Report Number: WUCS-98-28

1998-01-01

Learning from Examples with Unspecified Attribute Values

Sally A. Goldman, Stephen S. Kwek, and Stephen D. Scott

We introduce the UAV learning model in which some of the attributes in the examples are unspecified. In our model, an example x is classified positive (resp., negative) if all possible assignments for the unspecified attributes result in a positive (resp., negative) classification. Otherwise the classificatoin given to x is "?" (for unknown). Given an example x in which some attributes are unspecified, the oracle UAV-MQ responds with the classification of x. Given a hypothesis h, the oracle UAV-EQ returns an example x (that could have unspecified attributes) for which h(x) is incorrect. We show that any class learnable... **Read complete abstract on page 2.**

Learning from Examples with Unspecified Attribute Values

Sally A. Goldman, Stephen S. Kwek, and Stephen D. Scott

Complete Abstract:

We introduce the UAV learning model in which some of the attributes in the examples are unspecified. In our model, an example x is classified positive (resp., negative) if all possible assignments for the unspecified attributes result in a positive (resp., negative) classification. Otherwise the classificatoin given to x is "?" (for unknown). Given an example x in which some attributes are unspecified, the oracle UAV-MQ responds with the classification of x. Given a hypothesis h, the oracle UAV-EQ returns an example x (that could have unspecified attributes) for which h(x) is incorrect. We show that any class learnable in the exact model using the MQ and EQ oracles is also learnable in the UAV model using the MQ and UAV-EQ oracles as long as the counterexamples provided by the UAV-EQ oracle have a logarithmic number of unspecified attributes. We also show that any class learnable in the exact model using the MQ and EQ oracles is also learnable in the UAV model using the UAV-MQ and UAV-EQ oracles as well as an oracle to evaluate a given boolean formula on an example with unspecified attributes. (For some hypothesis classes such as decision trees and unate formulas the evaluation can be done in polynomial time without an oracle.) We also study the learnability of a universal class of decision trees under the UAV model and of DNF formulas under a representation-dependent variation of the UAV model.

Learning from Examples with Unspecified Attribute Values

Sally A. Goldman, Stephen S. Kwek and Stephen D. Scott

WUCS-98-28

December 1998

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

Learning From Examples With Unspecified Attribute Values*

Sally A. Goldman†
Dept. of Computer Science
Washington University
St. Louis, MO 63130-4899
sg@cs.wustl.edu

Stephen S. Kwek†
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-1035
kwek@eecs.wsu.edu

Stephen D. Scott†
Dept. of Computer Science and Engineering
University of Nebraska
Lincoln, NE 68588-0115
sscott@cse.unl.edu

WUCS-98-28

December 1998

Abstract

We introduce the *UAV learning model* in which some of the attributes in the examples are unspecified. In our model, an example x is classified positive (resp., negative) if all possible assignments for the unspecified attributes result in a positive (resp., negative) classification. Otherwise the classification given to x is "?" (for unknown). Given an example x in which some attributes are unspecified, the oracle UAV-MQ responds with the classification of x. Given a hypothesis h, the oracle UAV-EQ returns

*An earlier version appears in the Tenth Annual ACM Conference on Computational Learning Theory, 1997

†Supported in part by NSF NYI Grant CCR-9357707 with matching funds provided by Xerox PARC and WUTA.

an example x (that could have unspecified attributes) for which $h(x)$ is incorrect.

We show that any class learnable in the exact model using the MQ and EQ oracles is also learnable in the UAV model using the MQ and UAV-EQ oracles as long as the counterexamples provided by the UAV-EQ oracle have a logarithmic number of unspecified attributes. We also show that any class learnable in the exact model using the MQ and EQ oracles is also learnable in the UAV model using the UAV-MQ and UAV-EQ oracles as well as an oracle to evaluate a given boolean formula on an example with unspecified attributes. (For some hypothesis classes such as decision trees and unate formulas the evaluation can be done in polynomial time without an oracle.) We also study the learnability of a universal class of decision trees under the UAV model and of DNF formulas under a representation-dependent variation of the UAV model.

Keywords: Unspecified or missing attributes, exact learning, membership queries, equivalence queries, DNF formulas, decision trees

1 INTRODUCTION

Most theoretical work on learning boolean functions assumes that the examples are drawn from $\{0,1\}^n$ and each example is classified as positive ("+") or negative ("−") by the unknown target concept f. However, consider the situation of trying to predict whether or not an individual will default on a loan. There are a large number of attributes that one might consider (some about the individual and some about the individual's employer). Using standard approaches to learning boolean functions, a learning algorithm could consider each loan application as an example from $\{0,1\}^n$ where n is the number of attributes under consideration for loan applications. Let's assume that if we had correct data for all of the attributes, then such an algorithm could make a very good prediction as to whether or not the individual will default on the loan. However, in practice some attributes' values will not be provided (or specified). For example, some states may prohibit the release of certain credit information. The learner is to take an example with some unspecified attributes and reply that from the given information either (1) the individual is extremely likely to default on the loan, (2) the individual is extremely likely not to default on the loan, or (3) there is not enough information available to make a decision.

There are three key aspects in this scenario. First, if all attribute values were known then the learner could make perfect predictions after learning the target concept. Second, in any given example provided to the learner, some of the attribute values are likely to be unspecified. Finally, the learner's goal is to make a correct prediction if the values for the provided attributes are sufficient to do so, or otherwise indicate that insufficient information has been provided. Along with the scenario described above, one could imagine situations

in medical diagnosis in which these aspects occur. Another such setting comes from feature extraction for image processing. Suppose the learner is given the task of predicting whether a given image is a face. The attributes could be provided by the output of feature extractors which sometimes are unable to determine whether or not a given feature is present.

In this paper we introduce a variant of Angluin's exact learning model [3] in which some of the attribute values are left unspecified. When first defining the PAC model, Valiant [43] also suggested a variation in which some attribute values are unspecified but he did not study this variation in detail (see Section 3). In our new model, the examples are drawn from $\{0,1,*\}^n$ where "$*$" denotes unspecified. Given a standard boolean function f and an example $x \in \{0,1,*\}^n$, the classification of x given by f can naturally be viewed as either being positive (meaning that f is true regardless of the values for the unspecified attributes), negative (meaning that f is false regardless of the values for the unspecified attributes), or unknown (meaning that f could be true or false depending on the values of the unspecified attributes). Thus in our model the classification of an example is drawn from $\{+,-,?\}$ where "?" is used to denote the value of unknown. The goal of our learner is to exactly learn an unknown ternary function f over the domain $\{0,1,*\}^n$. Since there are unspecified attribute values, we refer to this model as the *UAV Exact Learning* model.

In this model an equivalence query, denoted by UAV-EQ, takes as input a hypothesis h and returns an example $x \in \{0,1,*\}^n$ (that could have any number of unspecified attributes) for which $h(x)$ is different from the correct classification for x. A corresponding model of *UAV PAC Learning* can be naturally defined. Namely, the learner is provided with labeled (from $\{+,-,?\}$) examples of f, drawn randomly according to some unknown target distribution $\mathcal{D}$ over $\{0,1,*\}^n$. The learner is also given as input ϵ and δ such that $0 < \epsilon, \delta < 1$, and an upper bound k on $|f|$. The learner's goal is to output, with probability at least $1-\delta$, a hypothesis $h \in \mathcal{C}_n$ that has probability at most ϵ of disagreeing with f on a randomly drawn example from $\mathcal{D}$. In both the exact UAV and PAC UAV models, membership queries can be added. In a UAV membership query, denoted by UAV-MQ, the learner gives the membership oracle an example x from $\{0,1,*\}^n$ and is given the value of $f(x)$ from $\{+,-,?\}$. We use MQ to denote the standard membership oracle that requires that the example $x \in \{0,1\}^n$ (and hence the output is from $\{+,-\}$), and EQ to denote the standard equivalence oracle that always returns a counterexample x from $\{0,1\}^n$. Hence, the UAV-MQ oracle is a generalization of the standard MQ oracle in which some of the attributes can be unspecified. Since, for some concept classes, an NP-hard problem must be solved by the UAV-MQ oracle, we also explore when a concept class is learnable in the UAV model using the UAV-EQ oracle and the standard membership (MQ) oracle.

