

Learning From a Consistently Ignorant Teacher

Michael Frazier*

Computer Science Dept.
Abilene Christian University
Abilene, TX 79699
frazier@cs.acu.edu

Nina Mishra

Dept. of Computer Science
University of Illinois
Urbana, IL 61801
nmishra@uiuc.edu

Sally Goldman[†]

Dept. of Computer Science
Washington University
St. Louis, MO 63130
sg@cs.wustl.edu

Leonard Pitt[‡]

Dept. of Computer Science
University of Illinois
Urbana, IL 61801
pitt@cs.uiuc.edu

Abstract

One view of computational learning theory is that of a learner acquiring the knowledge of a teacher. We introduce a formal model of learning capturing the idea that teachers may have gaps in their knowledge. In particular, we consider learning from a teacher who labels examples “+” (a positive instance of the concept being learned), “-” (a negative instance of the concept being learned), and “?” (an instance with unknown classification), in such a way that knowledge of the concept class and all the positive and negative examples is not sufficient to determine the labelling of any of the examples labelled with “?”. The goal of the learner is not to compensate for the ignorance of the teacher by attempting to infer “+” or “-” labels for the examples labelled with “?”, but is rather to learn (an approximation to) the ternary labelling presented by the teacher. Thus, the goal of the learner is still to acquire the knowledge of the teacher, but now the learner must also identify the gaps. This is the notion of learning from a consistently ignorant teacher. We present general results describing when known learning algorithms can be used to obtain algorithms that learn from a consistently ignorant teacher. We investigate the learnability of a variety of concept classes in this model, including monomials, monotone DNF formulas, Horn sentences, decision trees, DFAs, and axis-parallel boxes in Euclidean space, among others. Both learnability and non-learnability results are presented.

“Consistency requires you to be as ignorant today as you were a year ago.”
– Bernard Berenson (1865-1959)

*Supported in part by NSF Grant IRI-9014840, and by NASA grant NAG 1-613. This work was completed while at the University of Illinois at Urbana-Champaign, and at Southern University at New Orleans.

[†]Supported in part by NSF Grant CCR-9110108 and an NSF NYI Grant CCR-9357707 with matching funds provided by Xerox Corporation, Palo Alto Research Center.

[‡]Supported in part by NSF Grant IRI-9014840.

1 Introduction

Building machines that learn from experience is an important research goal in artificial intelligence. Considerable attention is devoted to the theoretical study of machine learning for the domain of *concept learning*. The general concept learning problem is to learn to discriminate between objects that satisfy some unknown rule, or “concept”, and those that do not. More formally, we assume a space \mathcal{X} of possible *examples*, and that some subset $f \subseteq \mathcal{X}$ cleaves \mathcal{X} into *positive* examples $x \in f$, and *negative* examples $x \notin f$. The unknown set f is referred to as the *target concept*, and is often assumed to come from some known *concept class* \mathcal{C} of possible target concepts. Equivalently, we view f as a boolean-valued function on \mathcal{X} , with $f(x) = “+”$ (respectively, “-”) indicating that x is a positive (respectively, negative) example of f . Typically, a learning algorithm obtains examples of f either randomly from nature, or from a teacher, and is told which examples are positive and which are negative. Often the learning algorithm is also allowed to pose a *membership query* which is an example x of its own choice, in response to which a teacher classifies x as either a positive or negative example. Such membership queries model not only the interaction with a human expert, but perhaps also the careful crafting of experiments by a learning agent in order to observe the response of the environment.

Variations of the basic theoretical models of concept learning allow for the possibility that the information given to the learning algorithm does not come from an “omniscient” teacher, but instead may be inaccurate. Nonetheless, in most of these variations, it is assumed that underlying the inaccurate information is some “correct” classification of examples as positive or negative examples of the concept to be learned. Thus, it is assumed that there is a well-defined border separating positive examples from negative ones. In practice, though, classification is often unclear. For example, an algorithm designed to read handwritten cheques will likely encounter many handwritten characters that look somewhat like a “4”, and somewhat like a “9”. In such cases, where even an expert does not have the knowledge to classify all objects, determining *which* objects are unclassifiable seems at least as important as determining the classification of objects which *are* classifiable. From the learner’s perspective, the regions of the example space that defy classification create a blurry border between the positive and negative examples that the learner must determine.

In this paper we introduce a formal learning model in which the teacher (or environment) with which the learner interacts has incomplete information about the target concept due to intrinsic uncertainty, or due to gaps in the teacher’s knowledge. A key requirement we place on the teacher is that all examples labeled with “?” (indicating unknown classification) are consistent with the teacher’s background knowledge about the concept class \mathcal{C} from which the target concept is known to belong. In particular, the classification of any example labeled with “?” should not be determinable from the positive and negative examples, and knowledge of the concept class \mathcal{C} . (Thus the teacher

is “consistently” ignorant.) The goal of the learner is similar to that for standard learning models: construct a reasonably accurate approximation to the knowledge of the teacher. However, in this case, the learner must construct a *ternary* classifier (i.e. with values $\{+, -, ?\}$) that, with high probability, classifies most examples exactly as the teacher does. We call such a ternary classifier a *blurry* concept, to distinguish it from standard concepts, which are typically boolean-valued classifiers. By “learning a blurry concept” we mean the problem of learning a (standard) concept from a consistently ignorant teacher.

We first review, in Section 2, the standard “PAC” learning model and discuss related work. In Section 3 we give precise definitions for the blurry concept class $\mathcal{C}_?$ induced by a concept class \mathcal{C} containing nonblurry (boolean-valued) classifiers; we consider basic structural properties of the blurry concepts in $\mathcal{C}_?$, and show that each such concept $f_? \in \mathcal{C}_?$ is naturally representable by unions and intersections of boolean-valued concepts from \mathcal{C} . This representation as unions and intersections of boolean-valued concepts leads to an exact characterization of blurry concepts of any class $\mathcal{C}_?$ as an *agreement* of boolean-valued concepts from \mathcal{C} . Intuitively, an agreement of a set $F \subseteq \mathcal{C}$ of concepts is a *ternary* function, that on input x , if all $f \in F$ agree on the classification of x , outputs the label they agree on and outputs “?” otherwise. Hence, the problem of learning concepts from \mathcal{C} from a consistently ignorant teacher (or learning the ternary blurry concepts from $\mathcal{C}_?$) is exactly the problem of learning agreements of (boolean-valued) concepts from \mathcal{C} .

We show in Section 4 that for any concept class \mathcal{C} for which efficient PAC-learning algorithms are known, these algorithms can be used to build an efficient algorithm for learning the agreement of *nested* concepts from \mathcal{C} . For the problem of learning the agreement of concepts from \mathcal{C} that are not necessarily nested, we show that if the intersection and union of arbitrarily many concepts from \mathcal{C} are learnable, then \mathcal{C} is learnable from a consistently ignorant teacher. As a corollary of these general techniques, we show that blurry (propositional) Horn clauses and blurry CLASSIC sentences (a first-order language for representing knowledge) are learnable. We also show, for example, that, with some restrictions, blurry monomials, blurry monotone DNF, blurry decision trees and blurry DFAs are learnable.

While there are not many concept classes \mathcal{C} for which efficient algorithms are known for learning unions and intersections of concepts from \mathcal{C} , under certain conditions it is still possible to learn the agreements of concepts from \mathcal{C} . For example, consider a class \mathcal{C} for which the intersection of concepts from \mathcal{C} is learnable, yet there is no known efficient algorithm to learn the union of concepts from \mathcal{C} . In some cases it may still be possible to efficiently learn \mathcal{C} from a consistently ignorant teacher by using information gained by learning the intersection of concepts in the agreement to aid in learning the union of these concepts. The learner’s ability to use intersection (union) information to obtain positive results for learning unions (intersections) of concepts from classes for which no algorithms are currently known is intriguing. In Section 4.3 we give an algorithm to learn the

agreement of s “boxes” in d -dimensional Euclidean space in time that depends polynomially in s and 2^d , assuming that there is at least one point common to all of the boxes. To do so, we learn both the intersection and union of such boxes. While there are efficient algorithms for learning the intersection of boxes, all known algorithms to learn the union of such boxes using random examples and membership queries requires time at least s^d or d^s , and hence are polynomial-time algorithms only for one of s, d constant. However, our algorithm for learning the agreement allows d to be as large as $\log s$. Our advantage comes from the ability to use information about the nonempty intersection (in this case, a single point is sufficient) in learning the union. To illustrate the limits of this approach, in Section 5 we show that learning blurry l -term DNF formulas, blurry decision trees, and blurry Horn sentences, is as hard as learning DNF formulas in standard learning models. The learnability of DNF formulas remains a centrally studied unsolved problem; thus, while l -term DNF formulas, decision trees, and Horn sentences are learnable in standard models [Ang87a, Bsh95, AFP92], learning these types of concepts from a consistently ignorant teacher would appear to be much more difficult. We also show in Section 5 that the problem of learning blurry DFAs is intractable, given standard cryptographic assumptions. Once again, while DFAs are learnable in standard models [Ang87b], their blurry counterparts seem much harder to learn.

In Section 6 we consider the extension of the model of a consistently ignorant teacher to the exact learning model using equivalence (and membership) queries, and demonstrate that analogous results hold in this more demanding learning model.

We conclude with a summary and some open questions in Section 7.

2 Background and Related Work

Boolean formulas: Many of our results are in the boolean domain. Let $V = \{v_1, \dots, v_n\}$ be a set of boolean variables. A *literal* is a boolean variable v_i or its negation \bar{v}_i . *Monomials* are propositional boolean formulas that can be expressed as a conjunction (AND) of (negated or unnegated) literals. *DNF formulas* are disjunctions (ORs) of monomials (also called *terms*), and *k -DNF formulas* are DNF formulas where each term contains at most k literals. A *Horn clause* is a 1-DNF formula containing at most one unnegated literal. A *Horn sentence* is a conjunction (AND) of Horn clauses. *CNF formulas* are conjunctions (ANDs) of 1-DNF formulas (also called *clauses*), and *k -CNF formulas* are CNF formulas where each clause contains at most k literals. A DNF or CNF formula is *monotone* if it contains only unnegated literals. A formula is *unate* if no variable appears both negated and unnegated. A *decision tree* is a rooted binary tree with internal vertices labeled with variables, and with leaves labeled “+” or “−”.

Boolean formulas over n variables as described above naturally represent boolean functions from the *example space* $\mathcal{X}_n = \{0, 1\}^n$ to $\{0, 1\}$ in the usual way. For example, if m is a monomial, and

$x \in \mathcal{X}_n$, then $m(x) = 1$ iff for each literal $\ell \in m$, the i th bit x_i of x is 1 if $\ell = v_i$, and $x_i = 0$ if $\ell = \bar{v}_i$. Similarly, $f(x) = 1$ for f a DNF formula iff for at least one term t of f , $t(x) = 1$. The value of a decision tree on a boolean input vector x is the label of the leaf reached by following a branch from the root, and branching left at a vertex with label v_i if $x_i = 0$, or branching right if $x_i = 1$.

We assume basic familiarity with the definitions of deterministic finite automata (DFAs) [HU79]. The CLASSIC description logic is a first-order logic used for representing objects and their relationships. A description of CLASSIC is beyond the scope of this paper; we note only that positive results for the learnability of CLASSIC sentences have recently been given [CH94c, CH94b, FP94].

It is helpful to view the example space \mathcal{X}_n as a lattice with componentwise “or” and “and” as the lattice operators. The top element is the vector $\{1\}^n$ and the bottom element is the vector $\{0\}^n$. The elements are partially ordered by \leq , where $x \leq y$ if and only if for all i , $x_i \leq y_i$. If $x \leq y$, we say, x is *below* y or y is *above* x .

Standard learning models: In Valiant’s distribution-free, or probably approximately correct (PAC) learning model [Val84], the learner’s goal is to infer how an unknown *target function* f , chosen from some known concept class \mathcal{C} , classifies all examples from the domain \mathcal{X} . Often \mathcal{C} is decomposed into subclasses \mathcal{C}_n according to some natural dimension measure n . For example, in the boolean domain, n is the number of variables. Let \mathcal{X}_n denote the set of examples to be classified for each problem of size n , and let $\mathcal{X} = \bigcup_{n \geq 1} \mathcal{X}_n$ denote the *example space*. We say each $x \in \mathcal{X}$ is an *example*. For each $n \geq 1$, we define each $\mathcal{C}_n \subseteq 2^{\mathcal{X}_n}$ to be a *family of concepts* over \mathcal{X}_n , and $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$ to be a *concept class* over \mathcal{X} . For $f \in \mathcal{C}_n$ and $x \in \mathcal{X}_n$, $f(x)$ denotes the classification of f on example x . That is, $f(x) = “+”$ if and only if $x \in f$. We say that x is a positive example of f if $f(x) = “+”$ and x is a negative example of f if $f(x) = “-”$. We use the symbols “+” and “1” interchangeably (likewise for “-” and “0”).

Because learning algorithms need a means for representing the functions to be learned, typically associated with each concept class \mathcal{C} is a language \mathcal{R} over a finite alphabet, used for representing concepts in \mathcal{C} . Each $r \in \mathcal{R}$ denotes some $f \in \mathcal{C}$, and every $f \in \mathcal{C}$ has at least one representation $r \in \mathcal{R}$. For example, arbitrary boolean functions may be represented in the language of DNF formulas, in CNF formulas, or using the representation language of decision trees. Often, we relax the distinction between the representations and the concepts that they induce. Thus, the class of “monomials” is taken to mean both the boolean formulas that are conjunctions of literals, as well as the boolean functions that they represent. Each concept $f \in \mathcal{C}_n$ has a *size* denoted by $|f|$, which is the representation length of the shortest $r \in \mathcal{R}$ that denotes f . Thus, the choice of a representation language \mathcal{R} for a concept class \mathcal{C} simultaneously gives a language for describing concepts, and a measure of the complexity of a given concept $f \in \mathcal{C}$. An efficient learning algorithm is required to learn in time polynomial in $|f|$ (and other parameters).

To obtain information about an unknown target function $f \in \mathcal{C}_n$, the learner is provided access to labeled (positive and negative) examples of f , drawn randomly according to some unknown target distribution D over \mathcal{X}_n . The learner is also given¹ as input $0 < \epsilon, \delta < 1$, and an upper bound s on the size of f . The learner’s goal is to output, with probability at least $1 - \delta$, the description of a function h that has probability at most ϵ of disagreeing with f on a randomly drawn example from D (thus, h has *error* at most ϵ). If such a learning algorithm \mathcal{A} exists (that is, an algorithm \mathcal{A} meeting the goal for any $n \geq 1$, any target concept $f \in \mathcal{C}_n$, any target distribution D , any $\epsilon, \delta > 0$, and any $s \geq |f|$), we say that \mathcal{C} is *PAC-learnable*. We say that a PAC learning algorithm is a polynomial-time (or efficient) algorithm if the number of examples drawn and computation time are polynomial in $n, s, 1/\epsilon$, and $1/\delta$. For any particular instance of a learning problem, the number of variables n is given. For simplicity, we henceforth drop the subscript “ n ”, and write \mathcal{C} and \mathcal{X} instead of \mathcal{C}_n and \mathcal{X}_n , noting that all algorithms run in time polynomial in n .

In the above definition, we have not specified what a “description of a function h ” is. In the literature, an algorithm is said to be a *proper* learning algorithm, if the hypothesis h is always chosen from the description language \mathcal{R} associated with the concept class. On the other hand, an *improper* (or representation-independent) learning algorithm may output *any polynomial-time algorithm* as a hypothesis. The less constrained model of improper learning is equivalent to polynomial “PAC-predictability” [HLW94, HKLW91]. Throughout this paper, we are concerned mostly with improper learning, and the default interpretation of “learnable” should be that of improper learning.

