We must have removed L_j because it was an insignificant literal that supports only insignificant literals. Thus $\hat{q}_j \leq \epsilon/4k$, and $q_j \leq \epsilon/2k$, so the error contributed by L_j is at most $\epsilon/2k$. Hence the total error in H from all such missing literals is at most $\epsilon/2$.

Now assume that $C(\mathbf{a})$ is 1 but $H(\mathbf{a})$ is 0. Clearly in this case there is a literal L_h in H that does not occur in C. We know that $\hat{p}_h \leq T$ and therefore $p_h \leq 2T = \epsilon/2k$. The total error contributed by L_h is at most $\epsilon/2k$, and the total error contributed by all such literals is at most $\epsilon/2$.

Thus the total error of H is at most ϵ, as desired. □

We can calculate that $\mathrm{BOTTOM}^*_d(P, m, M)$ has at most $(2m|M|)^{a^{d+1}}$ literals in its body, and this gives us the sample complexity. □

6 Forest Background Knowledge

In this section we assume that there is a single binary background predicate R. The facts $R(a_i, a_j)$ belonging to the background knowledge $\mathcal{B}$ may be viewed as a directed graph with edges (a_i, a_j). Horváth and Turán [11] showed that if this graph is a forest with edges directed towards the roots, then a single non-recursive Horn clause is polynomially PAC-learnable. Their algorithm is based on computing least general generalizations, and thus it is not noise tolerant. In this section, we give a noise-tolerant learning algorithm for this problem.

First let us introduce some notations. If a is a node then $f(a)$ denotes its parent and $f^{(i)}(a) = f(f^{(i-1)}(a))$ denotes its ith parent ($f^{(i)}(a)$ may be undefined). The *height* of a, denoted by $h(a)$, is the number of edges on the longest directed path ending at a. If $f^{(i)}(a)$ is undefined then we define $h(f^{(i)}(a))$ to be 0. A path is a (d_1, d_2)-*path* if it consists of d_1 forward edges followed by d_2 backward edges.

Let $\mathbf{x} = (x_1, \ldots, x_m)$ be an m-tuple of variables. We define the following predicates:

$$E_{u,v,d_1,d_2}(\mathbf{x}) \Leftrightarrow \text{ there is a } (d_1, d_2)\text{-path from } x_u \text{ to } x_v$$

for $1 \leq u < v \leq m$ and $0 \leq d_1, d_2 \leq n$,

$$F_{u,l,k}(\mathbf{x}) \Leftrightarrow h(f^{(l)}(x_u)) \geq k$$

for $1 \le u \le m$, $0 \le l < n$, and $1 \le k < n$, and

$$I_{u,j}(\mathbf{x}) \Leftrightarrow x_u = a_j$$

for $1 \le u \le m$ and $1 \le j \le n$.

We note that the predicates $E_{u,v,d_1,d_2}(\mathbf{x})$ and $F_{u,l,k}(\mathbf{x})$ introduce new, usually nondeterminate, variables when written out in terms of the background predicate.

Every concept defined by a nonrecursive Horn clause has a definition of the form $P(\mathbf{x}) \longleftarrow L_1(\mathbf{x}), \ldots, L_s(\mathbf{x})$, where each literal in the body is of the form $E_{u,v,d_1,d_2}(\mathbf{x})$, $F_{u,l,k}(\mathbf{x})$, or $I_{u,j}(\mathbf{x})$ [11]. Furthermore, for every u and v there is at most one literal of the form $E_{u,v,d_1,d_2}(\mathbf{x})$, for every u and l there is at most one literal of the form $F_{u,l,k}(\mathbf{x})$, and for every u there is at most one literal of the form $I_{u,j}(\mathbf{x})$ in the body.

Thus learning a nonrecursive Horn clause can be viewed as learning a conjunction of the literals $E_{u,v,d_1,d_2}(\mathbf{x})$, $F_{u,l,k}(\mathbf{x})$ and $I_{u,j}(\mathbf{x})$. The number of all these literals together is of the order m^2n^2. The statistical query learning algorithm described in Sect. 4 for learning a conjunction can be adapted to this case as well, similarly to the case of determinate clauses discussed in the previous section. The dependence of the sample size of this algorithm on m and n is of the form m^4n^4. (In this paragraph, we ignore log factors.) In order to improve this rather large sample size, one may observe that the target conjunction contains at most m^2+mn of the m^2n^2 literals. Kearns [12] showed that the sample size can be improved if the conjunctions are known to contain few literals. The bound of [12] applied to our case is of the order $(m^2+mn)^3$. We now give another statistical query learning algorithm for this particular case. The sample size provided by this algorithm is of the order m^4n^2. This is never worse than the bound for Kearns's algorithm referred to above and it is better, for example, when n (the number of constants in the database) is much larger than m (the arity of the target predicate), which is usually expected to be the case.