We prove that *any* concept class that is learnable in the exact model using the MQ and EQ oracles is UAV exactly learnable using the MQ and UAV-EQ oracles as long as the counterexamples provided by UAV-EQ have $O(\log n)$ missing attributes where n is the number of attributes. As a corollary to this result, we prove that any concept class that is learnable in the exact model

using the MQ and EQ oracles is UAV exactly learnable using the UAV-MQ and UAV-EQ oracles as well as an oracle to evaluate a given boolean formula on an example with unspecified attributes. For some hypothesis classes such as decision trees and unate formulas (a superset of read-once formulas), this "evaluation oracle" is not needed. Also, any monotone class that is exactly learnable using just an EQ oracle is exactly learnable in the UAV model using only the UAV-EQ oracle. It is also straightforward to show that *any* class learnable in the UAV model with only the UAV-EQ oracle is learnable in the standard model using only an EQ oracle. We also show that there is a class of universal decision trees that can be efficiently learned in the UAV model using only a UAV-MQ oracle (no evaluation oracle is needed regardless of how many attributes are unspecified), yet cannot be efficiently learned in the standard exact learning model using only an MQ oracle. Likewise, known results can be applied to show that read-once formulas are UAV exactly learnable using only a UAV-MQ oracle yet are not exactly learnable in standard model using only an MQ oracle.

For a known DNF formula f, the problem of determining the classification of an example $x \in \{0,1,*\}^n$ (with $\Omega(n^c)$ unknown attributes for some constant c) is NP-complete since it involves solving a satisfiability and a non-satisfiability problem. For the concept class of DNF formulas, we study a representation-dependent variation of the UAV model (which we call the *RUAV* model) in which the teacher (and learner) can efficiently classify any example from $\{0,1,*\}^n$ with respect to a given DNF formula. In this variation, the classification of an example $x \in \{0,1,*\}^n$ for the teacher's representation of the target function f is defined as follows. If any term in f is satisfied by the known attributes in x then $f(x) = +$. If every term in f is known *not* to be satisfied by the known attributes from x then $f(x) = -$. Otherwise, $f(x) = ?$. We study the relationship between the UAV and RUAV models, and then prove that the class of DNF formulas is learnable under the RUAV model using RUAV-MQ and RUAV-EQ oracles (no evaluation oracle is needed).

2 OUR LEARNING MODEL

We first briefly review the exact learning model introduced by Angluin [3]. The learner's goal is to exactly learn (using various types of queries) how an unknown (boolean) target function f, taken from some known concept class $\mathcal{C}$, classifies (from $\{+,-\}$) all instances from the domain. Many different types of queries have been studied. We now define the most common queries.

Membership Query The learner supplies an example $x \in \{0,1\}^n$ and is told $f(x)$.

Equivalence Query the learner presents a candidate function h and either is told that $h \equiv f$ (in which case learning is complete), or else is given a *counterexample* $x \in \{+,-\}$ for which $h(x) \neq f(x)$.

Subset Query The learner presents a candidate function h and is either told that for all $x \in \{0,1\}^n$, if $h(x) =$ "+" then $f(x) =$ "+" or otherwise is given an example $x \in \{0,1\}^n$ for which $h(x) =$ "+" and $f(x) =$ "−".

Superset Query The learner presents a candidate function h and is either told that for all $x \in \{0,1\}^n$, if $h(x) =$ "−" then $f(x) =$ "−" or otherwise is given an example $x \in \{0,1\}^n$ for which $h(x) =$ "−" and $f(x) =$ "+".

As suggested by Valiant [43], we consider when the target concept is defined over n boolean variables $x_1, \ldots, x_n$ with an instance space $\mathcal{X}_n = \{0,1,*\}^n$, where "$*$" indicates that the corresponding variable is *unspecified*. An example $x \in \{0,1\}^n$ is called *total*. We say that a total example y is a *completion* of example $x \in \{0,1,*\}^n$ if each specified attribute in x has the same value in y. Let f be a boolean function defined over $x_1, \ldots, x_n$, and let $x \in \{0,1,*\}^n$. We define $f(x) = +$ if and only if $f(y)$ is positive for all vectors y that are completions of x. Similarly, we define $f(x) = -$ if and only if $f(y)$ is negative for all vectors y that are completions of x. Otherwise, we say that $f(x) = ?$ (for unknown). Note that we use x_i to denote both the ith boolean variable and the ith bit of the example x. Thus for an example $x \in \{0,1,*\}^n$, x_i gives the value for variable x_i. We will always use $x_1, \ldots, x_n$ as the variables, so for an example $y \in \{0,1,*\}^n$, y_i gives the value for variable x_i. We denote the example obtained from x by setting all the unspecified bits to $b \in \{0,1\}$ by $x_{*\to b}$ and the example obtained by flipping the ith bit of x, x_i, to $b \in \{0,1,*\}$ by $x_{x_i \to b}$.

For target concept f and $x \in \{0,1,*\}^n$, UAV-MQ(x) returns the value of $f(x)$ from $\{+,-,?\}$. In response to UAV-EQ(h), the learner either is told that $h \equiv f$ or else is given a counterexample x (along with its classification) for which $h(x) \not\equiv f(x)$. Note that the target function is a ternary (versus the standard binary) function and all three values are distinct. So, for example, if $h(x) = +$ and $f(x) = ?$ then x could serve as a counterexample for h. When given a boolean formula h and a total example $x \in \{0,1\}^n$, in polynomial time one can evaluate $h(x)$. However, when given a boolean formula h from hypothesis class $\mathcal{H}$ and an example $x \in \{0,1,*\}^n$, the problem of evaluating $h(x)$ is NP-complete for many choices for $\mathcal{H}$. Thus we introduce an *evaluation oracle*, where on input h and $x \in \{0,1,*\}^n$, EV(h,x) outputs the value of $h(x)$. There are some important cases when the EV oracle is not needed since a polynomial time algorithm can replace it. First, for a projection closed class[1] if we can efficiently determine whether two different representations of a concept class are functionally equivalent then an EV oracle is not needed. Such equivalence tests exist for monotone DNF, read-once branching programs [36], Horn-sentences, free-branching programs [14, 21], read-once boolean formulas [28] and read-twice DNF [34]. Also, if the number of unspecified attributes is $O(\log n)$, then evaluation can be done in polynomial time by simply considering all completions of x. Likewise, when h is a unate formula, then it is easily seen that the evaluation (for *any* example from $\{0,1,*\}^n$) can be done in polynomial time.

[1] A concept class C is *projection closed* if for any $f \in C$, the function obtained by fixing a variable in f to 0 or 1 is $f' \in C$ for $|f'| \leq |f|$.

Finally, for the class of decision trees itself we can efficiently implement the EV oracle by computing the classification of all nodes in the order given by a post-order traversal of the decision tree.

Notice that for a target function f, the MQ oracle on input x returns the value of $\mathrm{EV}(f, x)$. For applications such as the medical diagnosis example, one view is that an expert serves as the UAV-MQ oracle and thus the computational issues are not a problem. However, when studying the learnability of boolean formulas under the UAV model, it is desirable that the UAV-MQ oracle runs in polynomial time. Thus we introduce variants of the UAV membership and equivalence oracles. The UAV-$\mathrm{EQ}_{\log}$ oracle takes as input a hypothesis h and returns a counterexample in which there are at most $O(\log n)$ unspecified attributes. Similarly, the UAV-$\mathrm{MQ}_{\log}$ oracle takes as input any example x with $O(\log n)$ unspecified attributes and outputs the classification (+,−, or ?) of x.

As in the standard exact learning model, the learner's goal is to exactly identify the target concept using membership and equivalence queries (i.e. for all $x \in \{0, 1, *\}^n$, $h(x) = f(x)$). Since there are unspecified attribute values, we call this model the *UAV Exact Learning* model. We also study variations in which only a UAV-EQ oracle or only a UAV-MQ oracle is allowed. In some of our results, the unspecified attributes of the examples given to the UAV-MQ oracle are a subset of the attributes that were unspecified in the recent counterexample that came from the UAV-EQ oracle. In these cases, as long as we use the UAV-$\mathrm{EQ}_{\log}$ oracle (and thus have $O(\log n)$ unspecified attributes), then we can replace the UAV-MQ oracle by a UAV-$\mathrm{MQ}_{\log}$ oracle, and hence using Observation 2, by a standard MQ oracle. Thus for these cases, we obtain positive results in the UAV model by just using a UAV-$\mathrm{EQ}_{\log}$ oracle and an MQ oracle. Finally, a corresponding model of *UAV PAC Learning* can be naturally defined.