A well-investigated alternative model of learning is that of *exact learning from equivalence queries* [Ang88]. In this model, the learner proposes as a hypothesis some $h \in \mathcal{C}$, and in response is told “yes” if $h = f$, or otherwise is given a counterexample x such that $h(x) \neq f(x)$. There is no distribution on examples; the learner is required to exactly identify f (obtain a “yes” answer) in time (and queries) polynomial in n (the length of the counterexamples) and s (the bound on the size of the representation of the target f), regardless of the choice of target function and sequence of (adversarially chosen) counterexamples. It is known that any class learnable exactly from equivalence queries can be learned in the PAC setting, via a simple transformation turning an algorithm in the former setting to one in the latter [Ang88]. The converse does not hold [Blu94].

Related work on learning with membership queries: The PAC and exact learning models are *passive* in that the learner cannot directly affect the type of examples it receives as input — in the PAC setting they are randomly generated, and in the exact setting they are chosen by an adversary. Evidence suggests that only relatively simple types of concepts can be learned passively in this way [Ang90, KV94, PW90]. Consequently, researchers have considered augmenting

¹If the demand of polynomial-time computation below is replaced with expected polynomial-time computation, then the learning algorithm need not be given the parameter s , but could “guess” it instead [HKLW91].

this learning protocol by allowing the learner to perform experiments. In addition to drawing a randomly labeled example (or posing a hypothesis, in the exact model), the learner can perform membership queries, as defined previously, in which it supplies an example $x \in \mathcal{X}$ and is told the value $f(x)$. We refer to these models as the *PAC-memb* and *exact-memb* models. Much work has been directed towards understanding what concept classes are efficiently learnable in each of these membership query models. Classes known to be learnable under one or both of these models include, for example, deterministic finite automata [Ang87b], read-once formulas over various bases [AHK93, BHH95, BHH92], and propositional boolean formulas representable in the following forms: k -term DNF [Ang87a, BR92], read-twice DNF [AP91, Han91, PR95], and Horn sentences [AFP92]. In contrast, Angluin and Kharitonov [AK95] have shown that, under cryptographic assumptions, read-thrice formulas, nondeterministic finite automata, and context-free grammars cannot be learned in the PAC-memb model, and that membership queries do not help in learning the general class of DNF formulas. Recently, techniques have been developed showing that some classes are not learnable in the exact-memb model [AHP92, PR94].

Related work on learning with incomplete information: Most of the work in both the PAC and exact models, both with and without membership queries, assumes that examples are labeled either positive or negative. In these situations the border between the positive and negative examples is well defined. There has been work addressing the issue of mislabeled training examples [AL88, Lai88, Slo88, SV88, SS92, Kea93, KL93, GS95, RR95]. In these situations, the border between the positive and negative examples may appear blurry to the learner, but this is just the result of the noise process that has been applied to the properly labeled example. There has also been some work considering learning from noisy membership queries [GKS93, Sak91].

Angluin and Slonim [AS94] introduced a model of *incomplete membership queries* in which each membership query is answered “don’t know” with a given probability. Furthermore, this information is persistent—repeatedly making a query that was answered “don’t know” always results in a “don’t know” answer. As in their work, one of our goals is to model the situation in which the teacher responding to the learner’s queries is not omniscient. In Angluin and Slonim’s model, since the teacher is *randomly* fallible, there is no guarantee that all of the teacher’s knowledge about the target concept is used in answering queries. To see this we briefly describe some aspects of their approach.

First notice that a monotone boolean formula has the property that in the boolean lattice defined over variable assignments, no negative example can be above any positive example. Thus, if we know that the target formula is a monotone DNF, and we know that a point y in the lattice is a positive example (e.g., $y = \langle 10100 \rangle$), then we can deduce that any point x (e.g., $\langle 11101 \rangle$) above y in the lattice is also a positive example. The algorithm of Angluin and Slonim (call it *AS*) takes

advantage of the fact that the learner can make such a deduction, while the teacher cannot.

While the details of algorithm AS are beyond the scope of this brief discussion, one key aspect of the algorithm involves the simulation of an earlier algorithm of Angluin [Ang88] (call it A) that learns monotone DNF using (complete) membership queries. Whenever a membership query on some point x is made by A , if the randomly fallible teacher responds “don’t know” on membership query x , then the algorithm AS searches below x in the lattice for some point y for which the teacher replies “ y is a positive example”. If such a y is found, then AS deduces that the true label of x is “+”, and returns this to algorithm A , continuing the simulation.

Thus, AS can sometimes compensate for the teacher’s ignorance by deducing what the teacher does not know: In learning a monotone function, any point above a positive example is also positive. This capability is not available to a consistently ignorant teacher. In the context of monotone DNF, the consistency requirement manifests itself as follows: The teacher should know that adding positive attributes to an already positive example yields a positive example. (Dually for negative examples.) In particular, a consistently ignorant teacher would have to label points in the boolean lattice in such a way that any example labeled “?” must have only positive or “?” examples above it, and only negative or “?” examples below it. This renders ineffective any approach similar to that of algorithm AS .

In our view, the notion of an incomplete membership oracle, as realized by a randomly fallible teacher, seems to better model noise than it models incomplete knowledge. Indeed, Angluin and Slonim note that their algorithm for learning monotone DNF with an incomplete membership oracle can be used to learn monotone DNF with random (false negative) one-sided errors.

Sloan and Turán [ST94] consider a variant of [AS94] in which a *limited membership oracle* labels a polynomial number of examples “don’t know”. The learner’s performance is measured only on the examples for which the limited membership oracle knows the answer. Thus, unlike our approach, the way that “don’t know” examples are classified is unimportant.

Other investigations have considered learning concept classes when membership query responses are incorrect (as opposed to “don’t know”): Angluin and Kriķis [AK94], and Angluin [Ang94] consider learning with a bounded number of such erroneous responses, and Frazier and Pitt [FP94] consider learning when such incorrect responses occur randomly with probability at most $\frac{1}{2}$.

In other related work, Kearns and Schapire [KS94] generalized the PAC setting to non-binary values using Haussler’s framework [Hau89]. They define a *p-concept* in which each example $x \in \mathcal{X}$ has some probability $p(x)$ of being classified as positive. In their model, the goal of the learner is to make optimal predictions, or more commonly, to accurately predict $p(x)$ for all $x \in \mathcal{X}$. One way to compare our model to theirs is to consider blurry concepts as *p-concepts*, but in our case, the learner’s goal is only that of determining whether $p(x) = 0$, $p(x) = 1$, or $0 < p(x) < 1$. (If a written numeral is sometimes identified as “4” and sometimes as “9”, the learner just wants to

know this—it does not need to determine what percentage of the population calls the numeral each value.)

Related work on learning boxes: Blumer et al. [BEHW89] present an algorithm to PAC-learn an s -fold *union* of boxes in E^d by drawing a sufficiently large sample of size $m = \text{poly}\left(\frac{1}{\epsilon}, \log \frac{1}{\delta}, s, d\right)$, and then choosing a greedy covering from the boxes consistent with the sample. The number of such boxes considered is shown to be at most $\left(\frac{\epsilon m}{2d}\right)^{2d}$, so, for d constant, this algorithm runs in polynomial time. Long and Warmuth [LW94] present an algorithm to PAC-learn this same class by again drawing a sufficiently large sample and constructing a hypothesis consistent with the sample that consists of at most $s(2d)^s$ boxes. Both the time and sample complexity of their algorithm depend polynomially on $(2d)^s$, $\frac{1}{\epsilon}$, and $\log \frac{1}{\delta}$. Recently, Bshouty et al. [BGMST95] present a noise-tolerant PAC-algorithm to learn any geometric concept defined by a boolean combination of s halfspaces for d constant, and Kwek and Pitt [KP95] give a algorithm to learn in the PAC-memb model the intersection of s halfspaces in d dimensions that has time and sample complexity polynomial in *both* s and d given there is a lower bound on the minimum distance between any positive and any negative point. (See also [BGM95] for earlier work.)

There has also been a lot of work on exactly learning unions of s boxes in the discretized space $\{1, \dots, n\}^d$. Recently, there have been several independent results (using very different techniques) to exactly learn this class using only equivalence queries² with time and sample complexity polynomial in d , s , and $\log n$ for either d constant [CH94a, BGGM94, BCH94, MW95]. or s constant [MW95]. One noteworthy difference is that the algorithms of Maass and Warmuth [MW95] have a sample complexity that is polynomial in $\log n$, s , and d . If the learning algorithm can also use membership queries then there is single algorithm to exactly learn this class in polynomial time when either s or d are constant [GGM94, BGGM94]. (See also [CM92, Che93, GGM94] for earlier work.) There has also been recent work that addresses learning more complex geometric concepts. Bshouty et al. [BGGM94, BCH94] present an algorithm to exactly learn the class of geometric concepts defined by s hyperplanes of known slopes for d constant. Auer et al. [AKMW95] present an algorithm for learning the class of depth two linear threshold circuits with a polynomial number of threshold gates and variables with the fan-in at the input gates bound by a constant.

Our algorithm to PAC-memb learn the *agreement* of s boxes in E^d runs in time polynomial in s , 2^d , $1/\epsilon$, and $1/\delta$. Consequently, the algorithm runs in polynomial time without demanding that one of s and d be constant: s can be arbitrary, and d can be as large as $\Theta(\log s)$. (There is an additional assumption required to prove the result: that the set of positive examples is samplable.)

²The hypotheses used by the equivalence queries in all of these algorithms do not come from the class being learned.

3 The Model of a Consistently Ignorant Teacher

Recall our motivation in the case of learning a boolean formula that is known to be monotone (i.e., has no negations). An “ignorant” representation of such a boolean formula f (call it “ $f_?$ ”) could be obtained by simply changing the classification of some examples to “?”. But the resulting ternary function is not *consistent* with the knowledge that the target formula is monotone if there are examples x and y with x above y in the boolean lattice, and such that $f(x) = “?”$ and $f(y) = “+”$, or such that $f(x) = “-”$ and $f(y) = “?”$. For an arbitrary concept class \mathcal{C} , what must hold for a ternary function to be “consistent” with knowledge of \mathcal{C} ? Following the example for monotone formulas, we require that the label of some point x whose classification according to $f_?$ is “?” should not be deducible from the positive and negative examples of $f_?$ and knowledge that the target function originated from some (boolean-valued) base class \mathcal{C} . In particular, if every (boolean-valued) function $f \in \mathcal{C}$ that agrees with $f_?$ on examples whose labels *are* known (i.e., for which $f_?(x) = “-”$ or “+”), happens to label x as, say, a positive example, then $f_?(x)$ must be +, and not “?”. (And similarly for negative examples.) This consideration is embodied in the following definition of a blurry concept.

Definition 1 Let $f_? : \mathcal{X} \rightarrow \{-, +, ?\}$, and let

$$P = \{x \mid f_?(x) = “+”\}, \quad N = \{x \mid f_?(x) = “-”\}, \quad \text{and} \quad Q = \{x \mid f_?(x) = “?”\}.$$

Then $f_?$ is a blurry concept for \mathcal{C} iff for every $q \in Q$, there exist functions f_- and f_+ in \mathcal{C} such that:

1. for all $x \in P$, $f_-(x) = f_+(x) = “+”$,
2. for all $x \in N$, $f_-(x) = f_+(x) = “-”$, and
3. $f_-(q) = “-” \neq “+” = f_+(q)$.

The blurry concept class $\mathcal{C}_?$ is defined by $\mathcal{C}_? = \{f_? \mid f_? \text{ is a blurry concept for } \mathcal{C}\}$.

The definition for PAC learning blurry concepts is analogous to the definition for PAC learning nonblurry concepts:

Definition 2 The blurry concept class $\mathcal{C}_?$ is PAC learnable (alternatively, \mathcal{C} is PAC learnable from a consistently ignorant teacher) iff there exists an algorithm $\mathcal{A}_?$ such that for all blurry concepts $f_? \in \mathcal{C}_?$, for any fixed, unknown distribution D over \mathcal{X} , and on input of ϵ, δ with $0 < \epsilon, \delta < 1$, and upper bound s for $|f_?|$, $\mathcal{A}_?$ draws examples according to D which are labeled according to $f_?$ (hence, by a consistently ignorant teacher) and $\mathcal{A}_?$ outputs a hypothesis $h_? : \mathcal{X} \rightarrow \{-, +, ?\}$ such that with probability at least $1 - \delta$, $D(\{x : h_?(x) \neq f_?(x)\}) \leq \epsilon$. The blurry concept class $\mathcal{C}_?$ is polynomially

PAC learnable if the above holds, and in addition, $\mathcal{A}_?$ is a polynomial-time algorithm: the number of examples drawn and computation time are polynomial in $s, n, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$.

We say that $\mathcal{C}_?$ is PAC-memb learnable (alternatively, \mathcal{C} is PAC-memb learnable from a consistently ignorant teacher) iff some $\mathcal{A}_?$ satisfies the above conditions, where $\mathcal{A}_?$ may also pose membership queries x , and is told the value of $f_?(x)$.

Comment: Note that one way a hypothesis h can disagree with f is if $f_?(x) = \text{"?"}$ and $h(x) \neq \text{"?"}$. Thus, “?” does *not* mean “don’t care”.

Except where otherwise noted, all of our results hold in a more demanding model of learning from a consistently ignorant teacher in the exact setting from equivalence and membership queries. We define this model and discuss some of the changes necessary in Section 6.

In the next several sections, we consider only the PAC-memb model for learning from a consistently ignorant teacher. In what follows, when we say “learnable” we mean, unless otherwise indicated, “polynomial time PAC-memb learnable”. It is easily shown that the problems we attack are hard without membership queries, where “hard” means at least as hard as standard open problems (e.g., DNF) in learning theory.

It should also be observed that in the definition, *both* random examples and membership queries may provide examples with “?” labels. This models a setting where examples are provided by nature at random, but the learner must query a teacher for the correct classification. In the literature on incomplete membership queries, a useful trick of the learner is to use the correctly labeled random examples to compensate for holes in the teacher’s knowledge. In our setting this is not possible. On the other hand, we do not seek a boolean classifier that approximates some underlying “true” concept, but rather we seek to learn how to classify exactly as the teacher does.

Finally, in the definition above, we have neglected to describe how the “size” $|f_?|$ of a blurry concept is measured. In the case of boolean-valued concepts, the class \mathcal{C} is usually a class of *representations* of functions, and thus the size of $f \in \mathcal{C}$ is the fewest symbols needed to represent f . In the case of blurry concepts, we have defined a set $\mathcal{C}_?$ of ternary-valued *functions*, and it is not clear how to define a natural and appropriate means for representing the blurry concepts $f_? \in \mathcal{C}_?$. We now address this issue.

Agreements — An Alternate Formulation of Our Model: To understand the representational issues involved in learning from a consistently ignorant teacher, we consider as an example the class \mathcal{C} of pure conjunctive concepts (monomials)—each concept is a simple conjunction of boolean variables or their negations. How can we represent a blurry monomial? One way to represent a blurry monomial is to determine which examples are categorically positive and negative (and, consequently, be able to infer that the remaining examples are “?”). Let P, Q , and N be

the sets of examples labeled “+”, “?”, and “-”, respectively, for some blurry monomial. Also, for each q in Q let m_-^q, m_+^q be the (non-blurry) monomials guaranteed to exist by the definition of a blurry concept (i.e., for all q in Q , $m_-^q(q) = \text{“-”} \neq \text{“+”} = m_+^q(q)$ and m_-^q, m_+^q are consistent with P and N). We can represent the examples that are categorically positive with a (non-blurry) monomial. In particular, there exists a monomial that only labels examples in P positive (and, hence, examples in $Q \cup N$ negative). The reason is that now the positive examples are those which are in *each* of m_-^q , for all q in Q . So, P is captured by the intersection of the monomials m_-^q for all q in Q . Since monomials are closed under intersection, there is a monomial that labels only the examples in P positive. Furthermore, we can represent the examples that are categorically negative with a (non-blurry) DNF formula. Namely, there exists a DNF formula that only labels examples in N negative (and, hence, examples in $P \cup Q$ positive). This is because the positive examples are those which are in *any* of m_+^q for all q in Q . In particular, the DNF formula can be represented by the disjunction of m_+^q for all q in Q .