Theorem 4. *Assume that there is a single binary background predicate and the ground atoms in the background knowledge form a forest. Then a nonrecursive Horn clause is polynomially PAC learnable with random classification noise with sample size*

$$O\left(\frac{m^4n^2}{\epsilon^2(1-2\eta_b)^2}\ln\left(\frac{m^2n^2}{\delta}\right)+\frac{1}{\epsilon^2(1-2\eta_b)^2}\ln\left(\frac{m^2n}{\delta\epsilon(1-2\eta_b)^2}\right)\right) \ .$$

Proof. Let us define the following probabilities:

$$\begin{aligned}
p &= \Pr(\mathbf{x} \text{ is a positive example}) \ , \\
p_{u,v,d_1,d_2} &= \Pr(\mathbf{x} \text{ is a positive example and } \overline{E_{u,v,d_1,d_2}(\mathbf{x})}) \ , \\
p_{u,l,k} &= \Pr(\mathbf{x} \text{ is a positive example and } \overline{F_{u,l,k}(\mathbf{x})}) \ , \text{ and} \\
p_{u,j} &= \Pr(\mathbf{x} \text{ is a positive example and } \overline{I_{u,j}(\mathbf{x})}) \ .
\end{aligned}$$

In each case the probability is with respect to $\mathbf{x}$. The ranges for the subscripts are the same as those for the corresponding predicates. Let $M = \binom{m}{2}n$ if $m > 1$ and $M = 1$ if $m = 1$. Then the algorithm works as follows.

Algorithm FOREST.

1. Obtain an estimate $\hat{p}$ of p with tolerance $\epsilon/12M$. If $\hat{p} \le \frac{5}{6}\frac{\epsilon}{M}$, stop and output the hypothesis that P is always false.
Otherwise, initialize the hypothesis clause $S := P(\mathbf{x}) \leftarrow$.
2. For $1 \le u \le m$, $0 \le l < n$ and $1 \le k < n$, obtain an estimate $\hat{p}_{u,l,k}$ of $p_{u,l,k}$ with tolerance $\epsilon/6mn$.
Include $F_{u,l,k}(\mathbf{x})$ in the body of S for the largest k (if any) such that $\hat{p}_{u,l,k} \le \epsilon/6mn$.
3. If $m > 1$, for $1 \le u < v \le m$, $0 \le d_1, d_2 < n$ obtain an estimate $\hat{p}_{u,v,d_1,d_2}$ of p_{u,v,d_1,d_2} with tolerance $\epsilon/12M$.
Include in the body of S all $E_{u,v,d_1,d_2}(\mathbf{x})$ such that $|\hat{p}_{u,v,d_1,d_2} - \hat{p}| \le \epsilon/6M$ (if any).
4. For $1 \le u \le m$, $1 \le j \le n$, obtain an estimate $\hat{p}_{u,j}$ of $p_{u,j}$ with tolerance $\epsilon/12M$.
If there is a j such that $|\hat{p}_{u,j} - \hat{p}| \le \epsilon/6M$, then include in the body of S the $I_{u,j}(\mathbf{x})$ such that $|\hat{p}_{u,j} - \hat{p}|$ is as small as possible.

We note that the literals included in S have to be expanded into literals of the background predicate in order to get a clause of the required form. For example, $F_{u,1,2}(\mathbf{x})$ is transformed into $R(x_u, y_1), R(y_2, y_1), R(y_3, y_2)$, corresponding to $h(f^{(1)}(x_u)) \ge 2$. Similarly, $E_{u,v,2,1}(\mathbf{x})$ is transformed into $R(x_u, z_1), R(z_1, z_2)$, $R(x_v, z_2)$, describing a $(2,1)$-path form x_u to x_v. All the new variables introduced should be different.

Let C be the target concept, and let H be the concept represented by the output of Algorithm FOREST. The correctness of the algorithm follows from the following lemma.

Lemma 5. *$H \subseteq C$ and* $\Pr(C \setminus H) \le \epsilon$.

Proof of lemma. First, if $\hat{p} \le \frac{5}{6}\frac{\epsilon}{M}$ in Step 1 of the algorithm, then $p \le \left(\frac{5}{6} + \frac{1}{12}\right)\frac{\epsilon}{M} \le \epsilon$. Thus the statement of the lemma holds in this case. In the remainder of the argument, we assume $\hat{p} > \frac{5}{6}\frac{\epsilon}{M}$.

Now we show that $\Pr(C \setminus H) \le \epsilon$.

If a literal of the form $F_{u,l,k}(\mathbf{x})$ is included in S, then the conditions of Step 2 imply that the probability weight of positive examples not satisfying $F_{u,l,k}(\mathbf{x})$ is $p_{u,l,k} \le \hat{p}_{u,l,k} + \frac{\epsilon}{6mn} \le \frac{\epsilon}{3mn}$. Summing up for all possible values of u and l, it follows that the probability weight of the positive examples lost by the inclusion of these literals is at most $\epsilon/3$.