3 RELATED WORK

Within the exact learning model a number of interesting polynomial time algorithms have been presented to learn target classes such as decision trees [19], deterministic finite automata [2], Horn sentences [4], read-once formulas [5, 18, 16], read-twice DNF formulas [1], k-term DNF formulas [13, 20], etc. For all of these classes it is known (using information-theoretic arguments), that neither membership queries nor equivalence queries alone suffice.

Although different from the goals of our work, there has been work on learning when the examples may be mislabeled [3, 32, 41, 29] and when there is attribute noise [40, 23, 33]. There has also been some work in which the answers to membership queries are noisy or missing [37, 7, 42, 6, 12]. Although we can get a "?" response from our membership oracle, this response is not adversarially generated to model an inconclusive outcome, but rather is structurally defined by the target concept and indicates that there is insufficient information to determine the classification of the provided example.

A learning model that has very similar motivations to the UAV model is

the model of RFA (restricted focus of attention) learnability [9, 10]. The k-RFA model is a variant of the PAC model in which for each example only k attributes (of the n attributes), as selected by the learner, are specified. Thus, unlike the UAV model in which the unspecified attributes are adversarially selected, in the k-RFA model the leaner can select which attributes are specified. Another key difference is in the criteria required of the learner. In the k-RFA model the requirements for successful learning are exactly as in the PAC model. Namely, the learner is given a random example $x \in \{0,1\}^n$ and asked to properly classify x with high probability. To explore the differences between these two models, we consider the task of medical diagnosis which has been suggested as an example application for both models. In the k-RFA model the view is that for the training data, the learner can select which test to run and hence which data is available. However, since the training data is likely to be medical records from past patients, any data not available (even if a test could have been run) cannot be obtained at a later date. Also, there are some attributes such as those related to family history that may just not be available. The UAV model captures this by having an adversary choose which attributes are not specified. Another key difference between the two models is that in the RFA model the examples to be classified have all attributes specified. However, in medical diagnosis (like many other scenarios), some data for the patient being diagnosed may just not be available. For example, some information about family history may be unavailable, and maybe due to the patient's condition some test(s) could not be run. Thus we feel it is important for algorithms in our model to classify examples that have unspecified attributes. This requirement led us to have a three-valued (versus binary) output.

In the different "learning to reason" framework, Khardon and Roth [31] investigate the construction of a knowledge base for representing "the world", i.e. some boolean function f. This knowledge base is then used to deduce (i.e. reason) if f logically implies α where α is a propositional query capturing the situation at hand. They consider various interpretations of applying a boolean function f to a partial instance x:

(1) Universal: $f(x) = +$ if all completions of x are classified as "+" by f, otherwise $f(x) = -$.

(2) Existential: $f(x) = +$ if there exists a completion x' of x such that $f(x') = +$, otherwise $f(x) = -$.

(3) Abbreviated: f classifies x by treating the "*"s as 0s and then classifying the total instance obtained.

Notice that the universal and existential interpretations are the same as treating the classification "?" in the UAV model as "−" and "+", respectively. The membership and equivalence query oracles under these two interpretations are clearly less informative than their UAV counterparts. The universal interpretation was introduced earlier when Valiant [43] first defined the PAC model.

Clearly, if the target concept is monotone, then both the universal and abbreviated interpretations are equivalent. Using this observation, Valiant constructs an efficient PAC learning algorithm, with the aid of a membership query oracle, for learning monotone DNF under the universal interpretation.

Some work that has similar motivations is the p-concepts model of Kearns and Schapire [30]. In the p-concepts model (when applied to the boolean domain), the learner is given a total example from $\{0,1\}^n$, yet there is some probabilistic process (or possibly something that appears probabilistic due to the learner being unaware of some important attributes) that determines whether the output is positive or negative. In our setting we assume that the learner knows which attributes it would like to gather, but cannot obtain values for all such attributes. Other key distinctions are that our target concept is ternary versus binary and deterministic versus probabilistic, i.e. given an example x, $f(x)$ always has the same value drawn from $\{+,-,?\}$ in the UAV model, whereas in the p-concepts model x can sometimes have a classification of "+" and sometimes have a classification of "−".

In other related work, Frazier, Goldman, Mishra, and Pitt [22] introduce a learning model that captures the idea that teachers may have gaps in their knowledge. They consider learning from a teacher who labels each example as "+", "−", or "?" in such a way that knowledge of the concept class and all the positive and negative examples are not sufficient to determine the labeling of any example labeled with "?". Their goal is to PAC-learn the ternary labeling presented by such a *consistently ignorant teacher*. Similar to their model, we are learning a ternary function. However, in their model all attribute values are known (i.e. the examples come from $\{0,1\}^n$). It is easily shown that under the domain $\{0,1,*\}^n$, the "?"s allowed by our model are placed in such a way to meet the definition of the teacher being consistently ignorant. Namely, no "?"'s value could be determined from the positive examples, negative examples, and knowledge of the concept class. Another key difference between the two models is that instead of having the "?"s placed adversarially (as in the consistently ignorant teacher model), in our UAV model the "?"s are defined based on the target concept[2]. Thus in the UAV model, the complexity of the ternary target function has the same complexity as that of the defining boolean function.

There has been some empirical work studying the task of learning from incomplete data [8, 35, 15]. With the goal of giving a theoretical explanation for the observed empirical phenomena, Schuurmans and Greiner [38, 39] studied the problem of learning accurate default concepts to be used when working with incomplete data. Given an example with unspecified attribute values, they call an example $x \in \{0,1,*\}^n$ *ambiguous* if the classification cannot be determined (i.e. using our terminology the classification of x is unknown). Their goal is to find a good way to compute a default classifier (which can have nonmonotonic behavior) for making a prediction of + or − for the ambiguous examples. In other words, unlike our UAV model, there are no "?" responses for either the target concept or the hypotheses. Also, instead of an adversary choosing which

[2]Although the "*"s in the examples are adversarially placed.

attributes are unspecified, in their model a total example is drawn from some fixed distribution and then some attribute values become unspecified according to some probability distribution. They investigate, under various conditions, whether the strategies [8, 35, 15] that are used in practice converge to some optimum hypothesis in the limit and the sample complexities required to achieve a certain PAC-like learning criterion.

Finally, there have been two oracles studied that are similar to the UAV-MQ oracle: the constrained instance oracle [26] and the projective equivalence oracle [27]. We use CIQ to denote a constrained instance query, and PEQ to denote a projective equivalence query. Let a (which can be viewed as partial assignment or an example in the UAV model) be drawn from $\{0, 1, *\}^n$, and for the target boolean formula f, let f_a be the projection defined by a. That is, f_a is the function defined by f when the variables assigned in a are replaced by a constant. Note that

$$\text{UAV-MQ}(a) = \begin{cases} + & \text{if } f_a \text{ is a tautology} \\ - & \text{if } f_a \text{ is a contradiction} \\ ? & \text{otherwise} \end{cases} .$$

The constrained instance query is defined as follows where $\ell \in \{-, +\}$: $\text{CIQ}(a, \ell)$ returns "yes" if and only if there is an assignment x that is a completion of a for which $f(x) = \ell$. Observe that $\text{UAV-MQ}(a) = +$ if and only if $\text{CIQ}(a, -) =$ "no" and $\text{CIQ}(a, +) =$ "yes." Similarly, $\text{UAV-MQ}(a) = -$ if and only if $\text{CIQ}(a, -) =$ "yes" and $\text{CIQ}(a, +) =$ "no." Finally, $\text{UAV-MQ}(a) = ?$ if and only if $\text{CIQ}(a, -) =$ "yes" and $\text{CIQ}(a, +) =$ "yes." Thus these two queries are essentially equivalent. The projective equivalence query is defined as follows. For a partial assignment a and a boolean formula f', $\text{PEQ}(a, f')$ replies "yes" if and only if $f_a \equiv f'$. Similar to the above, it is easily shown that the special case of a projective equivalence oracle in which f' is either the constant *true* or *false* is essentially equivalent to the UAV membership oracle.