In fact, we can generalize the argument we have given for blurry monomials to any blurry concept class. We use the following definition in proving the lemma.

Definition 3 *Let F be a set of boolean-valued functions. The function Union_F is a boolean-valued function whose classification of example x is given by*

$$\text{Union}_F(x) = \begin{cases} + & \text{if } f(x) = \text{“+”} \text{ for some } f \in F \\ - & \text{otherwise} \end{cases}$$

Likewise, the function Intersect_F is a boolean-valued function whose classification of example x is given by

$$\text{Intersect}_F(x) = \begin{cases} + & \text{if } f(x) = \text{“+”} \text{ for each } f \in F \\ - & \text{otherwise} \end{cases}$$

Lemma 1 *Let \mathcal{C} be a class of boolean-valued concepts. For $f_?$ in $\mathcal{C}_?$, let P, Q, N be the positive, “?”, and negative examples (respectively) of $f_?$. Then there exists $F_-, F_+ \subseteq \mathcal{C}$ such that the positive examples of Intersect_{F_-} are exactly P and the negative examples of Union_{F_+} are exactly N .*

Proof: By the definition of a blurry concept, for each q in Q , there exists f_-^q, f_+^q in \mathcal{C} such that $f_-^q(q) = \text{“-”} \neq \text{“+”} = f_+^q(q)$ and f_-^q, f_+^q are consistent with P and N . Let $F_- = \{f_-^q \mid q \in Q\}$ and $F_+ = \{f_+^q \mid q \in Q\}$. Since for every q in Q , there is an element of F_- that labels q negative (in particular, f_-^q), Intersect_{F_-} labels all q in Q negative. Also, since for all q in Q , f_-^q is consistent with P and N , the only positive examples of Intersect_{F_-} are P . A dual argument can be made for Union_{F_+} . \square

In the proof of Lemma 1, had we considered Intersect_F and Union_F where $F = F_- \cup F_+$, the lemma would still hold. This is demonstrated by the following corollary.

Corollary 2 *Let \mathcal{C} be a class of boolean-valued concepts. For $f_?$ in $\mathcal{C}_?$, let P, Q, N be the positive, “?”, and negative examples (respectively) of $f_?$. Then there exists $F \subseteq \mathcal{C}$ such that the positive examples of Intersect_F are exactly P and the negative examples of Union_F are exactly N .*

Proof: Let $F = \cup_{q \in Q} \{f_-^q, f_+^q\}$, where f_-^q, f_+^q are those functions guaranteed to exist from Definition 1. Observe that adding elements to F can only decrease the number of positive examples of Intersect_F . Since each f_+^q (for q in Q) is consistent with P , the positive examples of Intersect_F are exactly the positive examples of Intersect_{F_-} . (Dually for Union_F .) \square

Observe further that we can use Intersect_F and Union_F to construct a function that predicts the label of an example x as follows. If Intersect_F and Union_F agree on the label of x , then output the label they agree on; otherwise output “?”. This kind of function turns out to be quite useful:

Definition 4 *Let F be a set of boolean-valued functions. The function Agree_F is a ternary function whose classification on example $x \in \mathcal{X}$ is given by*

$$\text{Agree}_F(x) = \begin{cases} + & \text{if } f(x) = “+” \text{ for each } f \in F, \\ - & \text{if } f(x) = “-” \text{ for each } f \in F, \\ ? & \text{otherwise.} \end{cases}$$

Using Corollary 2 and the definition of agreement, we can show that for $f_?$ in $\mathcal{C}_?$, there exists $F \subseteq \mathcal{C}$ such that $f_? = \text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$:

Corollary 3 *Let \mathcal{C} be a class of boolean-valued concepts. Then for any $f_? \in \mathcal{C}_?$, there exists $F \subseteq \mathcal{C}$ such that $f_? = \text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$.*

Proof: By Corollary 2, there exists $F \subseteq \mathcal{C}$ such that the positive examples of Intersect_F are exactly P (and thus the negative examples of Intersect_F are $Q \cup N$). Dually, the negative examples of Union_F are N (and the positive examples of Union_F are $P \cup Q$). Thus, $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels an example x positive iff x is in P . Similarly, $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels an example x negative iff x is in N . Since $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels an example x positive (respectively, negative) iff $f_?$ labels x positive (respectively, negative), $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels x “?” iff $f_?$ labels x “?”. \square

Towards a representation for blurry concepts, we now show that for any blurry concept $f_?$ in $\mathcal{C}_?$, there is a corresponding subset F of \mathcal{C} whose agreement is equivalent to $f_?$. (Thus, the problem of learning agreements of concepts from \mathcal{C} is equivalent to learning \mathcal{C} from a consistently ignorant teacher, or equivalently, learning the blurry class $\mathcal{C}_?$.)

Lemma 4 *For a class \mathcal{C} of boolean-valued concepts, the blurry class $\mathcal{C}_? = \{\text{Agree}_F \mid F \subseteq \mathcal{C}\}$.*

Proof: (\subseteq) Let $f_? \in \mathcal{C}_?$. By Corollary 3 there exists $F \subseteq \mathcal{C}$ such that $f_? = \text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$. Observe that $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}} = \text{Agree}_F$ since for any example x , $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels x positive if and only if Intersect_F labels x positive (since Intersect_F is more specific than $\text{Union}_F(x)$). Thus, the function $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels x positive iff every f in F labels x positive. But, by definition, this is when Agree_F labels x positive. An analogous argument shows that the two functions are identical when x is a negative example. Finally, since Agree_F labels x positive (respectively, negative) iff $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels x positive (respectively, negative), Agree_F labels x “?” iff $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ labels x “?”.

(\supseteq) Let $F \subseteq \mathcal{C}$. We show that Agree_F is a blurry concept for \mathcal{C} . For every “?” example q of Agree_F , there must exist f, f' in F such that $f(q) \neq f'(q)$ (otherwise q would not be a “?” example) and both f and f' are consistent with the positive and negative examples of Agree_F (by Definition 4). Thus, for all $F \subseteq \mathcal{C}$, there exists $f_? \in \mathcal{C}_?$ such that $\text{Agree}_F = f_?$. \square

Measuring the Size of a Blurry Concept: Now that we have a way to represent blurry concepts, we can describe our complexity measure for the size of $f_?$. Recall that for a concept class \mathcal{C} , typically the (representation) size of a concept $f \in \mathcal{C}$ is taken to be the fewest number of symbols needed to write f as a member of \mathcal{C} . This definition extends naturally to give the representation size of a finite set of concepts F — define the size of F to be $\sum_{f \in F} |f|$, where the size $|f|$ is given by the size measure for the base class \mathcal{C} . Now define the size $|f_?|$ for $f_? \in \mathcal{C}_?$ to be the minimum, over all $F \subseteq \mathcal{C}$ for which $\text{Agree}_F = f_?$, of the representation size of F . Analogously, the sizes of Intersect_F and Union_F are each just the representation size of the subset F . Note that $|\text{Intersect}_F| = |\text{Union}_F| = |\text{Agree}_F|$. We only consider learning functions of the form Agree_F , where F is finite. If no such finite F exists, then the representation size of $f_?$ is infinite and, consequently, the learning problem is ill-defined.

The notion of an agreement of base concepts has independent interest, as it models a type of unanimous vote of independent agents.

Finally, there is an interesting relationship between Mitchell’s definition of version spaces [Mit82] and agreements. Given a concept class \mathcal{C} of boolean-valued functions and a set of examples M , the *version space* V is the set of concepts in \mathcal{C} consistent with M . As the version space can be large, it is often represented by the sets G and S where S is the subset of V that contains the most specific³ concepts and G is the subset of V that contains the most general concepts.⁴ The G and S sets

³We say that $f_1 \in \mathcal{C}$ is more specific (respectively, more general) than $f_2 \in \mathcal{C}$ if $f_1 \subset f_2$ (respectively, $f_1 \supset f_2$).

⁴In particular, $S = \{s \in V : \text{there is no } s' \text{ in } V \text{ that is strictly more specific than } s\}$ and $G = \{g \in V : \text{there is no } g' \text{ in } V \text{ that is strictly more general than } g\}$. Note also that not every version space can be represented by the boundary sets S and G [GNPS91, Hir91, Mit78].

induce a ternary function, which we call $\text{VS}_{[S,G]}$ defined as follows:

$$\text{VS}_{[S,G]}(x) = \begin{cases} + & \text{if } s(x) = "+" \text{ for all } s \in S, \\ - & \text{if } g(x) = "-" \text{ for all } g \in G, \\ ? & \text{otherwise.} \end{cases}$$

For a concept class \mathcal{C} , it can be shown that the agreement of a finite subset F of \mathcal{C} is equivalent to $\text{VS}_{[S,G]}$, for suitably defined S and G . Also, given sets S and G , it can be shown that the ternary function induced by S and G can be represented as an agreement. In particular, $\text{VS}_{[S,G]}(x) = \text{Agree}_{S \cup G}(x)$ for all x . However, there are cases where exponentially smaller subsets S' of S and G' of G are sufficient for prediction, i.e., for all x , $\text{VS}_{[S,G]}(x) = \text{Agree}_{S' \cup G'}(x)$ for $S' \subset S$ and $G' \subset G$ and $|G| + |S| = \exp(|G'| + |S'|)$. Since our goal is only to predict well, maintaining the set of most general and most specific consistent concepts may create much unneeded, time-consuming work.

4 Positive Results for Learning Agreements

We show that efficient PAC and PAC-memb learning algorithms can be designed to learn from consistently ignorant teachers. We first consider the problem of learning the agreement of a pair of nested concepts. We show that if both concepts are chosen from classes for which efficient learning algorithms exist, then we can use these algorithms to obtain an efficient algorithm for learning the agreement of the functions. We then present a general result addressing how known algorithms for learning from omniscient teachers can be applied to learn from consistently ignorant teachers even when the base functions are not nested. In particular, we show that when unions and intersections of concepts from \mathcal{C} are learnable, the blurry class $\mathcal{C}_?$ is learnable (equivalently, \mathcal{C} is learnable from a consistently ignorant teacher). These techniques are applied to show that Horn clauses, 1-DNF (1-CNF) formulas containing $O(\log n)$ literals, CLASSIC sentences, and monomials (1-DNF formulas) with at least one positive (respectively, negative) example, are learnable from a consistently ignorant teacher. Simple extensions show that those blurry concepts representable as an agreement of a constant number of monotone DNF (CNF) formulas, k -term DNF (k -clause CNF) formulas, decision trees, and DFAs, are learnable.

4.1 Learning Agreements of Nested Concepts

Recall that a concept $f \in \mathcal{C}$ is simply the subset of instances from \mathcal{X} that f classifies as positive. Thus for two concepts f_1 and f_2 , we write $f_1 \subseteq f_2$ if the set of positive examples of f_1 is a subset of the positive examples of f_2 . Given a set of concepts $F = \{f_1, \dots, f_k\}$ we say that these concepts are *nested* if $f_1 \subseteq f_2 \subseteq \dots \subseteq f_k$. Observe that $\text{Agree}_{\{f_1, \dots, f_k\}} = \text{Agree}_{\{f_1, f_k\}}$ and thus, without loss of generality, we consider learning the agreement, $\text{Agree}_{\{f_s, f_g\}}$, of two nested functions f_s and

f_g (s and g for “specific” and “general”). Suppose these are chosen, respectively, from known polynomial-time learnable concept classes \mathcal{C}_S and \mathcal{C}_G . Then the learning algorithms for \mathcal{C}_S and \mathcal{C}_G can be used to learn the following class :

$$\text{Nested}_?(\mathcal{C}_S, \mathcal{C}_G) = \{\text{Agree}_{\{f_s, f_g\}} \mid f_s \in \mathcal{C}_S, f_g \in \mathcal{C}_G, \text{ and } f_s \subseteq f_g\}.$$

(See Figure 1 for the algorithm.)

Theorem 5 *If \mathcal{C}_S and \mathcal{C}_G are polynomially PAC-memb (respectively PAC) learnable concept classes, then the class $\text{Nested}_?(\mathcal{C}_S, \mathcal{C}_G)$ is polynomially PAC-memb (respectively PAC) learnable.*

Proof: If the target is $\text{Agree}_{\{f_s, f_g\}}$ for f_s in \mathcal{C}_S and f_g in \mathcal{C}_G , note that a positive (respectively, negative) example of $\text{Agree}_{\{f_s, f_g\}}$ is classified as positive (respectively, negative) by both f_s and f_g and a “?” example is classified as negative by f_s and positive by f_g . Thus, algorithm Learn-Agreement-Nested-Concepts in Figure 1 learns $\text{Agree}_{\{f_s, f_g\}}$ by running the learning algorithm for \mathcal{C}_S treating “?” as “−” to obtain h_S , and running the algorithm for \mathcal{C}_G treating “?” as “+” to obtain h_G , and outputs $h = \text{Agree}_{\{h_S, h_G\}}$ as the final hypothesis.

Since h_S and h_G both have error at most $\epsilon/2$ with probability at least $1 - \delta/2$, it follows that h has error at most ϵ with probability at least $1 - \delta$. Finally, since \mathcal{A}_S and \mathcal{A}_G run in polynomial time, Learn-Agreement-Nested-Concepts runs in polynomial time. Also note that Learn-Agreement-Nested-Concepts only makes a membership query when either \mathcal{A}_S or \mathcal{A}_G does. \square

In the special case where $\mathcal{C}_S = \mathcal{C}_G = \mathcal{C}$ is learnable, Theorem 5 shows that nested concepts from \mathcal{C} are learnable.