Now let us assume that a literal of the for $E_{u,v,d_1,d_2}(\mathbf{x})$ is included in S. Then by definition, $|\hat{p}_{u,v,d_1,d_2} - \hat{p}| \le \epsilon/6M$. As the tolerance is $\epsilon/12M$, this implies that $|p_{u,v,d_1,d_2} - p| \le \epsilon/3M$. As we assume that $\hat{p} \ge \frac{5}{6}\frac{\epsilon}{M}$, it follows that $p \ge \frac{3}{4}\frac{\epsilon}{M}$, and so $\epsilon/3M < p/2$. Thus it holds that $p_{u,v,d_1,d_2} > p/2$. We now argue that we cannot include both $E_{u,v,d_1,d_2}(\mathbf{x})$ and $E_{u,v,d_3,d_4}(\mathbf{x})$ for $d_3 - d_4 \ne d_1 - d_2$. If both those literals were included, then $p_{u,v,d_1,d_2} > 1/2$ and $p_{u,v,d_3,d_4} > 1/2$. This would imply that there is a positive example $\mathbf{x}$ with both $E_{u,v,d_1,d_2}(\mathbf{x})$ and $E_{u,v,d_3,d_4}(\mathbf{x})$ true, but from the properties of trees, that is impossible.

Thus only literals $E_{u,v,d_5,d_6}(\mathbf{x})$ with $d_5 - d_6 = d_1 - d_2$ can be included in the hypothesis. There are at most n such literals. For these literals it holds that $|\hat{p}_{u,v,d_5,d_6} - \hat{p}| \leq \epsilon/6M$, and so $|p_{u,v,d_5,d_6} - p| \leq \epsilon/3M$. Hence, summing over all possible values of u and v, we get that the probability weight of positive examples lost by the inclusion of literals of the form $E_{u,v,d_1,d_2}(\mathbf{x})$ is at most $\epsilon/3$.

Similarly to the first case, if a literal of the form $I_{u,j}(\mathbf{x})$ is included in S, then $|\hat{p}_{u,j} - \hat{p}| \leq \epsilon/6M$, and so $|p_{u,j} - p| \leq \epsilon/3M$. Thus again, the probability weight of positive examples lost by the inclusion of literals of the form $I_{u,j}(\mathbf{x})$ is at most $\epsilon/3$. Considering the three cases together, it follows that $\Pr(C \setminus H) \leq \epsilon$.

It remains to be shown that $H \subseteq C$.

If there is a literal of the form $F_{u,l,k'}(\mathbf{x})$ in the body of the target clause, then $p_{u,l,k'} = 0$. Hence $\hat{p}_{u,l,k'} \leq \epsilon/6mn$. Therefore k' was a candidate in Step 2 and so there must be a literal $F_{u,l,k}(\mathbf{x})$ in S such that $k' \leq k$. But $F_{u,l,k}(\mathbf{x})$ implies $F_{u,l,k'}(\mathbf{x})$, so no new positive examples are included in H by replacing $F_{u,l,k'}(\mathbf{x})$ by $F_{u,l,k}(\mathbf{x})$.

In the other two cases, every literal that occurs in the body of the target clause will also occur in S. If a literal of the form $E_{u,v,d_1,d_2}(\mathbf{x})$ is contained in the body of the target clause, then $p_{u,v,d_1,d_2} = p$ and so $|\hat{p}_{u,v,d_1,d_2} - \hat{p}| \leq \epsilon/6M$. Thus $E_{u,v,d_1,d_2}(\mathbf{x})$ is included in S.

If a literal of the form $I_{u,j}(\mathbf{x})$ is contained in the body of the target clause, then $p_{u,j} = p$, and so $|\hat{p}_{u,j} - \hat{p}| \leq \epsilon/6M$. On the other hand, $p_{u,k} = 0$ for all $k \neq j$, and $\hat{p}_{u,k} \leq \epsilon/12M$. With $\hat{p} \geq \frac{5}{6}\frac{\epsilon}{M}$, this implies $|\hat{p}_{u,k} - \hat{p}| \geq \frac{2}{3}\frac{\epsilon}{M}$. Thus indeed $I_{u,j}(\mathbf{x})$ is included in S. □

The bound for the sample complexity follows by noting that the algorithm asks $O(m^2n^2)$ queries and the finest tolerance is $\Omega(\epsilon/m^2n)$. □

As noted above, Algorithm Forest is more efficient in terms of sample size than a direct reduction to learning Boolean conjunctions because we were able to use larger tolerances. When looking at statistical query algorithms, one may also try to increase their efficiency by *decreasing the number of statistical queries asked*.

In our case it appears to be possible to decrease the number of queries by performing binary searches on the probabilities in order to determine which one should be included in the hypothesis conjunction. Indeed, one can give an algorithm that uses $O((mn + m^2)\log n)$ statistical queries with tolerances of the same order of magnitude as Algorithm Forest.

However, it is an interesting feature of statistical query learning algorithms that this improvement in query complexity does *not* lead to an improvement in sample size for the following reason (also observed by Aslam and Decatur [2]).