It is known that read-once formulas are exactly learnable using only constrained instance queries [26] or using these restricted projective equivalence queries [27]. Thus read-once formulas are exactly learnable in the UAV model using *only* the UAV-MQ oracle. Furthermore, it is known [5] that a polynomial number of calls to the MQ oracle is not sufficient. This demonstrates (as one would expect) that the UAV-MQ oracle is more powerful than the MQ oracle.

Bshouty, Cleve, Kannan and Tamon [17] show that DNF formulas can be learned by a randomized algorithm in expected polynomial time with equivalence queries and the aid of an NP oracle. Using this result, they also show that DNF formulas can be learned using subset and superset queries. The hypothesis class of their algorithms is the class of depth-3 $\wedge$-$\vee$-$\wedge$ formulas. An obvious question to ask is whether the UAV-MQ oracle is powerful enough to simulate the NP oracle and EQ oracle (or the superset and subset oracles) needed by the algorithm of Bshouty et. al. If so, then it would immediately follow that the class of DNF formulas is learnable by a randomized algorithm in expected polynomial time using a UAV-MQ oracle alone. However, achieving either of these two simulation results does not seem possible. It is easily shown that a a

UAV-MQ oracle can simulate subset and superset query oracles where the hypotheses are conjunctions and disjunctions of literals respectively. For example, if a subset query is made with the hypothesis $\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_k$ for literals ℓ_i then this can be simulated with the UAV-MQ oracle by setting each of the k literals used in the hypothesis to be true and setting the remaining variables to "$*$". If the UAV-MQ oracles returns "+" then the answer to the subset query is "yes". If the UAV-MQ oracle returns "-" then return $x_{*\to 0}$ as a negative counterexample. Finally, if the UAV-MQ oracle responds with "?" then a **FormNegEx** (a variation of **FormPosEx** from Figure 1 with "+" and "-" exchanged) can be used to construct a counterexample. However, we believe that the UAV-MQ oracle cannot simulate subset and superset query oracles with the hypothesis class of depth-3 $\wedge$-$\vee$-$\wedge$ formulas as would be needed to use the algorithm of Bshouty et. al to learn DNF formulas with only a UAV-MQ oracle.

Recently, Birkendorf, Klasner, Kuhlman and Simon [11] investigated the UAV model further and answered a number of open problems posted in an earlier (conference) version of this paper [24]. They presented lower bound results on the number of UAV-EQs and UAV-MQs required to learn a concept class in terms of its Vapnik Chervonenkis dimension. Further, they extended Angluin's [3] sunflower lemma (which is useful in proving lower bound results in the exact model) to the UAV setting. In doing so, they establish exponentially large lower bounds on the number of UAV-MQs needed to learn monotone DNF, read-once DNF, $O(n)$-DNF, read-twice DNF and 2-decision lists. In contrast to these negative results, they also present efficient algorithms using UAV-MQs for learning read-once DNF, constant-term DNF and 1-decision lists. They also determine how restrictions on the number of unspecified attributes affects the strength of the UAV oracles. They show that a UAV-EQ oracle that returns counterexamples with at most $r-1$ unspecified attributes is strictly stronger than one that is allowed to return counterexamples with up to r unspecified attributes. They also establish a similar result for the UAV-MQ oracle. Besides these hierarchical results, they also compare the power among the standard and UAV oracles.

4 RELATIONSHIPS WITH THE STANDARD EXACT LEARNING MODEL

We begin with an observation that directly follows from the fact that within the UAV model the examples given as input to the UAV-MQ oracle can be selected from $\{0,1,*\}^n \supset \{0,1\}^n$.

Observation 1 *Let C be a concept class that is exactly learnable using the* MQ *oracle alone. Then C is UAV exactly learnable using only the* UAV-MQ *oracle.*

Next we consider when a UAV-$\text{MQ}_{\log}$ oracle can be simulated with a standard membership query oracle. Let x be an example given to the UAV-$\text{MQ}_{\log}$ oracle (so x has at most $c \log_2 n$ unspecified attributes for some constant c).

Since there are at most n^c completions of x, the value of UAV-$\text{MQ}_{\log}(x)$ can be computed from making n^c calls to the MQ oracle and using polynomial time. Thus we get the following observation.

Observation 2 *The* UAV-$\text{MQ}_{\log}$ *oracle can be simulated in polynomial time using the standard* MQ *oracle.*

While standard MQ oracle can be trivially simulated in our model, equivalence queries are more complicated since the UAV-EQ oracle returns an example $x \in \{0,1,*\}^n$ versus from $\{0,1\}^n$. We show the somewhat surprising result that the standard EQ oracle can be simulated by a UAV-$\text{EQ}_{\log}$ oracle and a *standard* MQ oracle.

Theorem 3 *Let $\mathcal{C}$ be a concept class that is exactly learnable in polynomial time using the* MQ *and* EQ *oracles. Then $\mathcal{C}$ is UAV exactly learnable in polynomial time using the* MQ *and* UAV-$\text{EQ}_{\log}$ *oracles.*

Proof: Let $\mathcal{A}$ be the exact learning algorithm for $\mathcal{C}$, and let $f \in \mathcal{C}$ be the target function. We construct an algorithm $\mathcal{A}'$ to learn $\mathcal{C}$ in the UAV model. Suppose that $\mathcal{A}$ makes an equivalence query with hypothesis h. Then $\mathcal{A}'$ uses its UAV-$\text{EQ}_{\log}$ oracle in the following manner. If UAV-$\text{EQ}_{\log}(h)$ responds "yes" then $\mathcal{A}'$ can output h and halt. Suppose instead that counterexample $x \in \{0,1,*\}^n$ is returned. In each of the following three cases we show how to generate an example $y \in \{0,1\}^n$ such that $h(y) \neq f(y)$. Then y is given to $\mathcal{A}$ as a counterexample. Note that in all of the cases, the unspecified attributes in any example given to the UAV-MQ oracle are a subset of those unspecified in the counterexample x. Since x had $O(\log n)$ unspecified attributes, all examples given to the membership oracle have $O(\log n)$ unspecified attributes. Thus by Observation 2 we can, in polynomial time, simulate the calls to the UAV-MQ oracle with an MQ oracle.

Case 1: $h(x) = -$. One possibility is that $f(x) = +$. Let $y = x_{*\to 0}$. By the definition of a positive example in the UAV model, we know that $f(y) = +$. By the definition of a negative example in the UAV model, we know that $h(y) = -$. Thus $y \in \{0,1\}^n$ is a counterexample for h.

The other possibility is that $f(x) = ?$. By definition, we know there is some completion y of x for which $f(y) = +$. We use membership queries to find such a y. The basic idea is to go through the unspecified attributes setting the next attribute to 1 (or to 0) and checking with the UAV-MQ oracle if this example is positive. If not, since $f(x) = ?$ at least one of these examples has a "?" classification, and thus we can continue with this new example (which has one less unspecified attribute). Figure 1 describes our procedure **FormPosEx** that receives an $x \in \{0,1,*\}^n$ for which $h(x) = -$ and $f(x) = ?$, and returns a total example $y \in \{0,1\}^n$ for which $h(y) = -$ and $f(y) = +$.

```
FormPosEx(x)

1    Let x_i be the first unspecified attribute in x
2    v := x_{x_i→0}
3    w := x_{x_i→1}
4    if UAV-MQ(v) = ? then
5        FormPosEx(v)
6    else if UAV-MQ(v) = + then
7        return v_{*→0} to A
8    else (so UAV-MQ(v) = −)
9        if UAV-MQ(w) = + then
10           return w_{*→0} to A
11       else (so UAV-MQ(w) = ?)
12           FormPosEx(w)
```

Figure 1: The algorithm **FormPosEx** which takes as input an example $x \in \{0,1,*\}^n$ for which $f(x) = ?$ and $h(x) = -$, and returns a total example $y \in \{0,1\}^n$ for which $h(y) = -$ and $f(y) = +$.

Case 2: $h(x) = +$. This is handled using a symmetric argument to Case 1. Notice that **FormPosEx** (and its proof of correctness given below) can be easily modified to create the procedure **FormNegEx**.

Case 3: $h(x) = ?$. One possibility is that $f(x) = +$. Here we can apply a variation of **FormNegEx** in which we use the evaluation oracle, $\mathrm{EV}(h, x)$, instead of UAV-MQ(x). (Recall that since there are $O(\log n)$ unspecified attributes in x, the computation of $\mathrm{EV}(h, x)$ can be done in polynomial time.) The other possibility is that $f(x) = -$ in which we apply a variation of **FormPosEx** where we again use $\mathrm{EV}(h, x)$ instead of UAV-MQ(x).