4.2 A General Technique for Learning Agreements

We use the characterization in Lemma 4 to obtain an efficient algorithm for learning a blurry concept class $\mathcal{C}_?$ (equivalently, learning a class \mathcal{C} from a consistently ignorant teacher) when intersections and unions from \mathcal{C} are known to be learnable. To aid the exposition, we define the following sets:

$$\mathcal{C}_\cap = \{\text{Intersect}_F : F \subseteq \mathcal{C}\}, \quad \mathcal{C}_\cup = \{\text{Union}_F : F \subseteq \mathcal{C}\}.$$

Theorem 6 *Let \mathcal{C} be a boolean-valued concept class for which \mathcal{C}_\cap and \mathcal{C}_\cup are PAC-memb (respectively PAC) learnable in polynomial time. Then $\mathcal{C}_?$ is PAC-memb (respectively PAC) learnable in polynomial time.*

Proof: Let $f_?$ be any element of $\mathcal{C}_?$, represented by Agree_F for some finite $F \subseteq \mathcal{C}$ (see Lemma 4). By Corollary 3, $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}} = \text{Agree}_F (= f_?)$. Since $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ and Agree_F represent the same ternary functions, an algorithm that learns the agreement of Intersect_F

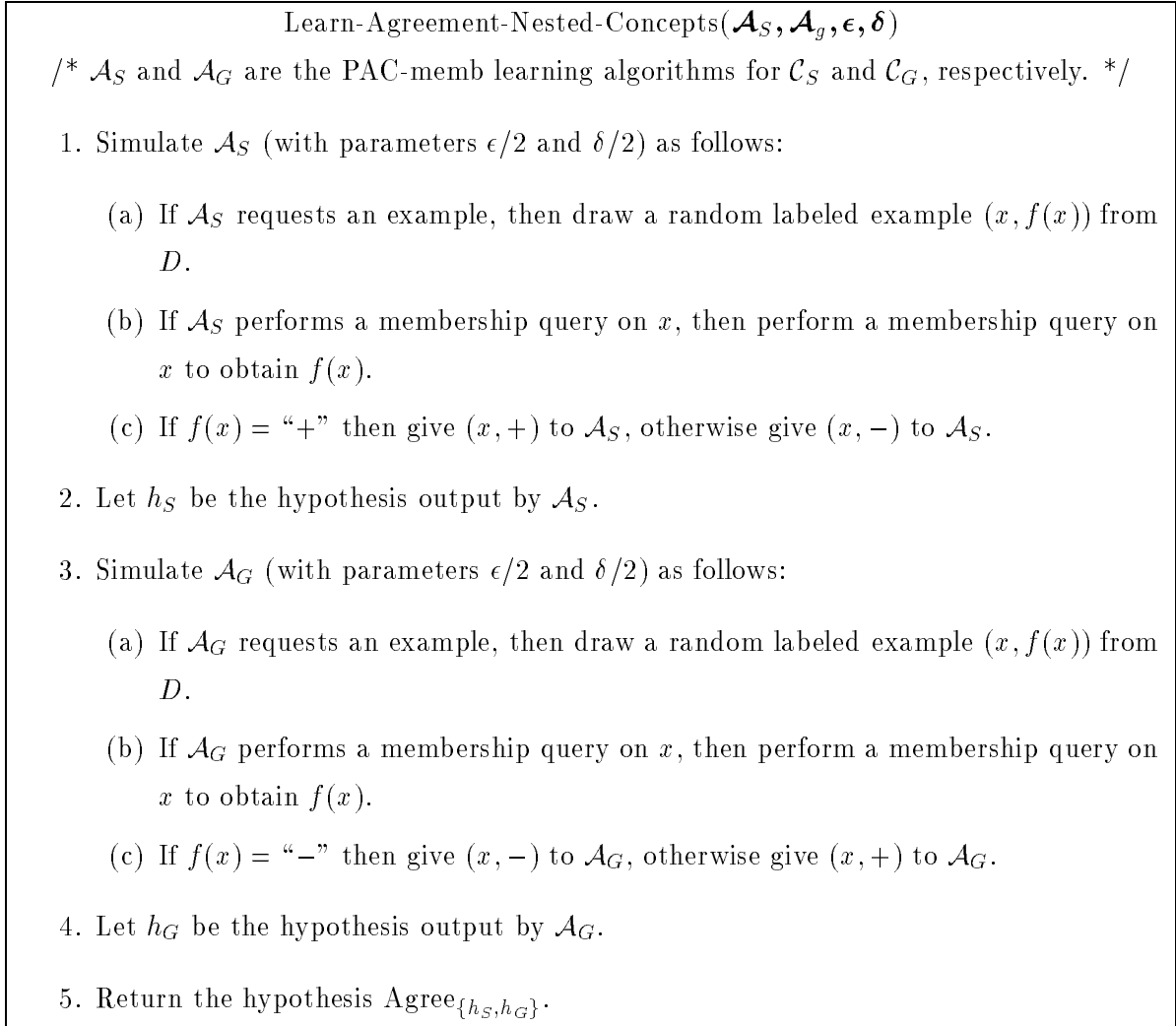


Figure 1: A method for learning the agreement of nested concepts

and Union_F can be used to learn Agree_F . It is important to note that $|\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}| \leq 2 \cdot |\text{Agree}_F|$, since $|\text{Intersect}_F| = |\text{Union}_F| = |\text{Agree}_F|$. Thus, the running time of the learning algorithm when learning $f? = \text{Agree}_F = \text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}$ is polynomial in $|\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}}|$, which is polynomial in $|\text{Agree}_F|$. To see that there is an algorithm to learn any such Agree_F , note that $\text{Intersect}_F \subseteq \text{Union}_F$, and thus $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}} \in \text{Nested?}(\mathcal{C}_\cap, \mathcal{C}_\cup)$. By assumption, \mathcal{C}_\cap and \mathcal{C}_\cup are efficiently learnable and so, by Theorem 5, $\text{Nested?}(\mathcal{C}_\cap, \mathcal{C}_\cup)$ is efficiently learnable. Thus, $\text{Agree}_{\{\text{Intersect}_F, \text{Union}_F\}} = f?$ can be learned. \square

As discussed in Section 6, Theorems 5 and 6 can be strengthened to hold in a suitably modified exact learning model (with membership queries).

We give some direct implications of Theorem 6.

Corollary 7 *For $\mathcal{C} \in \{\text{Horn clauses, 1-DNFs containing at most } O(\log n) \text{ literals, CLASSIC sentences}\}$, $\mathcal{C}_?$ is polynomially PAC-memb learnable.*

Proof: For \mathcal{C} the class of Horn clauses, \mathcal{C}_\cap is the class of Horn sentences which is known to be PAC-memb learnable [AFP92]. For \mathcal{C} the class of 1-DNF formulas containing at most $O(\log n)$ literals, \mathcal{C}_\cap is the class of $O(\log n)$ -CNF expressions, which is known to be PAC-memb learnable [Bsh95]. In both cases, \mathcal{C}_\cup is the class of 1-DNF expressions, which is known to be PAC learnable [Val84]. Hence, for \mathcal{C} the class of Horn clauses and 1-DNF formulas with at most $O(\log n)$ literals, by Theorem 6, $\mathcal{C}_?$ is PAC-memb learnable.

The class of CLASSIC sentences is known to be polynomially PAC-memb learnable [FP94]. Further, since the syntax of CLASSIC admits an “AND” construct, the intersection of any two CLASSIC sentences is itself a CLASSIC sentence of size that is the sum of the sizes of the sentences being intersected. It follows that intersections of CLASSIC sentences are polynomially PAC-memb learnable. There are different possible semantics for the “union” of CLASSIC sentences; in a recent extension of [FP94], it is shown that a “weak union” of CLASSIC sentences are PAC-memb learnable, and that this is sufficient to show that agreements of CLASSIC sentences are learnable, hence CLASSIC is learnable from a consistently ignorant teacher.⁵ \square

The corollary above also applies to the corresponding dual class, i.e., when \mathcal{C} is the class of 1-CNF formulas containing at most $O(\log n)$ literals.

For a moment we return to the example of learning the agreement of monomials. We argue that unless we make some restrictions on the target concept, the problem of learning the agreement of monomials is as hard as learning (boolean-valued) DNF formulas. Consider the task of learning the DNF formula $f = t_1 \vee \dots \vee t_k$. We can reduce the problem of learning f to the problem of learning the blurry concept $f? = \text{Agree}_{\{t_1, t_2, \dots, t_k, \text{false}\}}$. Note that $f?$ evaluates to “?” on x iff $f(x) = 1$, and

⁵The result that a weak union of CLASSIC sentences is learnable, and that CLASSIC is learnable from a consistently ignorant teacher, appear as Theorem 17 and Corollary 18, respectively, in the full version of [FP94].

evaluates to 0 on x iff $f(x) = 0$. Thus, any algorithm that learns the agreement of monomials can be used to learn (boolean-valued) DNF by simply interpreting all positive examples of the DNF algorithm as “?” examples of the agreement of monomials algorithm. Hence, we have the following:

Observation 8 *Learning blurry monomials is as hard as learning (boolean-valued) DNF formulas.*

In the above example, since the false function is included in the agreement, the set of positive examples becomes empty, precluding the possibility of using information about positive examples to aid in learning to distinguish between negative and “?” examples. However, if we require that there be at least one positive example to a blurry monomial, then Corollary 10 demonstrates that such concepts are efficiently learnable. Not surprisingly, the requirement that there is at least one positive example appears necessary to obtain positive results for other classes. Thus, we introduce the following definitions.

$$\mathcal{C}_\cap^+ = \{\text{Intersect}_{F^+} : F^+ \subseteq \mathcal{C}, \exists x \in \mathcal{X}, f(x) = “+” \text{ for all } f \in F^+\}$$

$$\mathcal{C}_\cup^+ = \{\text{Union}_{F^+} : F^+ \subseteq \mathcal{C}, \exists x \in \mathcal{X}, f(x) = “+” \text{ for all } f \in F^+\}$$

$$\mathcal{C}_?^+ = \{f? : f? \text{ has at least one positive example}\}$$

A simple modification of the proof of Lemma 4 shows that $\mathcal{C}_?^+ = \{\text{Agree}_{F^+} : F^+ \subseteq \mathcal{C}\}$, immediately yielding the following analog of Theorem 6.

Corollary 9 *Let \mathcal{C} be a boolean-valued concept class for which \mathcal{C}_\cap^+ and \mathcal{C}_\cup^+ are PAC-memb (respectively PAC) learnable in polynomial time. Then $\mathcal{C}_?^+$ is PAC-memb (respectively PAC) learnable in polynomial time.*

As a corollary to the above observations, we show that the class of blurry monomials with at least one positive example is learnable.

Corollary 10 *For \mathcal{C} the class of monomials, $\mathcal{C}_?^+$ is polynomially PAC-memb learnable.*

Proof: We show \mathcal{C}_\cap^+ and \mathcal{C}_\cup^+ are learnable when \mathcal{C} is the class of monomials, and the result follows by Corollary 9. The class \mathcal{C}_\cap^+ is learnable since the intersection of an arbitrary number of monomials can be represented as a monomial of length at most n (where n is the number of variables) and the class of monomials is known to be PAC learnable [Val84]. We show \mathcal{C}_\cup^+ is the class of unate DNF formulas and hence PAC-memb learnable by [AHK93]. To see that \mathcal{C}_\cup^+ is the class of unate DNF formulas, note that for $F^+ \subseteq \mathcal{C}$, since there is an example x that Intersect_{F^+} labels positive (by the definition of \mathcal{C}_\cup^+), x satisfies every monomial in F^+ . It follows that no variable can appear both negated and unnegated in F^+ . \square

Had we not restricted the class \mathcal{C} in the above corollary to only those blurry monomials with at least one positive example, the proof would fail — Union_F would not necessarily be unate, and could be instead an arbitrary DNF formula. The learnability of general DNF formulas in the PAC-memb model remains a challenging open question.⁶ Corollary 10 also applies to the dual of monomials (i.e., 1-DNF) by considering the analogous class $\mathcal{C}_?^-$.

Unfortunately, a priori knowledge that a blurry concept has at least one positive example does not always turn out to be useful. In particular, for each of the problems of learning blurry monotone DNF formulas, blurry decision trees, and blurry DFAs, the existence of at least one positive example does not make the problem any easier. In order to demonstrate this, we need the notion of a *prediction preserving reduction with membership queries* [AK95] (see also [PW90]). While the definition of a prediction-preserving reduction is somewhat involved, it in essence captures the idea that one can sometimes use an efficient learning algorithm for one concept class to construct an efficient learning algorithm for a different concept class. For a class \mathcal{C} let $\mathcal{L}(\mathcal{C})$ denote the learning problem for \mathcal{C} . For concept classes \mathcal{C}_1 and \mathcal{C}_2 , we use the notation $\mathcal{L}(\mathcal{C}_1) \trianglelefteq \mathcal{L}(\mathcal{C}_2)$ to mean that PAC-memb learning \mathcal{C}_1 reduces to PAC-memb learning \mathcal{C}_2 , in the sense of [AK95]. We will present such reductions informally — namely by showing how an efficient algorithm for PAC-memb learning \mathcal{C}_2 can be used to efficiently PAC-memb learn \mathcal{C}_1 .

Theorem 11 *For $\mathcal{C} \in \{\text{monotone DNF formulas, decision trees, DFAs}\}$, $\mathcal{L}(\mathcal{C}_?) \trianglelefteq \mathcal{L}(\mathcal{C}_?^+)$.*

Proof: For \mathcal{C} the class of monotone DNF formulas, we show that $\mathcal{L}(\mathcal{C}_?) \trianglelefteq \mathcal{L}(\mathcal{C}_?^+)$ by constructing an algorithm \mathcal{A} to learn $\mathcal{C}_?$ given that there exists an algorithm \mathcal{A}^+ to learn $\mathcal{C}_?^+$. If $f_?$ labels the example $\langle 1 \cdots 1 \rangle$ positive (which \mathcal{A} can check using a membership query) then \mathcal{A} runs \mathcal{A}^+ and outputs whatever it does. In this case, the output of \mathcal{A}^+ is correct because $f_?$ is in $\mathcal{C}_?^+$. Suppose that $f_?$ labels $\langle 1 \cdots 1 \rangle$ negative. Then the formula False must be in the agreement as it is the only monotone DNF formula that labels the example $\langle 1 \cdots 1 \rangle$ negative. Hence, there is only one boundary to learn — the “lower boundary” or the one separating “?” examples from “−” examples. Suppose that $f_?$ is represented as an agreement of the set of concepts $F \subseteq \mathcal{C}$. Observe that the “lower” boundary is expressed by the function Union_F , which is representable as a monotone DNF formula of size $\sum_{f \in F} |f|$. Thus, \mathcal{A} runs the monotone DNF learning algorithm [Ang88] treating all “?” examples as “+” examples, and obtains some hypothesis h that (with probability at least $1 - \delta$) correctly (within ϵ) classifies examples as does Union_F . To classify an example x , \mathcal{A} outputs “?” if $h(x) = “+”$, otherwise \mathcal{A} outputs “−”. The time taken by \mathcal{A} is polynomial in $|f_?|$, since the time taken by the monotone DNF learning algorithm is polynomial in $\sum_{f \in F} |f|$.

Next we show that for \mathcal{C} the class of decision trees, $\mathcal{L}(\mathcal{C}_?) \trianglelefteq \mathcal{L}(\mathcal{C}_?^+)$. In particular, we construct an algorithm \mathcal{A} to learn $\mathcal{C}_?$ given an algorithm \mathcal{A}^+ that learns $\mathcal{C}_?^+$. Suppose that $f_?$ is represented

⁶However, DNF formulas are PAC-memb learnable with respect to the uniform distribution [Jac94].

as an agreement of concepts f_1, \dots, f_t in \mathcal{C} over variables v_1, \dots, v_n . In order to learn $f_?$, \mathcal{A}^+ will be run on examples of the function $f_?^+$, the agreement of f'_1, \dots, f'_t , where $f'_i = v_0 \vee f_i$ (for v_0 a new variable). Such examples are easily constructed from examples of $f_?$ without knowledge of f_1, \dots, f_t , because for all i , f'_i labels as positive any $(n+1)$ -bit example $1 \cdot x$ (i.e., the example with 1 in the position v_0 and x in positions v_1, \dots, v_n), and labels the example $0 \cdot x$ the way that f_i labels x . With this observation, the algorithm \mathcal{A} to learn $\mathcal{C}_?$ can use the algorithm \mathcal{A}^+ for $\mathcal{C}_?^+$ (with the unknown target $f_?^+ = \text{Agree}_{\{v_0 \vee f_1, \dots, v_0 \vee f_t\}}$) in the following way. If \mathcal{A}^+ requests a random example of $f_?^+$, then \mathcal{A} draws a random example y of $f_?$ defined over the original variables v_1, \dots, v_n , and returns the example $0 \cdot y$. If \mathcal{A}^+ poses a membership query on example y of $f_?^+$, then \mathcal{A} returns the label “+” if the first bit position v_0 of y is 1, because such an example satisfies each $v_0 \vee f_i$. If the first bit position v_0 of y is 0, then \mathcal{A} poses a membership query (to the oracle for $f_?$) on the example y with bit position v_0 deleted and returns to \mathcal{A}^+ the label that this membership query returns. This transformation works because the formula $f_?^+$ with v_0 set to 0 is equivalent to $f_?$. Once \mathcal{A}^+ terminates, \mathcal{A} can use the hypothesis h output by \mathcal{A}^+ to determine the label of an example y by evaluating $h(0 \cdot y)$. Observe that the time taken by \mathcal{A} in learning $f_?$ is bounded by a polynomial in the time taken by \mathcal{A}^+ in learning $f_?^+$. But this is a polynomial in $|f_?|$, since \mathcal{A}^+ runs in polynomial time, and the size of the decision tree that represents f'_i is $O(|f_i|)$ — the root node of the new decision tree can now be v_0 with the right 1 branch terminating in a True leaf, and with the left 0 branch leading to the original decision tree f_i .