The queries of Algorithm Forest are *non-adaptive* in the sense that they can all be asked at the same time, without depending on the results of the other queries. Thus in this case one can use Hoeffding's Inequality (described in Appendix A) to *evaluate each query simultaneously* from a *single* sample that has size depending only *logarithmically* on the number of queries. On the other hand, the queries of the binary search algorithm are *adaptive*—that is, the outcome of a query will determine which query is asked next. So the queries are *dependent* and we need a different sample for the evaluation of each query. Thus in this case the sample size will depend *linearly* on the number of queries asked by the algorithm.

7 The Recursive Case

In this section, we show how to learn a linear closed recursive Horn clause that is determinate and constant depth together with a simple base case and ground background knowledge. Our concepts will be of the form

$$\begin{aligned} &P(\mathbf{x}) \leftarrow B_1, \ldots, B_\ell, P(\mathbf{y}) \\ &P(\mathbf{x}) \leftarrow D_1, \ldots, D_s \ , \end{aligned} \tag{5}$$

where s is constant and the D_i are atoms of the background predicates (which, recall, include Equal) instantiated with variables from the head and/or constants from a specified polynomial-size set of constants. (For learning problems involving $\mathbb{N}$, that set of constants might be $\{0, 1\}$; for learning problems involving lists that set might contain only the empty list.) For example, the base case for append would be $\text{append}(x, y, z) \leftarrow \text{Equal}(y, \text{nil}), \text{Equal}(x, z)$.

We call base cases meeting these assumptions *simple base cases*. Simple base cases are actually a special case of Cohen's "base case oracle" [3] that we believe can handle many practical logic programs. (Our assumptions can be slightly weakened; for instance, we could allow a constant number of such clauses.)

In this section, for $\text{BOTTOM}^*_d(P, m, M)$ we will just write BOTTOM, and we denote the literals of BOTTOM by $L_1, \ldots, L_k$. (Notice that k is polynomial in all the relevant parameters.) We will write $\text{BOTTOM}(\mathbf{z})$ for the BOTTOM clause built starting with the variables $\mathbf{z}$ as the arguments to the target predicate. We write $\mathbf{z}'$ for the full set of variables in $\text{BOTTOM}(\mathbf{z})$, and with some abuse of notation we can write $\mathbf{z} \subset \mathbf{z}'$. Notice that by determinateness the values of $\mathbf{z}$ determine the values of $\mathbf{z}'$. (Of course, some of the values of $\mathbf{z}'$ may be undefined.)

In the concept definition (5), each $B_i = L_{j_i}$, for some $1 \leq j_i \leq k$. We use $\mathcal{L}$ to denote the conjunction of all the nonrecursive literals in the recursive clause, and BASE to denote the body of the nonrecursive clause. Finally, we use $\mathbf{x}^*$ to denote the arguments to the first recursive call of P starting from $P(\mathbf{x})$, for some $\mathbf{x}^* \subset \mathbf{x}'$. Thus we can rewrite (5) as a logic program of the form

$$\begin{aligned} &P(\mathbf{x}) \leftarrow \mathcal{L}(\mathbf{x}'), P(\mathbf{x}^*) \\ &P(\mathbf{x}) \leftarrow \text{BASE}(\mathbf{x}) \ . \end{aligned} \tag{6}$$

We define $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{x}^{(i)} = (\mathbf{x}^{(i-1)})^*$.

Now we are ready to discuss learning recursive programs in the presence of noise. Let the target logic program, of the form of (6), be Π.

Observe that if $P(\mathbf{a})$ and $\overline{\text{BASE}(\mathbf{a})}$ hold, then $\mathcal{L}(\mathbf{a}')$ must be true. More generally, $P(\mathbf{a}), \overline{\text{BASE}(\mathbf{a}^{(0)})}, \ldots, \overline{\text{BASE}(\mathbf{a}^{(i)})}$ together imply $\mathcal{L}((\mathbf{a}^{(0)})'), \ldots, \mathcal{L}((\mathbf{a}^{(i)})')$. Since $P(\mathbf{a})$ is a positive example, Π must derive $P(\mathbf{a})$. Now $\overline{\text{BASE}(\mathbf{a}^{(0)})}, \ldots,$

$\overline{\mathrm{BASE}(\mathbf{a}^{(i)})}$ implies that Π cannot derive $P(\mathbf{a})$ with i or fewer recursive calls. Hence the recursive clause of Π must succeed on the first i calls, and so $\mathcal{L}((\mathbf{a}^{(0)})')$, $\ldots$, $\mathcal{L}((\mathbf{a}^{(i)})')$ must succeed.