We now prove that **FormPosEx** is correct. Let x be the example given as input to **FormPosEx** (for which we maintain the invariant that $f(x) = ?$). Let x_i be the first unspecified attribute in x, let $v = x_{x_i \to 0}$, and let $w = x_{x_i \to 1}$. Notice that if **FormPosEx** ever reaches the last unspecified attribute, then clearly $f(v)$ and $f(w)$ must be "+" or "−" since $v, w \in \{0,1\}^n$. Furthermore, since UAV-MQ$(x) = ?$ then either $f(v) = +$ or $f(w) = +$. Thus eventually **FormPosEx** returns an example $y \in \{0,1\}^n$ to $\mathcal{A}$. We now argue that y is a counterexample. First, since $h(x) = -$ and y is a completion of x, $h(y) = -$. Finally, it follows directly from **FormPosEx** that UAV-MQ$(y) = f(y) = +$. The last thing we must argue is that when we reach the final else clause (so $f(v) = -$), then $f(w) \neq -$. That is, if $f(w) \neq +$, then $f(w) = ?$. This is crucial, since it is important that the example for the recursive call to **FormPosEx** has a classification of "?".

We now prove the following slightly more general result. For $x \in \{0,1,*\}^n$ for which $f(x) = ?$ and for which x_i is unspecified, as we did in Figure 1 define $v = x_{x_i \to 0}$ and $w = x_{x_i \to 1}$. If $f(v) \neq ?$, we show that $f(v) \neq f(w)$. For the sake of contradiction, assume that $f(w) = f(v) = -$. Consider the total vectors $w' = w_{* \to 0}$ and $v' = v_{* \to 0}$. Note that $f(w') = f(v') = -$. Also, the only difference between w' and v' is that for the ith attribute $w'_i = 1$ and $v'_i = 0$, yet we are given that $f(x) = ?$. Thus there must be some completion z of x for which $f(z) = +$. Suppose that the ith attribute $z_i = 0$. Then since $f(v) = -$, it follows that $f(z) = -$. Likewise, suppose that the ith attribute $z_i = 1$. Then since $f(w) = -$, it follows that $f(z) = -$. This contradicts the existence of such a z, completing the proof that if $f(v) \neq ?$, then $f(v) \neq f(w)$.

Thus the algorithm **FormPosEx** will, after at most $2n$ calls to the UAV-MQ oracle (simulated by a polynomial number of calls to the MQ oracle) and polynomial time, return a counterexample y to $\mathcal{A}$ as desired. Thus both the membership and equivalence queries requested by $\mathcal{A}$ can be simulated by $\mathcal{A}'$ in polynomial time. □

As a corollary to this theorem we consider when the UAV-EQ oracle (versus the UAV-$EQ_{\log}$ oracle) is used. The only change required is that now the UAV-MQ oracle cannot necessarily be efficiently simulated by the MQ oracle). Also, now the EV oracle is needed since we may not be able to efficiently evaluate the learner's hypothesis on the examples from $\{0,1,*\}^n$.

Corollary 4 *Let C be a concept class that is exactly learnable in polynomial time using the* MQ *and* EQ *oracles. Then C is UAV exactly learnable in polynomial time using the* UAV-MQ, UAV-EQ, *and* EV *oracles.*

While in general our result requires the use of the EV oracle, it is easily seen that a polynomial time algorithm can simulate the EV oracle for a unate class. More specifically, consider the query $\mathrm{EV}(f,x)$ where $x \in \{0,1,*\}^n$. Let x_1 be the completion of x obtained by setting the unnegated attributes to 1 and the negated attributes to 0. Likewise, let x_2 be the completion of x obtained by setting the negated attributes to 0 and the unnegated attributes to 1. If $f(x_1) = f(x_2)$ then output $f(x_1)$ as the classification for $f(x)$. Otherwise, $f(x) = ?$. Thus only two examples from $\{0,1\}^n$ need to be classified. So for any learning algorithm that uses a unate class for the hypothesis class, the EV oracle is not needed.

Corollary 5 *Let C be exactly learnable in polynomial time with a unate hypothesis class using the* EQ *and* MQ *oracles. Then C is UAV exactly learnable in polynomial time using the* UAV-EQ *and* UAV-MQ *oracles.*

As discussed earlier, for the class of decision trees we can efficiently implement the EV oracle by computing the classification of all nodes in the order given by a post-order traversal of the decision tree. Thus, if the hypothesis class is the class of decision trees, the EV oracle is not needed.

Corollary 6 *Let C be exactly learnable in polynomial time with a hypothesis class of decision trees using the* EQ *and* MQ *oracles. Then C is UAV exactly learnable in polynomial time using the* UAV-EQ *and* UAV-MQ *oracles.*

Finally, for a projection closed class for whichwe can efficiently determine whether two different representations of a concept class are functionally equivalent then an EV oracle is not needed. Such equivalence tests exist for monotone DNF, read-once branching programs [36], Horn-sentences, free-branching programs [14, 21], read-once boolean formulas [28] and read-twice DNF [34].

Corollary 7 *Let C be exactly learnable in polynomial time using the* EQ *and* MQ *oracles with a hypothesis class $\mathcal{H}$ where $\mathcal{H}$ is projection closed and for which there is a polynomial time algorithm to determine if $h_1 \in \mathcal{H}$ and $h_2 \in \mathcal{H}$ are logically equivalent. Then C is UAV exactly learnable in polynomial time using the* UAV-EQ *and* UAV-MQ *oracles.*

Notice that the simulation used in Theorem 3 requires the use of membership queries by $\mathcal{A}'$ even when $\mathcal{A}$ uses only equivalence queries. One natural question is: When can a standard exact learning algorithm that uses only equivalence queries be simulated by a UAV algorithm that uses only equivalence queries? We now partly answer that question by arguing that **FormPosEx** (respectively, **FormNegEx**) need not use membership queries to simulate the equivalence queries when the concept class is monotone. In particular, given an example x for which $f(x) = ?$ (or $h(x) = ?$), **FormPosEx** (respectively, **FormNegEx**) and the variations used in Case 3 need just return $x_{*\to 1}$ (respectively, $x_{*\to 0}$).

Corollary 8 *Let C be exactly learnable in polynomial time with a monotone hypothesis class using only the* EQ *oracle. Then C is UAV exactly learnable in polynomial time using only the* UAV-EQ *oracle.*

All of the above results address the issue of converting an algorithm that works in the standard model to one that works in the UAV model. Since the example x returned from a standard equivalence query is a total example for which $f(x) \neq h(x)$, x can also serve as a response for the UAV equivalence query. Thus we get the following result.

Observation 9 *If concept class C is exactly learnable in polynomial time in the UAV model using only the* UAV-EQ *oracle, then C is exactly learnable in the standard model in polynomial time using only the* EQ *oracle.*

5 LEARNING ORDERED DECISION TREES WITH A UAV-MQ ORACLE

Combining our results from the last section with Bshouty's decision tree algorithm [19], we can learn the class of decision trees in the UAV model using the UAV-MQ, UAV-EQ, and EV oracles. As discussed in Section 2, for the class of

Tree **BuildTree**($proj$)

1 **if** UAV-MQ($proj * \cdots *$) $\neq$? **then return** tree(UAV-MQ($proj * \cdots *$), nil, nil)
2 $T_L :=$ tree(UAV-MQ($proj 0 \cdots 0$), nil, nil)
3 $i := i' - 1$
4 **while** $i >$ $|proj|$
5 **if IsParent**($proj, i, T_L$) $= true$
6 $T_L :=$ tree(x_i, T_L, **BuildTree**($proj \underbrace{0 \cdots 0}_{i-|proj|-1} 1$))
7 $i := i - 1$
8 **return** T_L

Figure 2: The algorithm **BuildTree** for learning an arbitrary boolean function f in its canonical ordered decision tree representation T_f. The input $proj$ gives a partial assignment for the first $|proj|$ attributes. In the initial call $proj$ is empty. Note that the projection sent recursively in Line 6 uses $proj$ to assign $x_1, \ldots, x_{|proj|}$, assigns 0 to $x_{|proj|+1}, \ldots, x_{i-1}$, and assigns $x_i = 1$.

decision trees we can implement the EV oracle efficiently. However, since the hypothesis class in Bshouty's algorithm is not a decision tree, the EV oracle must still be provided.