For \mathcal{C} the class of DFAs, we again show that $\mathcal{L}(\mathcal{C}_?) \preceq \mathcal{L}(\mathcal{C}_?^+)$. Suppose that $f_?$ is represented as an agreement of DFAs M_1, \dots, M_t in \mathcal{C} . As before, we show the existence of M'_1, \dots, M'_t that each label at least one example positive, and $|M'_i| = O(|M_i|)$. To achieve this goal, we associate a modified DFA, M'_i , for each DFA M_i that has a new start state with a transition to an accept state if the first bit of the input string is 1, and a transition to the old DFA M_i if the first bit of the input string is 0. Note that if r is the regular expression that corresponds to the DFA M_i , then $1^* + 0 \cdot r$ is the regular expression that captures M'_i . The argument proceeds analogously to the decision tree argument above constructing, an algorithm \mathcal{A} to learn $\mathcal{C}_?$ by running the algorithm \mathcal{A}^+ given for $\mathcal{C}_?^+$ (with the target $f_?^+ = \text{Agree}_{\{M'_1, \dots, M'_t\}}$). \square

So, while the restriction that a blurry concept class has one positive example was useful in the case of monomials (and, as we will see, is also useful when learning the agreement of boxes), this theorem demonstrates that it does not always alter the complexity of learning the arbitrary blurry concept class. In fact, in Section 5, we show that learning $\mathcal{C}_?$ for the classes considered in the above theorem is an apparently hard problem. Since having one positive example does not always affect the difficulty of a learning problem, we consider learning a different subset of blurry concept classes — those that can be represented as an agreement of a constant number of concepts. With this restriction, the blurry classes $\mathcal{C}_?$ we showed to be as hard as learning $\mathcal{C}_?^+$ are in fact learnable. We

use the following notation:

$$\mathcal{C}_\cap^k = \{\text{Intersect}_{F^k} : F^k \subseteq \mathcal{C}, \text{cardinality of } F^k \text{ is } k\},$$

$$\mathcal{C}_\cup^k = \{\text{Union}_{F^k} : F^k \subseteq \mathcal{C}, \text{cardinality of } F^k \text{ is } k\},$$

$$\mathcal{C}_?^k = \{\text{Agree}_{F^k} : F^k \subseteq \mathcal{C}, \text{cardinality of } F^k \text{ is } k\}.$$

For $F^k \subseteq \mathcal{C}$, $\text{Agree}_{F^k} = \text{Agree}_{\{\text{Intersect}_{F^k}, \text{Union}_{F^k}\}}$, by a simple instantiation of Lemma 4. Furthermore, $|\text{Agree}_{\{\text{Intersect}_{F^k}, \text{Union}_{F^k}\}}| \leq 2 \cdot |\text{Agree}_{F^k}|$. By (yet another) simple modification of Theorem 6, the learnability of $\mathcal{C}_?^k$ follows from the learnability of intersections and unions of a constant k number of concepts from \mathcal{C} .

Corollary 12 *Let \mathcal{C} be a boolean-valued concept class for which \mathcal{C}_\cap^k and \mathcal{C}_\cup^k are PAC-memb (respectively PAC) learnable in polynomial time. Then $\mathcal{C}_?^k$ is PAC-memb (respectively PAC) learnable in polynomial time.*

Corollary 12 is applied to show that the agreement of a constant number of monotone DNF formulas, ℓ -term DNF formulas (for ℓ constant), decision trees, and DFAs, is learnable.

Corollary 13 *For $\mathcal{C} \in \{\text{monotone DNF formulas}, \ell\text{-term DNF formulas}, \text{decision trees}, \text{DFAs}\}$, $\mathcal{C}_?^k$ is learnable, for each constant k .*

Proof: The proof for each \mathcal{C} in the corollary has the following structure. We show first that for any $F^k \subseteq \mathcal{C}$ of cardinality k constant, Intersect_{F^k} can be represented as an element of \mathcal{C} with size at most polynomial in $|\text{Agree}_{F^k}| = \sum_{f \in F^k} |f|$ (and similarly for Union_{F^k}). Noting that since \mathcal{C} is PAC-memb learnable in time polynomial in the size of its target, and since polynomials are closed under composition, \mathcal{C} is learnable in time polynomial in $|\text{Agree}_{F^k}|$. Hence \mathcal{C}_\cap^k and \mathcal{C}_\cup^k are efficiently learnable and, by Corollary 12, $\mathcal{C}_?^k$ is efficiently learnable. We fill in some of the details for each \mathcal{C} mentioned in the corollary.

For \mathcal{C} the class of monotone DNF formulas, let $F^k \subseteq \mathcal{C}$ with the cardinality of F^k equal to k . A monotone DNF representation for Intersect_{F^k} can be obtained by “multiplying out” the monotone DNF formulas in F^k . The size of Intersect_{F^k} is at most $\prod_{f \in F^k} |f| \leq (\max_{f \in F^k} |f|)^k$. So, for k constant, the size of Intersect_{F^k} is polynomial in $|\text{Agree}_{F^k}|$. A monotone DNF representation for Union_{F^k} can be obtained by disjoining the monotone DNF formulas in F^k . The size of this representation is at most $\sum_{f \in F^k} |f|$. Thus, since each function in \mathcal{C}_\cap^k and \mathcal{C}_\cup^k is efficiently representable as a monotone DNF, and the class \mathcal{C} of monotone DNF formulas is PAC-memb learnable [Val84], \mathcal{C}_\cap^k and \mathcal{C}_\cup^k are PAC-memb learnable.

For \mathcal{C} the class of ℓ -term DNF formulas, ℓ constant, let $F^k \subseteq \mathcal{C}$ be such that the cardinality of F^k is k . Intersect_{F^k} can be represented as an ℓ^k -term DNF formula (again, by “multiplying out”

the ℓ -term DNF formulas in F^k) and Union_{F^k} can be represented as an (ℓk) -term DNF formula. Since both ℓ and k are constant, each function in \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k is efficiently representable as an ℓ' -term DNF formula, for ℓ' constant. Moreover, since for each constant ℓ' , ℓ' -term DNF formulas are PAC-memb learnable [Ang87b, BR92], \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k are efficiently learnable.

For \mathcal{C} the class of decision trees, consider as an example the case when $k = 2$. We wish to show that for decision trees d_1 and d_2 , there exists a small representation for $d_1 \cap d_2$ and $d_1 \cup d_2$ as decision trees. To obtain a representation for $d_1 \cap d_2$ (respectively, $d_1 \cup d_2$), replace all “+” leaves (respectively, “−” leaves) in d_1 with the decision tree d_2 . Now observe that an example x is labeled positive by this decision tree iff x is labeled positive by both (respectively, at least one of) d_1 and d_2 . The size of such a decision tree is at most the number of leaves in d_1 multiplied by the size of d_2 . Analogously, for k constant, for $F^k \subseteq \mathcal{C}$ with the cardinality of F^k equal to k , $|\text{Intersect}_{F^k}| \leq (\max_{f \in F^k} |f|)^k$, and similarly for $|\text{Union}_{F^k}|$. Since each function in \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k is efficiently representable as a decision tree, and the class of decision trees is efficiently PAC-memb learnable [Bsh95], \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k are also efficiently learnable.

Finally, for \mathcal{C} the class of DFAs, standard arguments [HU79] show that the intersection and union of a constant k number of DFAs is representable as a DFA of size the product of the sizes of the DFAs in the intersection or union (but exponential in k). Since each function in \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k is efficiently representable as a DFA, and DFAs are efficiently PAC-memb learnable [Ang87b], \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k are also efficiently learnable. \square

The corollary above also applies to the corresponding dual classes, i.e., when \mathcal{C} is the class of monotone CNF formulas and ℓ -clause CNF formulas, \mathcal{C}_{\cap}^k is efficiently PAC-memb learnable.

4.3 Learning Agreements of Boxes in Euclidean Space

In this section we show that axis-parallel boxes (henceforth referred to as boxes) can be learned from a consistently ignorant teacher. We first apply Corollary 12 to show that the agreement of a constant number of boxes is learnable. Next we give a PAC-memb algorithm to learn the agreement of s boxes in d dimensional Euclidean space (E^d) when the set of boxes has a *samplable*⁷ intersection. It is easy to show that this class is a generalization of unate DNF formulas, and a specialization of the class of unions of boxes in E^d .

Corollary 14 *For \mathcal{C} the class of axis-parallel boxes in E^d , \mathcal{C}_{\cap}^k is PAC learnable*

Proof: For any $F^k \subseteq \mathcal{C}$ of cardinality k constant, Intersect_{F^k} can be represented as an element of \mathcal{C} since the intersection of axis-parallel boxes is a (possibly empty) single axis-parallel box. Hence, $|\text{Intersect}_{F^k}| = O(d)$ and is efficiently learnable by results of [BEHW89]. Further, Union_{F^k} can be

⁷A set of examples is *samplable* if the probability p^+ of drawing an example from that set is $\frac{1}{q(s,d)}$ where q is some polynomial.

represented as the union of a constant number of boxes, where $|\text{Union}_{F^k}| = \sum_{f \in F^k} |f|$. Observe that by results of [LW94], \mathcal{C}_{\cup}^k is PAC learnable in time polynomial in d , $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$. Since $|\text{Intersect}_{F^k}| \leq |\text{Union}_{F^k}| \leq |\text{Agree}_{F^k}|$, we have that both \mathcal{C}_{\cap}^k and \mathcal{C}_{\cup}^k are learnable in time polynomial in $|\text{Agree}_{F^k}|$, and, hence, by Corollary 12, $\mathcal{C}_{?}^k$ is efficiently PAC learnable. \square

Now we present an algorithm to PAC-memb learn the *agreement* of s boxes in E^d that runs in time polynomial in $1/\epsilon$, $1/\delta$, s , and 2^d . So, the algorithm runs in polynomial time without demanding that one of s and d be constant (d can be $\Theta(\log s)$). Before describing the details of our algorithm, we first provide a high-level overview. To aid in learning the agreement of boxes, we first learn the intersection region (which is itself a box). We can approximate the intersection box by treating all “?” examples in the sample as negative examples and running a known algorithm to learn one d -dimensional box [BEHW89]. Learning the intersection box allows us to distinguish between positive and non-positive examples. To successfully learn the ternary function, however, we must be able to distinguish between “?” and negative examples as well (and since neither the “?” or negative region is a box, we must take a different approach than the one used to learn the positive region). To accomplish this task, we choose a random positive example p . Now we have 2^d versions of the same problem. In particular, we have a union of a set of boxes that all lie in the same quadrant of E^d (where the origin is now p) and which all contain the origin as a corner point. We call such a box an *origin-incident* box. Treating all “?” examples as positive, we give a PAC-memb algorithm to learn the union of s origin-incident boxes within a single quadrant that runs in time polynomial in both s and d . Each quadrant can be learned by the algorithm for learning the union of origin-incident boxes (where the origin is now p). In the worst case, some piece of each of the s boxes will lie in each of the 2^d quadrants of the sub-divided problem forcing us to learn $O(s2^d)$ boxes. At this stage, we have a hypothesis that predicts properly. Specifically, if a point x lies in the intersection box, it is labeled positive, otherwise, if x lies in the union of the $O(s2^d)$ boxes learned for each quadrant, x is labeled “?”, and otherwise it is labeled negative. While such a hypothesis is sufficient for prediction, it does not form an agreement. We can obtain a hypothesis of the appropriate form by outputting the agreement of the intersection box and the boxes in the union extended to include the intersection box.

4.3.1 Approximately Learning the Union of Origin-incident Boxes

We present a PAC-memb algorithm to learn the union of s origin-incident (nonblurry) boxes in E^d where all of the boxes are in the same quadrant (for simplicity we only present the algorithm, Figure 2, for the positive quadrant). We refer to the class of origin-incident boxes in the positive quadrant as BPQ.

We denote the origin in BPQ by the zero vector $\vec{0}$. In general, we represent a box by any two opposing corners x and y using the ordered pair notation (x, y) . An origin-incident box is

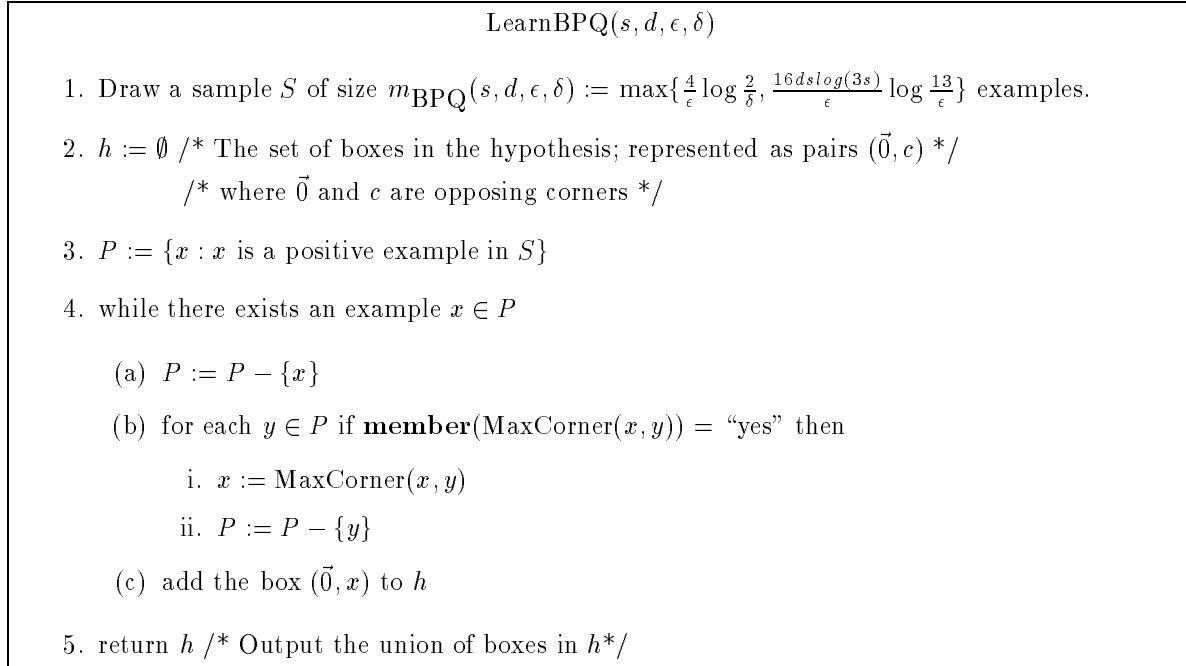


Figure 2: Algorithm to learn a union of origin-incident boxes.

represented by the ordered pair (o, c) where o is the origin and c is the corner opposing the origin. Finally, we define *MaxCorner* to be a function that takes two points (x, y) in the positive quadrant of E^d and returns the point z which, for $1 \leq i \leq d$, has i th coordinate $z_i = \max\{x_i, y_i\}$.

Theorem 15 *Let $\text{BPQ}_{\cup}(s)$ be the union of at most s origin incident boxes in the positive quadrant. The class $\text{BPQ}_{\cup}(s)$ is PAC-memb learnable with time and sample complexity polynomial in $s, d, 1/\epsilon, 1/\delta$.*

Proof: To prove the theorem, we show that

1. Algorithm LearnBPQ (Figure 2), constructs a sample S , runs in time polynomial in $|S|$, and outputs a union of at most s origin-incident boxes (that is, an element of $\text{BPQ}_{\cup}(s)$) that is consistent with the sample.
2. The VC-dimension⁸ of $\text{BPQ}_{\cup}(s)$ grows polynomially with s and d (namely, it is at most $2ds \log 3s$).