Let us use $\mathbf{z}'(j)$ to denote the subset of the variables $\mathbf{z}'$ that are arguments to literal L_j of the recursive clause of Π. Now if $P(\mathbf{a})$ is positive and L_j occurs in Π's body, then for all i, the argument above shows that

$$\left(\overline{\mathrm{BASE}(\mathbf{a}^{(0)})} \wedge \cdots \wedge \overline{\mathrm{BASE}(\mathbf{a}^{(i)})}\right) \Rightarrow \left(L_j((\mathbf{a}^{(0)})'(j)) \wedge \cdots \wedge L_j((\mathbf{a}^{(i)})'(j))\right) \ . \tag{7}$$

An equivalent formulation of (7) would keep only the last L_j. Switching to probability notation this gives

$$\Pr\left(P(\mathbf{x}) \wedge \bigvee_{i=0}^{\infty}\left(\bigwedge_{\ell=0}^{i} \overline{\mathrm{BASE}(\mathbf{x}^{(\ell)})} \wedge \overline{L_j((\mathbf{x}^{(i)})'(j))}\right)\right) = 0 \ .$$

Here, as in Sect. 5, falseness is taken to mean false or undefined.

Now for any literal $L_j \in \mathrm{BOTTOM}$, put

$$p_j = \Pr\left(P(\mathbf{x}) \wedge \bigvee_{i=0}^{\infty}\left(\bigwedge_{\ell=0}^{i} \overline{\mathrm{BASE}(\mathbf{x}^{(\ell)})} \wedge \overline{L_j((\mathbf{x}^{(i)})'(j))}\right)\right) \ . \tag{8}$$

We say that literal L_j is *harmful* if $p_j > \epsilon/2k$.

Lemma 6. *Let C be a concept represented by a closed linear recursive logic program Π of the form given by (6) that is determinate and constant depth. Let the hypothesis H have a logic program Π' of the form*

$$\begin{aligned} P(\mathbf{x}) &\leftarrow \mathcal{A}(\mathbf{x}), P(\mathbf{x}^*) \\ P(\mathbf{x}) &\leftarrow \mathrm{BASE}(\mathbf{x}) \ , \end{aligned}$$

where $P(\mathbf{x}^*)$ and $\mathrm{BASE}(\mathbf{x})$ are correct, and $\mathcal{A}$ contains all the literals in $\mathcal{L}$ and no harmful literals. Then $\Pr(C \bigtriangleup H) \le \epsilon/2$.

Proof. Since $\mathcal{A}$ contains all the literals in $\mathcal{L}$, $H \subseteq C$. So we need to show only that $\Pr(C \setminus H) \le \epsilon/2$. Without loss of generality, say that the literals in $\mathcal{A}$ but not in $\mathcal{L}$ are $L_1, \ldots, L_r$.

$$\Pr(\mathbf{x} \in C \setminus H) = \Pr\left(\bigvee_{i=0}^{\infty}(\Pi \text{ derives } \mathbf{x} \text{ in } i \text{ steps but } \Pi' \text{ doesn't derive } \mathbf{x})\right) \ . \tag{9}$$

Notice that Π and Π' make the same recursive calls, so BASE holds for both programs at exactly the same time. Thus the only way $\mathbf{x}$ can be in $C \setminus H$ is for Π' to have an extra literal that disqualifies $\mathbf{x}$ at some level of the recursion before Π accepts $\mathbf{x}$. The quantity in (9) is equal to

$$\Pr\left(\mathbf{x} \in C \wedge \bigvee_{i=0}^{\infty}\left(\bigwedge_{\ell=0}^{i} \overline{\mathrm{BASE}(\mathbf{x}^{(\ell)})} \wedge \overline{L_j((\mathbf{x}^{(i)})'(j))}\right) \text{ for some } j \le r\right) \ . \tag{10}$$

Now the quantity in (10) is at most $(\epsilon/2k)r \le \epsilon/2$, completing the proof. □

Lemma 6 proves the correctness of the heart of the following algorithm, which we only sketch. The formal details are broadly similar to the nonrecursive case. Recall that BOTTOM is $P(\mathbf{x}) \leftarrow L_1, \ldots L_k$. Besides the usual inputs, the following algorithm must also be given a set of constants to use in constructing possible base cases.

Algorithm Rec.

1. Compute $S :=$ BOTTOM. Put $T = \epsilon/4k$.
2. Compute the sets U_B of all possible base cases and U_r of all possible instantiations of the recursive literal with variables from BOTTOM.
3. Using one sample of examples, run the following algorithm on that sample for all possible choices of a base case from U_B and a recursive literal from U_r. (Each run will output its own hypothesis.)
   - For each nonrecursive literal L_j of the recursive clause, determine an estimate $\hat{p}_j$ of p_j as in (8), with accuracy T .
     If $\hat{p}_j > T$, delete it and all literals that it supports. (This is very similar to the algorithm of the nonrecursive case in Sect. 5.)
4. Draw a new sample of examples. Test each hypothesis from the previous stage on this sample. Output the hypothesis with the lowest error rate.

Theorem 7. *Algorithm* Rec *is a polynomial statistical query algorithm for the class of of determinate, constant-depth closed linear recursive logic programs with simple base cases and ground background knowledge. Hence those logic programs are polynomially* PAC *learnable with random classification noise.*

Remark. Recall that for recursive logic programs, the definition of *polynomial PAC* learnability, given in Sect. 2.2, is different from the one used elsewhere in this paper.