In this section we consider a class $\mathcal{T}$ of "ordered" decision trees defined with respect to a linear ordering of the variables, say $x_1 < \cdots < x_n$, such that the variable in each internal node is greater than the variables in its ancestors in this ordering. We give an algorithm to learn $\mathcal{T}$ in the UAV model using only the UAV-MQ oracle. Since our hypothesis class is also an ordered decision tree, we do not need an EV oracle because we can efficiently evaluate $f(x)$ for a decision tree T_f and example $x \in \{0, 1, *\}^n$. Since the class of singletons can be represented by trees in $\mathcal{T}$ with $O(n)$ nodes and the class of singletons cannot be learned with standard membership queries [3], $\mathcal{T}$ cannot be learned using standard membership queries. This provides a second example (along with read-once formulas, see Section 3) demonstrating that the UAV-MQ oracle is more powerful than standard MQ oracle.

We denote the size of a tree T by $|T|$. We also adopt the convention that we branch to the left subtree if the variable associated with the current node is "0". Otherwise we branch to the right. Notice that any boolean concept has a representation in $\mathcal{T}$ although the size of the representation may be exponential in the size of its DNF representation. We now prove that the representation of any boolean function f using $\mathcal{T}$ with the smallest number of nodes is unique.

Lemma 10 *The representation of any boolean function f using $\mathcal{T}$ with the smallest number of nodes is unique.*

Proof: The proof is by induction on the number of variables n. Clearly, it is

true when $n = 1$. For the sake of contradiction, suppose f has two minimum representations T_1 and T_2. If the variables at the roots of both trees are the same, then by our inductive hypothesis, both left and right subtrees of both trees are identical. Without loss of generality, suppose variable x_i at the root of T_1 is smaller than that of T_2. Then x_i must not be relevant to f meaning that T_1 could be replaced by either of its subtrees, contradicting the minimality assumption for T_1. □

We now prove the main result of this section.

Theorem 11 *The class of ordered decision trees is efficiently exactly learnable in the UAV model using only the* UAV-MQ *oracle. Specifically, for an arbitrary boolean function f, let T_f be its canonical (minimal) representation as an ordered decision tree. Then T_f can be exactly identified using $O(n^2|T_f|)$ time and calls to the* UAV-MQ *oracle.*

Proof: We present an algorithm **BuildTree** (see Figure 2) for learning T_f. The basic idea is to first determine the subtree rooted at a node along the path from the root to the leftmost leaf, then determine the label of its parent, and recursively determine its right sibling. We can then iteratively continue this process moving up the tree. More specifically, **BuildTree** takes as input a partial assignment $proj$ and builds the canonical ordered decision tree for f_{proj} (the projection of f defined by $proj$). The assignment $proj$ assigns a 0 or 1 for attributes $x_1, \ldots, x_j$ for some j with the remaining attributes left unspecified. (Initially, all attributes are unspecified.) We denote $proj$ by a length j bit string (i.e. $proj \in \{0,1\}^j$) where only the specified attributes are included. We let $projb\cdots b$ denote the example obtained by padding $proj$ by repeating $b \in \{0, 1, *\}$ $n - j$ times.

The procedure **BuildTree** is called recursively to construct the canonical ordered decision tree for f_{proj}. The terminating condition (see Line 1) is reached when UAV-MQ$(proj * \cdots *) \neq ?$, in which case we return a leaf with label $f(proj * \cdots *) =$ UAV-MQ$(proj * \cdots *)$. The notation $\mathbf{tree}(x_i, T_L, T_R)$ denotes the tree where the root is labeled x_i, the left subtree is T_L, and the right subtree is T_R. A leaf with label $\ell \in \{+, -\}$ is denoted as $\mathbf{tree}(\ell, nil, nil)$. If the terminating condition does not hold, then the root of the tree is an internal node. Define P_L as the path from the tree's root to its leftmost leaf. Clearly, the subtree rooted at the last node of P_L is simply a leaf with label $f(proj0\cdots0)$. In Line 2 we construct such a leaf (as a single-node subtree) and call it T_L. Next we determine the attribute x_i associated with the parent of T_L (see Figure 3). Let $x_{i'}$ be the attribute associated with the root of T_L. Observe that the index i for the attribute x_i must be between 1 and $i' - 1$. We begin by setting $i = i' - 1$ and testing (in Line 5) whether x_i is the attribute for the parent of T_L by calling the function $\mathbf{IsParent}(proj, i, T_L)$. If we find x_i is not the parent, then we decrement i and repeat the test. If the test succeeds for i, we then construct (in Line 6) the subtree rooted at the parent which consists of a root labeled x_i, T_L as its left subtree, and the right subtree T_R that we recursively learn by calling **BuildTree** with the projection $proj' = proj\underbrace{0\cdots0}_{i-j-1}1$. (So $proj'$

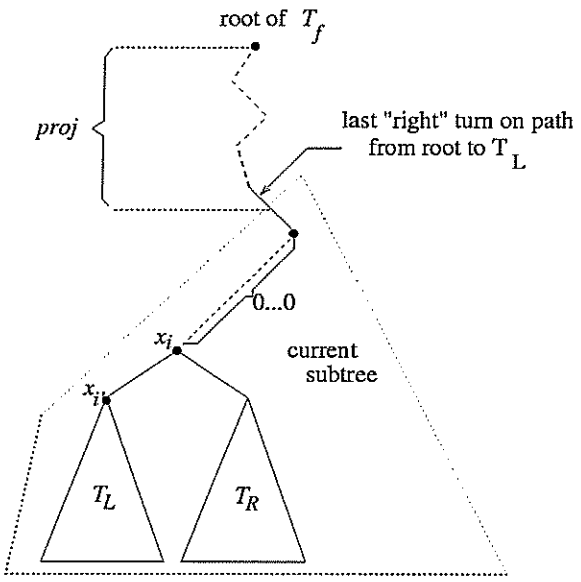

Figure 3: This figure shows the configuration we have during the main step of **BuildTree** when T_L (and $|proj|$) are known and it must determine x_i and then recursively construct T_R.

assigns $x_1, \ldots, x_j$ according to $proj$, assigns 0 to the attributes $x_{j+1}, \ldots, x_{i-1}$ and finally assigns 1 for x_i.) We then let this newly constructed tree be our next T_L and iteratively repeat the process of finding its parent and a new T_L until we have constructed the entire tree.

We now describe how **IsParent**$(proj, i, T_L)$ (see Figure 4) tests whether x_i is the parent of T_L in the subtree $T_{J_{proj}}$. Let i' be the index for the attribute at the root of T_L (see Figure 3). Notice that the way **BuildTree** calls **IsParent** ensures that the attributes between x_i and $x_{i'}$ have failed the test (and thus cannot appear as the label in the parent of T_L). For leaf ℓ of T_L, let a_ℓ be the partial assignment for $x_{i'}, \ldots, x_n$ which defines the path to ℓ from the root of T_L. First consider the case where x_i is not the parent of T_L. Then changing x_i from "$*$" to "1" does not change its classification. Thus the partial assignment $proj \underbrace{0 \cdots 0}_{i-j-1} 1 \underbrace{* \cdots *}_{i'-i} a_\ell$ must be classified consistently with the label of ℓ. Conversely, if x_i is the parent of T_L then T_L is different from T_R. That is there is a leaf ℓ in T_L such that the classification of the partial assignment $proj \underbrace{0 \cdots 0}_{i-j-1} 1 \underbrace{* \cdots *}_{i'-i} a_\ell$ is inconsistent with the labeling of ℓ.

We now use induction on n to prove the stated upperbound for the number of calls to the UAV-MQ oracle (and hence the time complexity). Clearly, the bound holds for $n = 1$. For $n > 1$, the subroutine **IsParent** makes at most

$|T_f|$ queries and is called by **BuildTree** at most n times. The total number of queries needed in Line 6 to recursively construct the right subtrees of the nodes along the path P_L is at most $(n-1)^2$ times the total size of these right subtrees, which is bounded by $|T_f|$. Together with the two initial queries at Lines 1 and 2, the number of queries made is $2 + O(n|T_f|) + O((n-1)^2|T_f|) = O(n^2|T_f|)$ as desired.