It then follows from Theorem 2.1 of Blumer et al. [BEHW89] that if LearnBPQ chooses at least $m_{\text{BPQ}} = \max\{\frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{16ds \log(3s)}{\epsilon} \log \frac{13}{\epsilon}\}$ random examples, then with probability at least $1 - \delta$, it will output a hypothesis h with error at most ϵ .

⁸The VC-dimension is a combinatorial parameter of a concept class that directly relates to the number of examples necessary (and sufficient) for sufficient generalization [BEHW89].

Proof of Part 1: We first show that LearnBPQ produces a hypothesis that is consistent with the sample S . The hypothesis produced is consistent with the positive examples of S since the algorithm does not terminate until all positive examples of S have been removed from P (Step 4) and no point is removed unless the box about to be placed in h contains it (Step 4(b)ii). Furthermore, if the box $(\vec{0}, x)$ was placed in h , then x was a positive example (either it was in P (Step 4a) or verified to be positive (Step 4b) with the membership query $\mathbf{member}(x)$). Since x is a positive example, the box $(\vec{0}, x)$ is contained within some box of the target. Thus no negative examples (even those not in S) can be contained in any of the boxes placed in h .

We prove the hypothesis h output by LearnBPQ contains at most s boxes. Let $(\vec{0}, b_n)$ be the box added at the n th execution of Step 4. Suppose h contained more than s boxes. Then since each box placed in h must be contained within a target box, it follows that for some $i < j$, boxes $(\vec{0}, b_i)$ and $(\vec{0}, b_j)$ are both contained in some target box $(\vec{0}, b^*)$. Let p_j be the initial value of x at the j th execution of Step 4. Looking back at the i th execution of Step 4, note that p_j remained in P throughout the entire i th execution, since it is the initial value in the j th execution of Step 4. Thus, at some iteration of step 4b during the i th execution of Step 4, the point $y = p_j$ is chosen, and the query $\mathbf{member}(\text{MaxCorner}(\hat{x}, p_j))$ is made, where \hat{x} is the value of x at this moment. Note that $\mathbf{member}(\text{MaxCorner}(\hat{x}, p_j))$ must have returned the answer “no”, otherwise p_j would have been removed from P contradicting that p_j is in P at the beginning of the j th execution of Step 4. Observe that \hat{x} is in the box $(\vec{0}, b_i)$, p_j is in the box $(\vec{0}, b_j)$ and the box $(\vec{0}, b^*)$ contains both of these boxes. But $\text{MaxCorner}(\hat{x}, p_j)$ is a negative example (since $\text{MaxCorner}(\hat{x}, p_j)$ returned “no”), and is in any box that contains both boxes $(\vec{0}, b_i)$ and $(\vec{0}, b_j)$. Hence, the box $(\vec{0}, b^*)$ contains this negative example, a contradiction. Thus h contains at most s boxes.

LearnBPQ runs in polynomial time, since there are at most s iterations of the while loop, each taking $O(d \cdot m_{BPQ}(\epsilon, \delta, s, d))$ time. This completes the proof of Part 1.

Proof of Part 2: Note that the VC-dimension of BPQ is at most d (this is easily shown), and by Lemma 3.2.3 of Blumer et al. [BEHW89], the VC-dimension of $\text{BPQ}_{\cup}(s)$ is at most $2ds \log(3s)$. This completes the proof of Part 2 and hence of the theorem. \square

LearnBPQ is easily modified to obtain an algorithm to learn the union of s origin-incident boxes in E^d when all of the boxes are in any single quadrant and when the origin is shifted to any point o . We call this new algorithm LearnBAQ(S, o) (Learn Boxes Any Quadrant) which takes a sample S and a point o and learns the union of axis-parallel boxes assuming o is the origin and all the examples in S fall in some single quadrant induced by the origin o . We call the corresponding class of concepts BAQ_o . Observe that LearnBAQ(S, o) is different from LearnBPQ in that it does not draw any examples. Instead the sample S is provided as input to the algorithm. The algorithm presented in the next section calls LearnBAQ with a sample S that is large enough to ensure that with high probability, LearnBAQ(S, o) outputs a sufficiently accurate hypothesis.

4.3.2 Approximately Learning the Agreement of Boxes with Samplable Intersection

We give an algorithm to learn the agreement of s boxes in E^d (hence, an algorithm to learn boxes from a consistently ignorant teacher) when the intersection region is samplable. Our algorithm, Figure 3, has polynomial time and sample complexity in both d and s when $d = O(\log s)$.

As we mentioned earlier, our algorithm draws a sample of positive, negative, and “?” examples and learns the intersection box, treating all “?” examples as negative examples. Then, a random positive example p is chosen from the sample and the algorithm runs 2^d versions of LearnBAQ $_p$ (with p as the origin) treating all “?” examples as positive examples. While such a hypothesis is sufficient for prediction, it does not form an agreement. We can obtain a hypothesis of the appropriate form by outputting the agreement of the intersection box and the boxes in the union extended to include the intersection box.

Let OneBox(S) be a procedure that takes a sample S and returns the smallest box consistent with the examples in S . We state the main result of this section.

Theorem 16 *Let p^+ be the probability of receiving a positive example from the example oracle. LearnBoxesAgreement is a PAC-memb algorithm for learning the agreement of s axis-parallel boxes in E^d that has sample complexity $m = O\left(\frac{4^d}{\epsilon^2} \log \frac{2^d}{\delta} + \frac{4^d}{\epsilon^2} ds \log s \log \frac{2^d}{\epsilon} + \frac{1}{p^+} \log \frac{1}{\delta}\right)$, and time complexity $O(sd2^d m)$.*

Before proving the theorem, we introduce some definitions and prove a technical lemma. Define a quadrant Q to be *significant* if $\Pr(\text{a random example is in } Q) > \frac{\epsilon}{2^{d+1}}$. We show that when LearnBAQ is applied to a significant quadrant it produces a hypothesis with error at most $\frac{\epsilon}{2 \cdot 2^d}$ (with high probability). Thus since there are 2^d quadrants, the overall error caused by the calls to LearnBAQ is at most $\epsilon/2$ (with high probability). Let p^+ be the probability of receiving a positive example. Let $m_{BPQ}(\epsilon, \delta)$ be the value given in Step 1 of Figure 2 and let m_{LBA_1} (respectively, m_{LBA_2}) be the value given in Step 1 (respectively, Step 3) of Figure 3. (So as to simplify notation, we have omitted some parameters in m_{BPQ} , m_{LBA_1} and m_{LBA_2} .) Intuitively, the lemma states that by drawing sufficiently large samples, it is possible to ensure that the probability that a positive example is not drawn is small, the probability that the intersection box has large error is small, and the probability that there are not enough examples for LearnBAQ to output a “good” hypothesis is small. More formally,

Lemma 17 *Let S_1 be a random sample of size m_{LBA_1} , and S_2 be a different random sample of size m_{LBA_2} . Then there are enough examples in S_1 and S_2 to ensure that:*

1. $\Pr(\text{A positive example in } S_1 \text{ is not drawn}) \leq \frac{\delta}{3}$.
2. $\Pr(\text{The procedure OneBox}(T) \text{ produces a hypothesis with error more than } \frac{\epsilon}{2}) \leq \frac{\delta}{3}$.

LearnBoxesAgreement($s, d, \epsilon, \delta, p^+$)

1. Draw a sample S_1 of size $m_{LBA_1}(\delta, p^+) := \frac{1}{p^+} \ln \frac{3}{\delta}$
2. If there are no positive examples in S_1 halt and report failure. Else, let p be an arbitrarily selected positive example from S_1 .
3. Draw a sample S_2 of size $m_{LBA_2}(s, d, \epsilon, \delta) := \max \left\{ \frac{8}{\epsilon} \log \frac{6}{\delta}, \frac{16d}{\epsilon} \log \frac{26}{\epsilon}, \frac{2^{2d+5}}{\epsilon^2} \log \frac{3 \cdot 2^{d+1}}{\delta}, \frac{sd \cdot 2^{2d+7}}{\epsilon^2} \log(3s) \log \frac{13 \cdot 2^{d+1}}{\epsilon} \right\}$
4. Let T be the set of examples obtained by relabeling all “?” examples of S_2 as negative.
5. $(i_1, i_2) := \text{OneBox}(T)$
6. $H := \{(i_1, i_2)\}$ /* Initialize the hypothesis H to be the intersection box. */
7. Partition E^d into 2^d quadrants using the example p selected in Step 2. Let Q be the set of 2^d quadrants induced by considering p as the origin.
8. For each quadrant q in Q
 - (a) Let S_q be the examples from S that fall in quadrant q relabeled so that all “?” examples are positive.
 - (b) $B := \text{LearnBAQ}(S_q, p)$
 - (c) $B := \text{Boxes in } B \text{ extended so that they include the intersection box } (i_1, i_2).$
 - (d) $H := H \cup B$
9. Output Agree_H

Figure 3: The algorithm LearnBoxesAgreement for learning the agreement of a set of axis-parallel boxes with samplable intersection region.

3. For a particular significant quadrant Q ,

$$\Pr(S_2 \text{ contains fewer than } m_{BPQ}(\frac{\epsilon}{2^{d+1}}, \frac{\delta}{3 \cdot 2^d}) \text{ examples in } Q) \leq \frac{\delta}{3 \cdot 2^d}.$$

Proof of Part 1: Since p^+ is the probability of drawing a positive example, the probability of not drawing a positive example in a sample of size m is at most $(1 - p^+)^m$. Then using the inequality $(1 - x)^{\frac{1}{x}} \leq e^{-1}$, we have:

$$\Pr\left(\text{no positive example in a sample of size } \frac{1}{p^+} \ln \frac{3}{\delta}\right) \leq (1 - p^+)^{\frac{1}{p^+} \ln \frac{3}{\delta}} \leq \frac{\delta}{3}$$

Since $m_{LBA_1} = 1/p^+ \ln 3/\delta$, Part 1 follows.

Proof of Part 2: Observe that the intersection region is a d -dimensional axis-parallel box and the VC-dimension of a d -dimensional axis-parallel box is $2d$. Hence, by a direct application of Theorem 2.1 in Blumer et al. [BEHW89], the probability that the procedure OneBox(T) (in step 5) produces a hypothesis that has error more than $\frac{\epsilon}{3}$ is at most $\frac{\delta}{3}$, provided that $|T| \geq \max\left\{\frac{8}{\epsilon} \log \frac{6}{\delta}, \frac{16d}{\epsilon} \log \frac{26}{\epsilon}\right\}$. But $|T| = |S_2| = m_{LBA_2}$, which was specifically chosen so that $|T|$ satisfies this inequality, and Part 2 follows.

Proof of Part 3: Let $LE(p, m, x)$ denote the probability of at most x successes in m independent trials of a Bernoulli random variable with probability of success p . The probability in the statement of Part 3 is bounded above by:

$$LE\left(\frac{\epsilon}{2^{d+1}}, m_{LBA_2}, m_{BPQ}\left(\frac{\epsilon}{2^{d+1}}, \frac{\delta}{3 \cdot 2^d}\right)\right) \quad (1)$$

since each example is independently drawn and falls in Q with probability $p \geq \frac{\epsilon}{2^{d+1}}$ (because Q is significant). By applying a version of Chernoff bounds presented in [AV79], we know:

$$LE(p, m, pm/2) \leq e^{-mp/8} \quad (2)$$

It is easily verified that $m_{BPQ}(\frac{\epsilon}{2^{d+1}}, \frac{\delta}{3 \cdot 2^d}) \leq p \cdot m_{LBA_2}/2$. Thus by substituting $p = \frac{\epsilon}{2^{d+1}}$, $m = m_{LBA_2}$ into Equation (2) and using this observation, we can apply it to Equation (1) to obtain that for a particular significant quadrant Q :

$$\begin{aligned} \Pr\left(S_2 \text{ has } < m_{BPQ}\left(\frac{\epsilon}{2^{d+1}}, \frac{\delta}{3 \cdot 2^d}\right) \text{ examples in } Q\right) &\leq LE\left(\frac{\epsilon}{2^{d+1}}, m_{LBA_2}, m_{BPQ}\left(\frac{\epsilon}{2^{d+1}}, \frac{\delta}{3 \cdot 2^d}\right)\right) \\ &\leq LE\left(\frac{\epsilon}{2^{d+1}}, m_{LBA_2}, \frac{\epsilon}{2^{d+1}} \frac{m_{LBA_2}}{2}\right) \\ &\leq e^{-m_{LBA_2}(\epsilon/2^{d+4})}. \end{aligned}$$

which is bounded by $\frac{\delta}{3 \cdot 2^d}$ since $m_{LBA_2} \geq \frac{2^{d+4}}{\epsilon} \ln \frac{3 \cdot 2^d}{\delta}$. \square

Proof of Theorem 16: Since we draw m_{LBA_1} examples in Step 1 of the algorithm given in Figure 3, by Part 1 of Lemma 17, the probability that the algorithm fails in Step 2 (equivalently,

the probability no positive examples are drawn) is at most $\frac{\delta}{3}$. Also, since we draw m_{LBA_2} examples, the probability that the intersection box found in Step 5 of the algorithm has error more than $\frac{\epsilon}{2}$ is at most $\frac{\delta}{3}$ by Lemma 17, Part 2. Observe that any insignificant quadrant contributes error at most $\frac{\epsilon}{2^{d+1}}$ and the probability that any significant quadrant contributes error more than $\frac{\epsilon}{2^{d+1}}$ is at most $\frac{\delta}{3 \cdot 2^d}$, by Theorem 15. Since there are 2^d quadrants total, when LearnBAQ is run on all of the 2^d quadrants in Step 8, the probability that it outputs a hypothesis with error greater than $2^d \cdot \frac{\epsilon}{2^{d+1}} = \frac{\epsilon}{2}$ is at most $2^d \cdot \frac{\delta}{3 \cdot 2^d} = \frac{\delta}{3}$ by Part 3 of Lemma 17.

Suppose we only learn the intersection box (i_1, i_2) with the procedure OneBox and the union of boxes U in each of the 2^d quadrants with LearnBAQ. Further, suppose we use the following algorithm as our hypothesis: If an example x lies in the intersection box (i_1, i_2) , label x positive, otherwise, if x lies in one of the boxes in the union U , label x “?”, otherwise label x negative. We compute the error of the above hypothesis on the distribution over examples labeled positive, negative, and “?”. Since the procedure OneBox returns the tightest box around the positive examples, its only error is in misclassifying positive examples of the target. By Lemma 17, Part 2, the probability that (i_1, i_2) differs from the target intersection box by more than $\frac{\epsilon}{2}$ is at most $\frac{\delta}{3}$. Also, since for each quadrant q , procedure LearnBAQ outputs a union of boxes in q that is contained in the target union of boxes in q , its only error is in misclassifying “?” examples of the target. By Lemma 17, Part 3, the probability that the union of boxes in all 2^d quadrants differs from the target union of boxes by more than $\frac{\epsilon}{2}$ is at most $\frac{\delta}{3}$. Note that no error is made misclassifying negative examples of the target. Finally, since we must have a positive example to run the algorithm, using Lemma 17, Part 1, the probability that the final hypothesis has error more than $2 \cdot \frac{\epsilon}{2} = \epsilon$ is at most $3 \cdot \frac{\delta}{3} = \delta$.