Proof idea. To apply Lemma 6, we must show that for the right choice of the recursive literal and the base case, our algorithm included all literals in the body of Π's recursive clause, and no harmful literals. This is true because if L_j occurs in the body of Π's recursive clause, then $p_j = 0$, and our estimate $\hat{p}_j$ must be less than or equal to $\epsilon/4k$. The same applies to literals that support L_j. Conversely, for any harmful literal L_h, we know $p_h > \epsilon/2k$, and therefore $\hat{p}_h > \epsilon/4k$.

We also must show that the statistical query can be evaluated in polynomial time. A simple counting argument shows that there is a polynomial bound on the depth of recursion of Π such that if Π goes beyond this bound, then it will never succeed [3]. Thus the limit ∞ in (8) can be replaced by a polynomial bound.

In Step 4, we will need the correct Step 3 hypothesis to have error $\epsilon/2$. The reason for this is explained in Appendix A.2.

Beyond the use of Lemma 6, the key points are that the sets U_B and U_r are of polynomial size. Recall that BOTTOM has only polynomially many literals.

The number of choices for the recursive literal ($|U_r|$) is the number of ways of choosing m variables from all the variables mentioned in BOTTOM. The arity of the background predicates is a constant, so the total number of variables in BOTTOM, which is at worst the number of literals times the number of variable per literal, is polynomial. We have assumed that m is constant.

The base case contains only up to some constant s literals, and every literal is chosen from the background knowledge and instantiated with either some of the m variables in the head and/or constants from a polynomial-size set. Thus U_B is clearly polynomial in size. □

8 Remarks

8.1 Learning Theory Results on Noise

As noted in the introduction, we have studied random classification noise in some detail because for that noise model, there is a nontrivial procedure for coping with the noise.

We now present a brief overview of other models of noise in the learning theory literature. Some models that make the most sense from the point of view of ILP include *malicious* noise [23], *malicious classification* noise [21], and *random attribute* noise [20, 21].

With malicious noise, an omnipotent, omniscient adversary (i.e., the adversary has unlimited computing time, and knowledge of the learner's algorithm) gets to alter a small fraction of the training examples in any way he chooses. Kearns and Li [13] show that the fraction of the examples the adversary alters (i.e., the noise rate) must be less than $\epsilon/(1+\epsilon)$ (where ϵ as usual is the desired accuracy) for PAC learning to be possible. They also provide a meta-algorithm that converts any ordinary PAC learning algorithm into one that tolerates a small amount of malicious noise. For conjunctions, that meta-algorithm can tolerate a noise rate up to $O\left(\epsilon/n\right)$, where n is the number of variables. For recent work on the $\Theta(1/n)$ gap between that rate and the lower bound, see [5, 18]. Džeroski discusses the application of the model to ILP [6].

In the case of malicious classification noise, the omnipotent, omniscient adversary can alter the label of some fraction of the training data, but not the instances. The difference between this model and random classification noise is that the adversary may choose not to alter the labels of some examples. For example, malicious classification noise can model the case where "boundary" instances are frequently misclassified but "core" instances are never misclassified. The statistical query method relies heavily on the independence of instances and label noise, and, as far as we know, does not work at all for random misclassification noise. (If computational efficiency is not an issue, then it suffices to choose the hypothesis that has minimal disagreement with the training data [21], but this is almost always NP-hard.)

With random attribute noise, the label is always accurate, but the value of each Boolean variable in each training instance is flipped with some probability. If this noise is *uniform*, then each variable is altered with the same probability; with *product* random attribute noise, there is a distinct probability p_i of each variable i being flipped. For some simple classes including conjunctions, large amounts of uniform random attribute noise can be tolerated [9, 20]. For the more realistic case of product random attribute noise, the noise rate (i.e., maximum probability that any bit is flipped) must be less than ϵ for PAC learning to be possible [9]. Thus, somewhat surprisingly, the PAC algorithms situation for product random attribute noise is essentially the same as for malicious noise.

8.2 Further Research

The bounds of Sect. 6 can perhaps be improved by considering Step 3 of Algorithm Forest more carefully. One could also consider the query complexity of the algorithm mentioned at the end of Sect. 6 in more detail.

It is an open problem whether a monotone Boolean conjunction of n variables is polynomially PAC-learnable with random classification noise using sample size *less than quadratic* in n. If computation time is not taken into account, then it is sufficient to take a sample of size linear in n and output a hypothesis that makes the least number of errors in the training set [1].

Another topic for further work is noise models between the random and the malicious classification models. For example, Kearns [14] considers the case where the noise rate may be different for each round of drawing a random example, but it is still assumed that in each round the error probability is the same for each instance. He showed that statistical query algorithms generalize to this model. It would be interesting to study models where the error probabilities of different instances may differ, as appears to be the case in practice.

Another issue is to study types of noise that are specific to learning in predicate logic, such as noise in the background knowledge. Gennaro [8] presents an initial proposal in this direction.