□

6 LEARNING DNF FORMULAS UNDER THE RUAV MODEL

For many concept classes, the problem of determining the classification of an example $x \in \{0,1,*\}^n$ for a *known* target function f is NP-complete since it involves solving a satisfiability and a non-satisfiability problem. As we have noted, for unate formulas and decision trees this evaluation problem can be efficiently solved. On the other hand, for the concept class of DNF formulas, SAT can trivially be reduced to our evaluation problem and thus the evaluation problem is NP-complete. For the concept class of DNF formulas, we introduce a representation-dependent variation of the UAV model in which the learner and teacher can efficiently classify any example from $\{0,1,*\}^n$. Let f be the target DNF formula. For a moment suppose a computationally unbounded teacher builds a DNF formula $f' \equiv f$ that has one term for each (of the possibly exponentially many) prime implicants[3] of f. The teacher could now compute $f(x)$ for $x \in \{0,1,*\}^n$ as follows. If any term in f' is satisfied by the known attributes in x, then $f(x) = +$. If every term in f' is known *not* to be satisfied by the known attributes in x, then $f(x) = -$. Otherwise $f(x) = ?$.

Using the above observation as a motivation, we now define the following representation-dependent version of the UAV model (which we call the RUAV model) in which a computationally bounded agent can evaluate the target formula f on example $x \in \{0,1,*\}^n$. The idea is to approximate the above procedure directly using f. Namely, in the RUAV model, if any term in f is satisfied by the known attributes in x, then $f(x) = +$. If every term in f is known *not* to be satisfied by the known attributes in x, then $f(x) = -$. Otherwise $f(x) = ?$. Note that it is possible for $f(x) = ?$ in the RUAV model when $f(x) = +$ in the UAV model. For example, suppose $f = x_1x_2 + \overline{x_1}x_3$. So $f' = x_1x_2 + \overline{x_1}x_3 + x_2x_3$. Now consider the example $x = *11$. Notice that both x_1x_2 and $\overline{x_1}x_3$ individually could be satisfied or not satisfied. Thus in the RUAV model, $f(*11) = ?$. However, the prime implicant x_2x_3 is satisfied by $*11$ and thus in the UAV model, $f(*11) = +$. Since the classification of some examples depend on the teacher's representation of the target concept, we refer to this as a *representation-dependent* model.

It is common for a human expert to (maybe subconsciously) encode a scheme

[3] A *prime implicant* of a boolean formula f is a conjunction t (not containing contradictory literals) such that t implies f, but no proper subset of t implies f.

Boolean **IsParent**$(proj, i, T_L)$

1 $i' :=$ index for the attribute at the root of T_L ($n+1$ if T_L is a leaf)
2 **for** each leaf ℓ in T_L
3 Let a_ℓ be the partial assignment for $x_{i'}, \ldots, x_n$ that defines the path to ℓ in T_L
4 **if** UAV-MQ$(proj \underbrace{0 \cdots 0}_{i-|proj|-1} \; 1 \underbrace{* \cdots *}_{i'-i-1} a_\ell) \neq$ label of ℓ **then return** true
5 **return** false

Figure 4: The Algorithm **IsParent** which tests whether x_i is the parent of T_L in the subtree $T_{f_{proj}}$. The assignment a_ℓ assigns all variables from $x_{i'}, \ldots, x_n$ on the path from the root of T_L to the leaf ℓ according to the path, and the rest are left unspecified.

for classifying examples in $\{0,1,*\}^n$ as a collection of rules (i.e. a DNF formula). However, as we have noted, evaluating DNF formulas on $\{0,1,*\}^n$ examples is NP-complete. To circumvent this problem, a typical human expert response is likely to evaluate the formula as described above in the definition of the RUAV model. Thus, the RUAV model captures the situation where we are trying to acquire knowledge from such an expert.

Although the class of universal decision trees $\mathcal{T}$ considered in the previous section can be learned in the UAV model using membership queries only, the size of the representation can be exponential in the size of the DNF representation. We now show that DNF formulas are learnable in the RUAV model.

Theorem 12 *The class of DNF boolean formulas can be exactly learned in the RUAV model using $O(n^2m)$ time, $O(n^2m)$ calls to the* RUAV-MQ *oracle, and at most m calls to the* RUAV-EQ *oracle where m is the number of terms in the target concept.*

Proof: Our algorithm **LearnDNF** (see Figure 5) maintains a hypothesis h for the target concept f (where initially $h = \mathit{false}$) with the invariant that the terms of h are a subset of those of f. Thus we do not receive any negative counterexamples and we never incorrectly predict positive (i.e. if $h(x) = +$ then $f(x) = +$). **LearnDNF** processes a counterexample x as follows:

Case 1: Suppose $f(x) = +$. Then we know that there is at least one term which x satisfies but is not present in h. We use a subroutine **ExtractTerm**(x) to construct t and replace h by $h \vee t$.

Case 2: Suppose $f(x) = ?$. Since we never incorrectly predict positive, $h(x) = -$. Thus some term t in f is missing from h where $t(x) = ?$. If $f(x_{*\to 0}) = +$, the instance $x_{*\to 0}$ satisfies f but not h. Thus, as in Case 1, we construct t by calling the subroutine **ExtractTerm**$(x_{*\to 0})$. Now suppose $f(x_{*\to 0}) =$

```
DNF LearnDNF

1   h := false
2   while RUAV-EQ(h) ≠ "yes" do
3       Let x be the counterexample provided
4       if f(x) = + then h := h ∨ ExtractTerm(x)
5       else if f(x_{*→0}) = + then h := h ∨ ExtractTerm(x_{*→0})
6       else
7           x^+ := PositiveCompletion(x)
8           h := h ∨ ExtractTerm(x^+)
9   return h
```

Figure 5: The algorithm **LearnDNF** that learns the class of DNF formulas in the RUAV model using membership and equivalence queries. The model is defined in such a way that no evaluation oracle is needed.

```
Term ExtractTerm(x)

1   t := true
2   for i = n downto 1
3       if RUAV-MQ(x_{x_i→*}) = + then x := x_{x_i→*}
4       else if x_i = 1 then t := t ∧ x_i
5       else t := t ∧ x̄_i
6   return t
```

Figure 6: The algorithm **ExtractTerm** that takes a positive example x and constructs a term t in $f - h$.

$-$. Then we use **PositiveCompletion**(x) to construct a positive example y that is a completion of x. We then call **ExtractTerm**(y) to construct t.

Given an example x for which $f(x) = +$, the procedure **ExtractTerm** (see Figure 6) determines a term t in $f - h$ for which $t(x) = +$. It achieves this goal by flipping as many attributes in x to $*$ as possible without making the example non-positive. The remaining bits then specify t. We now argue that **ExtractTerm** is correct. If $f(x_{x_i \to *}) = +$ then there still exists some term in f that is satisfied by $x_{x_i \to *}$ and hence we can replace x by $x_{x_i \to *}$. Now if $f(x_{x_i \to *}) = -$ then depending on the value of x_i, either x_i or $\overline{x_i}$ must appear in all the terms in f that are satisfied by x. Thus after all attributes have been considered, there is only one term in f that satisfies x. Otherwise some non-*

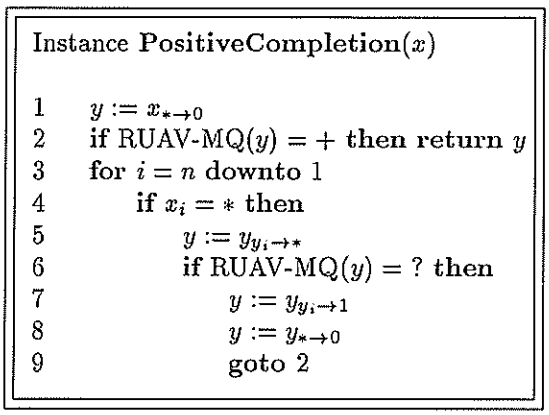

Figure 7: The algorithm **PositiveCompletion** that takes an example x for which $f(x) = ?$ and $f(x_{*\to 0}) = -$ and constructs a positive example y that is a completion of x.

bit in x would have been flipped to "$*$".