But, our algorithm does not output a ternary classifier as described above (since our goal is to output an agreement). We argue that the agreement that is instead output in Step 9, Agree_H, classifies all examples as this ternary classifier thus satisfying the PAC criterion. First note that by extending the boxes to include the intersection box, we ensure that all points in the intersection box (i_1, i_2) are classified as positive by Agree_H. Hence, Agree_H classifies the positive examples as in the above ternary hypothesis. Since the union of origin-incident boxes in a particular quadrant generated by LearnBAQ is a subset of the target union of boxes for that quadrant, extending the boxes to include the intersection box will *not* cause any box in the final agreement output to contain a negative example. Thus, the “?” examples of Agree_H are exactly those examples in a box of the union U that are not in the intersection box (i_1, i_2) , and consequently, the “?” and negative examples of Agree_H are exactly as in the ternary classifier. We conclude that the probability that the final agreement output by the algorithm on Step 9 has error more than ϵ is at most δ .

To compute the time complexity, observe that lines 2 and 4 require $O(m_{LBA_1} + m_{LBA_2})$ time, and line 5 requires $O(dm_{LBA_2})$ time. Step 6 and 7 require $O(1)$ time. For the 2^d iterations of the

while loop in line 8, line 8a requires $O(m_{LBA_2})$ time, line 8b requires $O(sdm_{LBA_2})$ time and lines 8c and 8d require $O(m_{LBA_2})$ time. Thus the total running time is $O(m_{LBA_1} + sd2^d m_{LBA_2})$. \square

5 Negative Results

In this section we explore the non-learnability of some unrestricted blurry concept classes. We demonstrate that not every class known to be learnable from an omniscient teacher is necessarily learnable from a consistently ignorant teacher. (In other words, the learnability of \mathcal{C} may not imply the learnability of $\mathcal{C}_?$.) In particular, while the classes of ℓ -term DNF formulas, decision trees, and Horn sentences are known to be PAC-memb learnable [Ang87a, Bsh95, AFP92], we show here that learning their blurry counterparts is as hard as learning (non-blurry) DNF. Since the learnability of DNF is a widely attacked open problem in computational learning theory, we have evidence that learning blurry unrestricted versions of these classes may be hard. And, while DFAs are learnable from omniscient teachers [Ang87b], we show that blurry DFAs are not learnable under widely accepted cryptographic assumptions. Recall that Theorem 11 showed that for decision trees and DFAs, the learning problem is no easier even when the set of positive examples is guaranteed to be nonempty. Thus for these classes, the negative results presented here hold even under those circumstances. We leave open the question of whether knowledge of a single positive example can facilitate learning the class of blurry ℓ -term DNF expressions.

Observation 18 *For $\mathcal{C} \in \{\ell\text{-term DNF formulas, decision trees}\}$, $\mathcal{L}(\text{DNF}) \not\leq \mathcal{L}(\mathcal{C}_?)$.*

Proof: We show how an algorithm for learning blurry ℓ -term DNF formulas or blurry decision trees can be used to learn blurry monomials. Since any algorithm for learning blurry monomials can be used to learn non-blurry DNF formulas (Observation 8), the result follows.

A monomial is a 1-term DNF formula, so an algorithm for learning blurry ℓ -term DNF formulas is in fact a blurry monomial learning algorithm.

Every monomial has a small representation as a decision tree (with a single “+” leaf, reached by a single long branch in the tree that corresponds to the literals in the monomial). Consequently, an algorithm for learning blurry decision trees also immediately gives an algorithm for learning blurry monomials. \square

Observation 19 *Blurry DFAs are not learnable under standard cryptographic assumptions.⁹*

Proof: For \mathcal{C} the class of DFAs, we show how an algorithm \mathcal{A} for $\mathcal{C}_?$ can be used to learn the union of DFAs. Since learning the union of DFAs is not possible under cryptographic assumptions [AK95], the result follows.

⁹For example, assuming the intractability of inverting RSA encryption, factoring Blum integers, or determining quadratic residuosity.

To learn the union of DFAs, M_1, \dots, M_t , we run the algorithm \mathcal{A} for $\mathcal{C}_?$ on the target $f_? = \text{Agree}_{\{M_1, \dots, M_t, M_\emptyset\}}$ where M_\emptyset is the DFA that rejects every string. Observe that since M_\emptyset does not accept any strings, $f_?$ is a two-valued function – in particular, $f_?$ labels a string x “?” if and only if the union of M_1, \dots, M_t labels x “+”. Thus, any algorithm that learns blurry DFAs can be used to learn the union of DFAs by simply treating all “+” examples as “?” examples. \square

Next we show that learning blurry Horn sentences is as hard as learning DNF formulas. Since every monomial is in fact a Horn sentence (with clauses of size one), this follows immediately from Observation 8, which shows that learning blurry monomials (hence Horn sentences), without the restriction that there be a positive example, is as hard as learning (boolean valued) DNF formulas.

However, as we have seen with monomials and boxes, if there are positive examples in the agreement, then this information can be valuable in distinguishing between the negative and “?” examples. We show that for Horn sentences, even when the set of positive examples of the agreement is nonempty and samplable, learning remains as hard as learning the class of (boolean-valued) DNF formulas. Thus, we obtain the following stronger “negative” result than the one that follows from Observation 8.

Theorem 20 *PAC-memb learning the agreement of Horn sentences for which the set of positive examples is samplable is as hard as PAC-memb learning the class of DNF formulas.*

Proof: In proving the theorem, we define the following classes of concepts:

DHF _{n}	Disjunctive Horn Form —disjunctions of Horn sentences over n variables. (e.g., $f = [(\overline{v}_1 \vee \overline{v}_3 \vee v_5) \wedge (\overline{v}_2 \vee v_4)] \vee [(\overline{v}_1 \vee \overline{v}_2) \wedge (v_6)]$).
DHF-1pos _{n,p}	DHF _{n} formulas with exactly one positive example p satisfying every disjunct (p is known to the learner).
agree-Horn-1pos _{n,p}	Agreement of Horn sentences over n variables with exactly one positive example p (p is known to the learner).
agree-Horn-pos-samp _{n}	Agreement of Horn sentences over n variables (with a samplable set of positive examples).

We prove the theorem through a sequence of prediction preserving reductions, showing that PAC-memb learning DNF formulas reduces to PAC-memb learning DHF formulas (i.e., the first problem on the list) and also that PAC-memb learning each problem on the list reduces to PAC-memb learning the next problem on the list.

(i) $\mathcal{L}(\mathbf{DNF}) \preceq \mathcal{L}(\mathbf{DHF})$

Every DNF formula f is in fact a DHF formula: f is a disjunction of terms, and each term can be viewed as a Horn sentence — a conjunction of clauses, each containing a single literal (hence, at most one unnegated literal). Thus, f is a disjunction of Horn sentences. So, if we had a learning algorithm for DHF formulas, we could use it to learn any DNF formula, f , by simply running the DHF learning algorithm on the same formula f . \square

(ii) $\mathcal{L}(\mathbf{DHF}) \preceq \mathcal{L}(\mathbf{DHF-1pos})$

We construct an algorithm \mathcal{A} to learn a formula f in \mathbf{DHF}_n using the algorithm \mathcal{A}^+ that learns a formula f^+ in $\mathbf{DHF-1pos}_{n+1,p}$. In order to learn an unknown formula f in \mathbf{DHF}_n , \mathcal{A} will run \mathcal{A}^+ on examples of the formula f^+ in $\mathbf{DHF-1pos}_{n+1,p}$, in which the zero vector $\vec{0}$ is the single known positive example. The definition of f^+ ensures that such examples are easily constructed from examples of f , without knowledge of f . The target f^+ of $\mathbf{DHF-1pos}_{n+1,p}$ is of the form $\alpha \vee \beta$, where α is f with the extra literal $\overline{v_0}$ disjoined to every clause of every Horn sentence of the disjunction of f , and β is the Horn sentence $\overline{v_0} \wedge \overline{v_1} \wedge \cdots \wedge \overline{v_n}$. For the example DHF formula f given in the definition of DHF formulas, $f^+ = [(\overline{v_0} \vee \overline{v_1} \vee \overline{v_3} \vee v_5) \wedge (\overline{v_0} \vee \overline{v_2} \vee v_4)] \vee [(\overline{v_0} \vee \overline{v_1} \vee \overline{v_2}) \wedge (\overline{v_0} \vee v_6)] \vee (\overline{v_0} \wedge \cdots \wedge \overline{v_8})$.

Clearly, the only example that can satisfy every disjunct of f^+ is $\vec{0}$, since this is the only one that satisfies β . Further, $\vec{0}$ satisfies every Horn sentence in α because $\overline{v_0}$ appears in every clause of every Horn sentence in α .

Observe that x is a positive example of f if and only if $1 \cdot x$ is a positive example of f^+ . To see this, note that $f^+(\langle 1, x_1, \dots, x_n \rangle) = 1$ iff α evaluates to 1, and f is exactly the formula obtained by reducing α after setting v_0 to 1. Consequently, when \mathcal{A}^+ (the algorithm for DHF-1pos) requests a random example, \mathcal{A} (the algorithm for DHF) obtains a random example $\langle x_1, \dots, x_n \rangle$ with classification label ℓ , and returns the example $\langle 1, x_1, \dots, x_n \rangle$ with classification label ℓ .

If \mathcal{A}^+ makes a membership query on an example $\langle x_0, x_1, \dots, x_n \rangle$, \mathcal{A} checks the value of x_0 . If $x_0 = 0$ then \mathcal{A} returns the label “+” because every clause in every Horn sentence of α is satisfied. If $x_0 = 1$ then \mathcal{A} returns the result of a membership query on the example $\langle x_1, \dots, x_n \rangle$.

Once \mathcal{A}^+ terminates with the hypothesis h^+ , \mathcal{A} can predict the label of any (n -bit) example x by setting the extra variable $v_0 = 1$ and evaluating h^+ on the example $1 \cdot x$. Note that setting v_0 to 1 falsifies β and “forces” α to be f .

(iii) $\mathcal{L}(\mathbf{DHF-1pos}) \preceq \mathcal{L}(\mathbf{agree-Horn-1pos})$

It is here that we switch from learning a standard boolean-valued concept to learning an

agreement (i.e., a blurry concept). We construct an algorithm \mathcal{A}^+ to learn any function f^+ in $\text{DHF-1pos}_{n,p}$ given an algorithm $\mathcal{A}_?^+$ to learn any $f_?^+$ in $\text{agree-Horn-1pos}_{n,p}$. Suppose f^+ is the target function. We use $\mathcal{A}_?^+$ to learn a related ternary function $f_?^+$ in $\text{agree-Horn-1pos}_{n,p}$, defined as follows: $f_?^+$ labels p positive, the remaining positive examples of f^+ “?”, and all other examples negative. When $\mathcal{A}_?^+$ requests a random example, \mathcal{A}^+ draws a random example x and returns x with a label determined as follows. If $x = p$ then \mathcal{A}^+ returns positive. If $x \neq p$ and x is a positive example then \mathcal{A}^+ returns “?”. Otherwise \mathcal{A}^+ returns negative. When $\mathcal{A}_?^+$ makes a membership query on example x , \mathcal{A}^+ returns positive if $x = p$. If $x \neq p$ then if a membership query on x returns the label positive then \mathcal{A}^+ returns the label “?”. Otherwise \mathcal{A}^+ returns the label negative. Once $\mathcal{A}_?^+$ terminates with hypothesis $h_?^+$, \mathcal{A}^+ can predict the label of any example x by outputting positive when $h_?^+$ labels x positive or “?”, and outputting negative otherwise.

Observe that the error of the hypothesis h^+ output by \mathcal{A}^+ on any distribution D on \mathcal{X}_n with examples labeled by f^+ is precisely the error that the hypothesis $h_?^+$ of $\mathcal{A}_?^+$ has on D with examples labeled according to $f_?^+$.

(iv) $\mathcal{L}(\text{agree-Horn-1pos}) \triangleleft \mathcal{L}(\text{agree-Horn-pos-samp})$

We construct an algorithm $\mathcal{A}_?^+$ that learns any $f_?^+$ in $\text{agree-Horn-1pos}_{n,p}$ using algorithm $\mathcal{A}_?$ that learns $f_?$ in $\text{agree-Horn-pos-samp}_n$. The target of both algorithms is $f_?^+$ (an agreement of Horn sentences with exactly one positive example p). When $\mathcal{A}_?$ requests a random example, $\mathcal{A}_?^+$ flips a fair coin and with probability $\frac{1}{2}$ returns p as a positive example (and so the set of positive examples, which in this case has one element, is samplable). Otherwise, $\mathcal{A}_?^+$ draws a random example and returns it to $\mathcal{A}_?$. When $\mathcal{A}_?$ makes a membership query on example x , $\mathcal{A}_?^+$ returns the label of a membership query on x . Clearly, $\text{agree-Horn-pos-samp}$ is a generalization of agree-Horn-1pos , and thus at least as hard.

It follows from this sequence of reductions that PAC-memb learning the agreement of Horn sentences when the set of positive examples is samplable is as hard as PAC-memb learning the class of DNF formulas. \square

Finally, we strengthen this result by using the hardness result of Angluin and Kharitonov [AK95] which shows, under the assumption that one-way functions exist, that membership queries do not help in learning DNF formulas.

Corollary 21 *PAC-memb learning the agreement of Horn sentences for which the intersection region is samplable is as hard as PAC learning the class of DNF formulas (assuming that one-way functions exist).*

6 Learning Blurry Concepts in the Exact Learning Model

We demonstrate that many of the results we have presented in the PAC model also hold in the exact model. Recall that an equivalence query on a hypothesis h returns “yes” if h is equivalent to the target and otherwise returns an example on which the hypothesis and target disagree. Note that a counterexample (from a consistently ignorant teacher) to a hypothesis h may be one that the target labels “?” and the hypothesis does not. We begin by defining exact learning from a consistently ignorant teacher.

Definition 5 $\mathcal{C}_?$ is exact learnable (alternatively, \mathcal{C}_n is exact learnable from a consistently ignorant teacher) if there exists an algorithm $\mathcal{A}_?$ such that for all blurry concepts $f_?$ in $\mathcal{C}_?$, $\mathcal{A}_?$ outputs a hypothesis $h_? = f_?$ making at most polynomial in $|f_?|$ and n equivalence queries. If $\mathcal{A}_?$ also makes (polynomially many) membership queries, we say $\mathcal{C}_?$ is exact-memb learnable.

We show how $\text{Nested}_?(C_S, C_G)$ is learnable in the exact model assuming the classes C_S and C_G are exactly learnable. Let \mathcal{A}_S and \mathcal{A}_G be the learning algorithms for C_S and C_G , respectively. Our algorithm to learn $\text{Nested}_?(C_S, C_G)$ first runs the algorithm \mathcal{A}_S until it poses an equivalence query on h_S and then runs \mathcal{A}_G until it poses an equivalence query on h_G . Then the $\text{Nested}_?(C_S, C_G)$ algorithm poses the equivalence query $\text{Agree}_{\{h_S, h_G\}}$. If there is a positive or negative counterexample to this equivalence query then it is passed to \mathcal{A}_S and \mathcal{A}_G . If there is a counterexample for which the proper label is “?” then it is passed to \mathcal{A}_S (respectively, \mathcal{A}_G) as a negative (respectively, positive) example. Each of these algorithms can then check if that example is a counterexample to its hypothesis and if so continue running the algorithm until the next equivalence query is posed. If it is not a counterexample, the same hypothesis is used for subsequent equivalence queries. Note that the counterexample received from the equivalence query must be a counterexample to at least one of h_S or h_G . Thus, if C_S and C_G are exact (respectively, exact-memb) learnable then $\text{Nested}_?(C_S, C_G)$ is exact (respectively, exact-memb) learnable.

As an immediate consequence we have the analog of Theorem 6 (and Corollaries 9 and 12) for the exact model. That is, if \mathcal{C} is a concept class for which \mathcal{C}_\cap and \mathcal{C}_\cup are exact-memb (respectively exact) learnable in polynomial time, then $\mathcal{C}_?$ is exact-memb (respectively exact) learnable in polynomial time. Also, all of the positive learning results cited either are given in the exact setting, or have exact learning analogs. Consequently, exact-memb variants of Corollaries 7, 10, and 13 all hold.