Acknowledgments

We thank an anonymous referee for calling our attention to references [6] and [8]. We thank Jorge Lobo for useful discussions.

References

1. D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.
2. J. A. Aslam and S. E. Decatur. Improved noise-tolerant learning and generalized statistical queries. Technical Report TR-17-94, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, 1994.
3. W. W. Cohen. Pac-learning recursive logic programs: efficient algorithms. *J. AI Research*, 2:501–539, 1995.
4. W. W. Cohen. Pac-learning recursive logic programs: negative results. *J. AI Research*, 2:541–573, 1995.

5. S. E. Decatur. Statistical queries and faulty PAC oracles. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 262–268. ACM Press, New York, NY, 1993.
6. S. Džeroski. Learning first-order clausal theories in the presence of noise. In *Proc. 5th Scandinavian Conf. on Artificial Intelligence*, Amsterdam, 1995. IOS Press.
7. S. Džeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 128–135. ACM Press, New York, NY, 1992.
8. R. Gennaro. PAC-learning PROLOG clauses with or without errors. Tech. Memo 500, MIT Laboratory for Computer Science, 1994.
9. S. A. Goldman and R. H. Sloan. Can PAC learning algorithms tolerate random attribute noise? *Algorithmica*, 14:70–84, 1995.
10. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar. 1963.
11. T. Horváth and G. Turán. Learning logic programs with structured background knowledge. In L. De Raedt, editor, *5th Int. Workshop on Inductive Logic Programming*, pages 53–76, 1995. Also in *Advances in Inductive Logic Programming* (ed. L. De Raedt). IOS Press, 1996, pages 172–191. (IOS Frontiers in AI and Appl.).
12. M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 392–401. ACM Press, New York, NY, 1993.
13. M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22:807–837, 1993.
14. M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, Cambridge, Massachusetts, 1994.
15. N. Lavrač and S. Džeroski. Inductive learning of relations from noisy examples. In S. H. Muggleton, editor, *Inductive Logic Programming*, pages 495–514, London, 1992. Academic Press.
16. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, New York, 1994.
17. N. Lavrač, S. Džeroski, and I. Bratko. Handling imperfect data in inductive logic programming. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 48–64. IOS Press, 1996.
18. Y. Mansour and M. Parnas. On learning conjunctions with malicious noise. In *Israel System and Theory Computer Symposium (ISTCS 96)*, 1996. (To appear).
19. S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Inductive Logic Programming*, pages 281–298. Academic Press, 1992.
20. G. Shackelford and D. Volper. Learning k-DNF with noise in the attributes. In *Proc. 1st Annu. Workshop on Comput. Learning Theory*, pages 97–103, San Mateo, CA, 1988. Morgan Kaufmann.
21. R. H. Sloan. Four types of noise in data for PAC learning. *Inf. Process. Lett.*, 54:157–162, 1995.
22. A. Srinivasan, S. H. Muggleton, and M. Bain. Distinguishing exceptions from noise in non-monotonic learning. In *Proc. Second International Workshop on Inductive Logic Programming*, Tokyo, Japan, 1992. ICOT TM-1182.
23. L. G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, vol. 1*, pages 560–566, Los Angeles, California, 1985. International Joint Committee for Artificial Intelligence.

A Sample complexity of statistical query simulation

We now derive the sample complexity values specified in (3) and (4).

A.1 Bound on m_1

We need to do a sensitivity analysis of (2), which we can rewrite as

$$\Pr_D(\chi = 1) = p_1 \Pr_{D_1,\eta}(\chi = 1)\left(\frac{1}{1-2\eta}\right) - p_1\eta\left(\frac{1}{1-2\eta}\right) + \Pr_{D,\eta}(\chi = 1 \wedge x \in X_2) \ . \tag{11}$$

We need to figure out how much error we can allow in each of the three terms of the sum so that the total error in the value of $\Pr_D(\chi = 1)$ is at most α. In fact, we want to know how much additive error in the basic quantities p_1, $\Pr_{D_1,\eta}(\chi = 1)$, $\Pr_{D,\eta}(\chi = 1 \wedge x \in X_2)$, and η we can allow. Notice that all those quantities have values strictly between 0 and 1, although $1/(1-2\eta)$ does not.

Kearns [12, 14] suggests how to find an estimate $\widehat{\frac{1}{1-2\eta}}$ of $1/(1-2\eta)$ that has additive error at most τ. Now consider two quantities $0 < A, B < 1$, and two estimates of them, $\hat{A}$ and $\hat{B}$, such that $A - \tau(1-2\eta_b) \le \hat{A} \le A + \tau(1-2\eta_b)$, and similarly for $\hat{B}$.

What is the maximum additive error in the estimate $\hat{A}\hat{B}\widehat{\frac{1}{1-2\eta}}$ of $AB/(1-2\eta)$? We have

$$\begin{aligned}\hat{A}\hat{B}\widehat{\frac{1}{1-2\eta}} &\le (A + \tau(1-2\eta_b))(B + \tau(1-2\eta_b))\left(\frac{1}{1-2\eta} + \tau\right) \\ &\le AB/(1-2\eta) + 3\tau + 4\tau^2 \ .\end{aligned}$$

We get something smaller for the lower bound, so the additive error in this estimate is at most $3\tau + 4\tau^2$.

Now taking $A = p_1$ and $B = \Pr_{D_1,\eta}(\chi = 1)$, we see that if we estimate A, B, and $1/(1-2\eta)$ to the stated accuracies, then the total additive error in the first term of the sum in (11) would be at most $3\tau + 4\tau^2$. The same error bound holds for the second term. The third term of (11) obviously has additive error at most τ. Thus the total additive error in (11) is at most $7\tau + 8\tau^2$.

Now, if we estimate each of p_1, $\Pr_{D_1,\eta}(\chi = 1)$, η, and $\Pr_{D,\eta}(\chi = 1 \wedge x \in X_2)$ to within additive error $\alpha(1-2\eta_b)/8$, and $1/(1-2\eta)$ to within additive error $\alpha/8$, then the the additive error of our estimate of $\Pr_D(\chi = 1)$ is at most α.

The value of Δ was chosen to give the desired accuracy on η and $1/(1-2\eta)$ [14]; we will make our sample size m_1 big enough to guarantee the other accuracies with high probability.

We are going to estimate all three quantities p_1, $\Pr_{D_1,\eta}(\chi = 1)$, and $\Pr_{D,\eta}(\chi = 1 \wedge x \in X_2)$ for all queries χ from the same large sample of m_1 examples. Thus, the total number of quantities to estimate is at most three times the maximum number of queries; that is, $3|\mathcal{Q}|$. We want the probability that all of those estimates are within $\alpha(1-2\eta_b)/8$ of the true value to be at least $1-\delta/2$. We will achieve this if the probability that any one estimate is more than $\alpha(1-2\eta_b)/8$ away from its true value is at most $\delta/6|\mathcal{Q}|$.

To ensure this, we will need Hoeffding's Inequality [10], which is as follows.

Let $X_1, X_2, \cdots, X_m$ be independent 0-1 random variables each with probability p of being 1. Let $S = \sum_1^m X_i$. Then for any positive t, $\Pr(|S - pm| \geq tm) \leq 2e^{-2mt^2}$.

Thus we need $2e^{-2m_1(\alpha(1-2\eta_b)/8)^2} \leq \delta/(6|\mathcal{Q}|)$, which we can solve for m_1 getting

$$m_1 \geq \frac{32}{\alpha^2(1-2\eta_b)^2} \ln\left(\frac{12|\mathcal{Q}|}{\delta}\right) \quad .$$

As we are about to explain, α should be chosen so that the statistical query algorithm achieves an error rate of at most $\epsilon/2$.

A.2 Bound on m_2

We test all our hypotheses on one sample of m_2 examples from the noisy oracle. For every hypothesis H_i we get an empirical estimate $\hat{\gamma}_i$ of γ_i, which we define to be the probability that H_i disagrees with the label of an example from the random classification noise oracle. We pick a large enough sample so that $|\hat{\gamma}_i - \gamma_i| \leq (1-2\eta_b)\epsilon/4$ with probability at least $1 - \delta/2$.

If a hypothesis has error rate r on noise-free examples, then its error rate on examples with random classification noise will be $\eta + (1-2\eta)r$ [12].

We know that there is a "good" hypothesis H_g with error at most $\epsilon/2$. For this hypothesis, (with high probability)

$$\gamma_g \leq \eta + (1-2\eta)\epsilon/2 + (1-2\eta_b)\epsilon/4 \leq \eta + \frac{3\epsilon(1-2\eta_b)}{4}$$

For any "bad" hypothesis H_b with error greater than ϵ, (with high probability)

$$\gamma_b > \eta + (1-2\eta)\epsilon - (1-2\eta_b)\epsilon/4 \geq \eta + \frac{3\epsilon(1-2\eta_b)}{4}$$

Thus, if our estimates $\hat{\gamma}_i$ all meet our desired accuracy with high probability, and we choose as our final output the hypothesis with the smallest $\hat{\gamma}_i$, then with high probability we will have chosen a hypothesis with error less than ϵ.

We have η_b/Δ possible hypotheses, and $\eta_b < 1/2$, so we have at most $1/2\Delta = 8/\alpha(1-2\eta_b)^2$ hypotheses and estimates. It is sufficient if the probability that any one estimate is too far off is at most $\delta\alpha(1-2\eta_b)^2/8$.

By Hoeffding's Inequality, we need $2e^{-2m_2(\epsilon(1-2\eta_b)/2)^2} \leq \delta\alpha(1-2\eta_b)^2/8$. Solving for m_2 gives

$$m_2 \geq \frac{2}{\epsilon^2(1-2\eta_b)^2} \ln\left(\frac{16}{\delta\alpha(1-2\eta_b)^2}\right) \quad .$$

This article was processed using the LaTeX macro package with LLNCS style