We demonstrate that the results of Theorem 15 can also be obtained in the exact-memb learning model in the discretized space. Namely, there is an efficient algorithm that uses equivalence and membership queries to exactly learn the union of s boxes in one quadrant in the discretized space $\{1, \dots, n\}^d$. The algorithm maintains a hypothesis that is a subset of the true union of boxes in the discretized plane. For every positive counterexample obtained by an equivalence query,

the algorithm performs a binary search away from the origin in each dimension so as to find the “border” (i.e., the point x which is a positive example such that every point away from the origin one unit in each dimension is negative). More specifically, if the positive counterexample p has i th coordinate equal to ℓ (i.e., $p_i = \ell$) then the learner locates the “border” by performing a binary search in the following way: the learner poses a membership query on the point with i th-coordinate $\ell' = \lceil (n + \ell)/2 \rceil$ and with all other coordinates values unchanged. If this point is positive then the binary search continues between ℓ' and n . And if this point is negative then the binary search continues between ℓ and $\ell' - 1$. Once the “border” point x is found in this way, the box $(\vec{0}, x)$ is then added to the hypothesis. It is easily seen that using $O(d \log n)$ membership queries a previously undiscovered corner of one of the boxes defining the target concept is found from each counterexample. Thus the time complexity and number of membership queries made is $O(sd \log n)$ and the number of equivalence queries made is $O(s)$.

We describe how to extend the result of Theorem 16 to exactly learn the agreement of boxes in the discretized domain. As we did when PAC-memb learning the agreement of boxes, we assume that the intersection is non-empty and furthermore, that the learner is provided a positive example with which it divides $\{1, \dots, n\}^d$ into 2^d quadrants. The basic idea of the algorithm to exactly learn the agreement of boxes is to apply Theorem 5 (as modified for the exact model) in each quadrant where \mathcal{C}_S is the intersection and \mathcal{C}_G is the union of the portion of the target concept that falls in the given quadrant. As we saw in the analysis of LearnBoxesAgreement, in each quadrant we have at most s origin-incident boxes. We can use the algorithm of Chen and Maass [CM92] to learn \mathcal{C}_S and the algorithm described above to learn \mathcal{C}_G . We need just one additional modification to this procedure — when receiving a positive counterexample x , the boxes in the hypotheses for each of the $2^d - 1$ quadrants that do not contain x must be extended so that they contain x .

7 Conclusions

We have introduced a model in which a learning algorithm for a concept class \mathcal{C} interacts with a teacher who labels some examples “don’t know” (i.e., the teacher is ignorant), but does so in a manner that ensures the learner cannot infer the label of “don’t know” examples from other positive and negative examples and knowledge of \mathcal{C} (i.e., the teacher is consistent with \mathcal{C}). We presented a result that allows us to “plug in” results involving learning from an omniscient teacher in order to learn from a consistently ignorant teacher. An essential result showed that if intersections and unions of concepts from \mathcal{C} are learnable from an omniscient teacher, then \mathcal{C} is learnable from a consistently ignorant teacher. In the process of proving the above result, we introduced the notion of agreements, and showed that learning from a consistently ignorant teacher is equivalent to learning agreements of sets of concepts from \mathcal{C} . We summarize our results in Table 1.

\mathcal{C}	$\mathcal{L}(\mathcal{C}_?^k)$	$\mathcal{L}(\mathcal{C}_?^+)$	$\mathcal{L}(\mathcal{C}_?)$
Horn clauses	yes	yes	yes ^a
CLASSIC	yes	yes	yes ^a
ℓ -term DNF (ℓ -clause CNF)	yes ^b	open	\succeq DNF ^c
monomials	yes ^b	yes ^d	\succeq DNF ^e
monotone DNF (CNF)	yes ^b	equiv to \rightarrow^f	open
decision trees	yes ^b	equiv to \rightarrow^f	\succeq DNF ^c
DFAs	yes ^b	equiv to \rightarrow^f	no, crypto ^g
boxes in E^d	yes ^h	$\text{poly}(s, 2^d, \frac{1}{\epsilon}, \frac{1}{\delta})^i$	open
Horn Sentences	open	\succeq DNF ^j	\succeq DNF

Table 1: For each class \mathcal{C} , the table shows the status of the PAC-memb learnability of $\mathcal{C}_?^k$, $\mathcal{C}_?^+$, and $\mathcal{C}_?$. The entries “ \succeq DNF” denote the problem is as hard as learning DNF formulas (without membership queries, if one-way functions exist). The entries “equiv to \rightarrow ” under $\mathcal{L}(\mathcal{C}_?^+)$ indicate that the problem is equivalent to $\mathcal{L}(\mathcal{C}_?)$. The entry “no, crypto” indicates the class is not learnable under standard cryptographic assumptions. The superscripts indicate where in the paper the result is given, according to the following key: (a) Corollary 7; (b) Corollary 13; (c) Observation 18; (d) Corollary 10; (e) Observation 8; (f) Theorem 11; (g) Observation 19; (h) Corollary 14 (PAC-learnable without membership queries); (i) Theorem 16, assuming that the positive examples are samplable; (j) Theorem 20, even when the positive examples are samplable.

In addition to the open problems listed in the table, we list some other interesting unanswered problems:

- We have only investigated blurriness in the classification of examples and not blurriness in the examples themselves. In particular, we have generally considered functions that map $\{0, 1\}^n$ onto $\{+, -, ?\}$ and it may be interesting to consider functions that map blurry examples $\{0, 1, ?\}^n$ onto clear labels $\{+, -\}$ or onto blurry labels $\{+, -, ?\}$.
- We have not investigated learning blurry concepts with read restrictions. For example, are blurry read-once formulas or blurry read- k sat- j DNF formulas¹⁰ learnable?
- Our algorithm to learn the agreement of boxes (LearnBoxesAgreement) only uses membership queries when learning the union of origin-incident boxes in a single quadrant (BAQ). Since LearnBoxesAgreement requires time polynomial in s and 2^d , BAQ can run in time polynomial in s and 2^d without affecting the asymptotic running time of LearnBoxesAgreement. We leave open the question of whether BAQ is learnable without membership queries in time polynomial in s and 2^d (in the continuous or discretized domain).

Acknowledgements

We thank Dana Angluin and Haym Hirsh for helpful conversations regarding this work. We also thank William Cohen for suggesting that CLASSIC was learnable from a consistently ignorant teacher. The anonymous referees made a number of helpful comments.

References

- [AFP92] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.
- [AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40:185–210, 1993.
- [AHP92] H. Aizenstein, L. Hellerstein, and L. Pitt. Read-thrice DNF is hard to learn with membership and equivalence queries. In *Proc. of the 33rd Symposium on the Foundations of Comp. Sci.*, pages 523–532. IEEE Computer Society Press, Los Alamitos, CA, 1992. (A revised manuscript, with additional results, is Aizenstein, Hegedus, Hellerstein, and Pitt, “Complexity Theoretic Hardness Results for Query Learning”, to appear in *Computational Complexity*.)

¹⁰A formula is a read- k sat- j DNF if each variable occurs at most k times and each positive example satisfies at most j terms.

- [AK94] D. Angluin and M. Krikis. Learning with malicious membership queries and exceptions. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 57–66. ACM Press, New York, NY, 1994.
- [AK95] D. Angluin and M. Kharitonov. *Journal of Computer and System Sciences*, 50, 1995. (Special issue for the 23rd Annual ACM Symposium on Theory of Computing.)
- [AKMW95] P. Auer, S. Kwek, W. Maass, and M. Warmuth. On-line prediction of depth two linear threshold circuits. Manuscript in preparation.
- [AL88] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.
- [Ang87a] D. Angluin. Learning k-term DNF formulas using queries and counterexamples. Technical Report YALEU/DCS/RR-559, Department of Computer Science, Yale University, August 1987.
- [Ang87b] D. Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75(2):87–106, November 1987.
- [Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.
- [Ang90] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.
- [Ang94] D. Angluin. Exact learning of μ -DNF formulas with malicious membership queries. Technical Report YALEU/DCS/TR-1020, Yale University, March 1994.
- [AP91] H. Aizenstein and L. Pitt. Exact learning of read-twice DNF formulas. In *Proc. 32th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 170–179. IEEE Computer Society Press, 1991.
- [AS94] D. Angluin and D. Slonim. Randomly fallible teachers: Learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14(7):7–26, 1994. Special issue for the Fourth Ann. Workshop on Comput. Learning Theory (Santa Cruz, CA., July 1991).
- [AV79] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18(2):155–193, 1979.
- [BCH94] N. Bshouty, Z. Chen, and S. Homer. On learning discretized geometric concepts. In *Proc. of the 35th Symposium on the Foundations of Comp. Sci.*, pages 54–63, IEEE Computer Society Press, Los Alamitos California, November 1994.

- [BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.
- [BGGM94] N. Bshouty, P. Goldberg, S. Goldman, and D. Mathias. Exact learning of discretized geometric concepts. Technical Report WUCS-94-19, Technical Report, Washington University, July 1994.
- [BGM95] N. Bshouty, S. Goldman, and D. Mathias. Noise-tolerant parallel learning of geometric concepts. In *Proc. 8th Annu. ACM Workshop on Comput. Learning Theory*, pages 345–352. ACM Press, New York, NY, 1995.
- [BGMST95] N. Bshouty, S. Goldman, D. Mathias, S. Suri, and H. Tamaki. Noise-tolerant distribution-free learning of general geometric concepts. Manuscript in preparation.
- [BHH92] N. Bshouty, T. Hancock, and L. Hellerstein. Learning Boolean read-once formulas with arbitrary symmetric and constant fan-in gates. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 1–15. ACM Press, New York, NY, 1992.
- [BHH95] N. Bshouty, T. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM Journal of Computing*, 24(4):706–735, 1995.
- [Blu94] A. Blum. Separating distribution-free and mistake-bound learning models over the Boolean domain. *SIAM J. Comput.*, 23(5):990–1000, October 1994.
- [BR92] A. Blum and S. Rudich. Fast learning of k -term DNF formulas with queries. In *Proc. of the 24th Symposium on Theory of Computing*, pages 382–389. ACM Press, New York, NY, 1992.
- [Bsh95] N. Bshouty. Exact learning Boolean functions via the monotone theory. *Inform. Comput.* 123(1):146-153, 15 November 1995.
- [CH94a] Z. Chen and S. Homer. The bounded injury priority method and the learnability of unions of rectangles. Unpublished, May 1994.
- [CH94b] W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*. Morgan Kaufmann, 1994.
- [CH94c] W. Cohen and H. Hirsh. Learnability of description logics with equality constraints. *Machine Learning*, 17(2/3):169–199, November/December 1994. Special issue for the Fifth Ann. Workshop on Comput. Learning Theory (Pittsburgh, PA., July 1992).

- [Che93] Z. Chen. Learning unions of two rectangles in the plane with equivalence queries. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 243–252. ACM Press, New York, NY, 1993.
- [CM92] Z. Chen and W. Maass. On-line learning of rectangles. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 16–28. ACM Press, New York, NY, 1992.
- [FP94] M. Frazier and L. Pitt. CLASSIC learning. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 23–34. ACM Press, New York, NY, 1994. To appear, *Machine Learning*.
- [GGM94] P. Goldberg, S. Goldman, and D. Mathais. Learning unions of boxes with membership and equivalence queries. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 198–207. ACM Press, New York, NY, 1994.
- [GKS93] S. Goldman, M. Kearns, and R. Schapire. Exact identification of circuits using fixed points of amplification functions. *SIAM J. Comput.*, 22(4):705–726, August 1993.
- [GNPS91] C. Gunter, T. Ngair, P. Panangaden, and D. Subramanian. The common order-theoretic structure of version spaces and atoms's. In *Proceedings of the National Conference on Artificial Intelligence*, pages 500–505, Anaheim, CA, July 1991. AAAI Press/MIT Press.
- [GS95] S. Goldman and R. Sloan. Can PAC learning algorithms tolerate random noise. *Algorithmica*, 14(1), 1995.
- [Han91] T. Hancock. Learning 2μ -DNF formulas and $k\mu$ decision trees. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 199–209, San Mateo, CA, 1991. Morgan Kaufmann.
- [Hau89] D. Haussler. Generalizing the PAC model for neural net and other learning applications. Technical Report UCSC-CRL-89-30, University of California Santa Cruz, September 1989.
- [Hir91] H. Hirsh. Theoretical underpinnings of version spaces. In *Proceedings of the Twelfth Joint International Conference on Artificial Intelligence*, pages 665–670, Sydney, Australia, August 1991.
- [HKLW91] D. Haussler, M. Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. *Inform. Comput.*, 95(2):129–161, December 1991.

- [HLW94] D. Haussler, N. Littlestone, and M. Warmuth. Predicting $\{0,1\}$ functions on randomly drawn points. *Inform. Comput.*, 115(2):284–293, 1994.
- [HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.
- [Jac94] J. Jackson. Learning DNF under the uniform distribution. In *Proc. of the 35th Symposium on the Foundations of Comp. Sci.*, pages 42–53, IEEE Computer Society Press, Los Alamitos California, November 1994.
- [Kea93] 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.
- [KL93] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22:807–837, 1993.
- [KP95] S. Kwek and L. Pitt. Polynomial-time learning of intersections of halfspaces, 1995. Manuscript in preparation.
- [KS94] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. *J. of Comput. Syst. Sci.*, 48(3):464–497, 1994.
- [KV94] M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41(1):67-95, 1994.
- [Lai88] P. Laird. Learning from good and bad data. In *Kluwer international series in engineering and computer science*. Kluwer Academic Publishers, Boston, 1988.
- [LW94] P. Long and M. Warmuth. Composite geometric concepts and polynomial predictability. *Inform. Comput.*, 113(2):230–252, 1994.
- [Mit78] T. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, December 1978.
- [Mit82] T. Mitchell. Generalization as search. *Art. Int.*, 18:203–226, 1982.
- [MW95] W. Maass and M. Warmuth. Efficient learning with virtual threshold gates. In *Proc. XII International Conf. on Machine Learning*, pages 378–386, Morgan Kaufmann, San Francisco, CA, 1995.
- [PR94] K. Pillaipakkamnatt and V. Raghavan. On the limits of proper learnability of subclasses of DNF formulas. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 118–129. ACM Press, New York, NY, 1994. To appear, *Machine Learning*.

- [PR95] K. Pillaipakkamnatt and V. Raghavan. Read-twice DNF formulas are properly learnable. *Inform. Comput.* 122(2):236–267, November, 1995.
- [PW90] L. Pitt and M. Warmuth. Prediction preserving reducibility. *J. of Comput. Syst. Sci.*, 41(3):430–467, December 1990. Special issue for the *Third Annual Conference of Structure in Complexity Theory* (Washington, DC., June 1988).
- [RR95] D. Ron and R. Rubinfeld. Learning fallible deterministic finite automata. *Machine Learning*, 18(2/3):149–186, 1995. Special issue for the Sixth Ann. Workshop on Comput. Learning Theory (Santa Cruz, CA., July 1993).
- [Sak91] Y. Sakakibara. On learning from queries and counterexamples in the presence of noise. *Inform. Proc. Lett.*, 37(5):279–284, March 1991.
- [SS92] Y. Sakakibara and R. Siromoney. A noise model on learning sets of strings. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 295–302, ACM Press, New York, NY, 1992.
- [Slo88] R. Sloan. Types of noise in data for concept learning. In *Proc. 1st Annu. Workshop on Comput. Learning Theory*, pages 91–96, Morgan Kaufmann, San Mateo, CA, 1988.
- [ST94] R. H. Sloan and G. Turán. Learning with queries but incomplete information. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 237–245. ACM Press, New York, NY, 1994.
- [SV88] 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. Morgan Kaufmann, San Mateo, CA, 1988.
- [Val84] L. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.