It remains to show that given a "?" instance x where $f(x_{*\to 0}) = -$, the algorithm **PositiveCompletion** (see Figure 7) constructs a positive instance y that is a completion of x. Since x is a "?" instance but $h(x) = -$, we know there is at least one term in $f - h$ not falsified by x. Denote the set of such terms by T'. Let P be the set of positive literals from T' that are unspecified in x. Since $f(x_{*\to 0}) = -$, we know that each term in T' has at least one positive literal in P. The idea here is to find all the positive literals of one term in T' and then derive a positive counterexample by setting these literals to 1 and the "$*$" bits to 0. **PositiveCompletion** begins with $y = x_{*\to 0}$ (and since **PositiveCompletion** was called we know that $f(y) = -$). Next, the unspecified bits of x, now 0 in y, are flipped back to "$*$" one at a time starting from x_n until we find a "?" example y. This will eventually occur when we reach the variable x_i where

$$i = \max_{term\ t \in T'} (\text{min index of the positive literals in } t).$$

After this, **PositiveCompletion** flips y_i back to 1 and all the "$*$" bits to 0. Let j_ℓ denote the value of i when we exit the for loop (via the goto in Line 9) for the ℓth time and let $\mathcal{L}_t$ be the set of literals in $P - \{x_{j_1}, \ldots x_{j_\ell}\}$ that are also in t. Note that if all the positive literals t are in $\{x_{j_1}, \ldots, x_{j_\ell}\}$ then the label of y created after the ℓth execution of the for loop will be positive, and thus we will return y in Line 2. Otherwise, for every term in T' at least one positive literal is missing from $\{x_{j_1}, \ldots, x_{j_\ell}\}$. In this case,

$$j_{\ell+1} = \max_{term\ t \in T'} \min_{x_i \in \mathcal{L}_t} i$$

and hence we will reach a "?" example and go to Line 2.

Finally, since **PositiveCompletion** and **ExtractTerm** make $O(n^2)$ and $O(n)$ membership queries respectively, we have the desired result. □

7 CONCLUSION

In this paper, we introduced a variant of Angluin's [3] exact learning model where some of the attributes in the examples are unspecified. While we gave some initial results for this model, many interesting questions have been raised.

An interesting direction is to see if the class of DNF formulas is efficiently learnable in the UAV model using UAV-MQ and UAV-EQ oracles as well as an EV oracle. An easier question in this direction is to see if the class of DNF formulas with constant monotone dimension is learnable in the UAV model. While we can learn general decision trees in the UAV model (see Corollary 4), an EV oracle is needed since the hypothesis class is not a decision tree. Can general decision trees be properly learned in the UAV model, thus removing the need for an EV oracle?

Another direction would be to look at a variation of the UAV model in which there is a cost for obtaining the value of an initially unspecified attribute. There has been some research in this direction (such as the work of Greiner, Grove and Roth [25]). By allowing the cost to possibly be infinite this could model the situation in which some unspecified attributes could be obtained (at some given cost), yet the values of some unspecified attributes are just not available.

Acknowledgments

We thank Krishnan Pillaipakkamnatt, Lisa Hellerstein, Nina Mishra and Lenny Pitt for helpful conversations regarding this work. We also thank the COLT committee members for their helpful comments. Stephen Scott and Stephen Kwek performed this work at Washington University.

References

[1] 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.

[2] D. Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75(2):87–106, November 1987.

[3] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[4] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.

[5] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40:185–210, 1993.

[6] D. Angluin and M. Kriķis. 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.

[7] Dana Angluin and Donna K. Slonim. Randomly fallible teachers: Learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14:7–26, 1994.

[8] R. Bareiss B. W. Porter and R. Holte. Concept learning and heuristic classifcation in weak-theory domains. *Artificial Intelligence*, 45:229–263, 1990.

[9] S. Ben-David and E. Dichterman. Learning with restricted focus of attention. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 287–296. ACM Press, New York, NY, 1993.

[10] Andreas Birkendorf, Eli Dichterman, Jeffrey Jackson, Norbert Klasner, and Hans Ulrich Simon. On restricted-focus-of-attention learnability of Boolean functions. *Machine Learning*, 30:89–123, 1998.

[11] Andreas Birkendorf, Eli Dichterman, Norbert Klasner, and Hans Ulrich Simon. Structural results about exact learning with unspecified attribute values. In *Proc. 11th Annu. Conf. on Comput. Learning Theory*, pages 144–153, 1998.

[12] Avrim Blum, Prasad Chalasani, Sally A. Goldman, and Donna K. Slonim. Learning with unreliable boundary queries. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 98–107. ACM Press, New York, NY, 1995.

[13] Avrim Blum and Stephen Rudich. Fast learning of k-term DNF formulas with queries. *J. of Comput. Syst. Sci.*, 51(3):367–373, 1995.

[14] M. Blum, A. Chandra, and M. Wegman. Equivalence of free Boolean graphs can be decided probabilistically in polynomial time. *Information Processing Letters*, 10:80–82, 1980.

[15] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984.

[16] N. Bshouty, T. Hancock, L. Hellerstein, and M. Karpinski. An algorithm to learn read-once threshold formulas, and transformations between learning models. *Computational Complexity*, 4:37–61, 1994.

[17] N. H. Bshouty, R. Cleve, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *J. of Comput. Syst. Sci.*, 52(3):421–433, 1996.

[18] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. Comput.*, 24(4):706–735, 1995.

[19] Nader H. Bshouty. Exact learning via the monotone theory. *Inform. Comput.*, 123(1):146–153, 1995.

[20] Nader H. Bshouty. Simple learning algorithms using divide and conquer. *Computational Complexity*, 6(2):174–194, 1997.

[21] S. Fortune, J. Hopcroft, and E. Schmidt. The complexity of equivalence and containment for free single variable program schemes. In *Lecture Notes in Computer Science 62: Proceedings of Automata, Languages and Programming*, pages 227–240. Springer-Verlag, 1978.

[22] M. Frazier, S. Goldman, N. Mishra, and L. Pitt. Learning from a consistently ignorant teacher. *J. of Comput. Syst. Sci.*, 52(3):472–492, 1996. Special Issue on COLT '94.

[23] S. A. Goldman and R. H. Sloan. Can PAC learning algorithms tolerate random attribute noise? *Algorithmica*, 14:70–84, 1995.

[24] Sally A. Goldman, Stephen S. Kwek, and Stephen D. Scott. Learning from examples with unspecified attribute values. In *Proc. 10th Annu. Conf. on Comput. Learning Theory*, pages 231–242. ACM Press, New York, NY, 1997.

[25] Russell Greiner, Adam J. Grove, and Dan Roth. Learning active classifiers. In *Proc. 13th International Conference on Machine Learning*, pages 207–215. Morgan Kaufmann, 1996.

[26] T. Hancock. Identifying μ-decision trees and μ-formulas with constrained instance queries. Manuscript, Harvard University, 1989.

[27] L. Hellerstein and M. Karpinski. Learning read-once formulas using membership queries. In *Proc. of the Second Annual Workshop on Computational Learning Theory*, pages 146–161. Morgan Kaufmann, 1989.

[28] H. Hunt and R. Stearns. Monotone Boolean formulas, distributive lattices, and the complexity of logics, algebraic structures, and comutation structures. In *Lecture Notes in Computer Science: Proceedings of 3rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 277–287. Springer-Verlag, 1986.

[29] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22:807–837, 1993.

[30] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497, 1994.

[31] Roni Khardon and Dan Roth. Learning to reason with a restricted view. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 301–310. ACM Press, New York, NY, 1995.

[32] P.D. Laird. *Learning from Good and Bad Data.* Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1988.

[33] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 147–156, San Mateo, CA, 1991. Morgan Kaufmann.

[34] K. Pillaipakkamnatt and V. Raghavan. Read-twice DNF formulas are properly learnable. *Inform. Comput.*, 122(2):236–267, 1995.

[35] J. R. Quinlan. Unknown attribute values in induction. In *Proceedings of the 6th International Machine Learning Workshop*, pages 164–168, 1989.

[36] V. Raghhavan and D. Wilkins. A nearly-linear time equivalence test and characterization of μ-branching programs. *Mathematical Systems Theory*, 30:249–283, 1997.

[37] Y. Sakakibara. On learning from queries and counterexamples in the presence of noise. *Inform. Proc. Lett.*, 37(5):279–284, March 1991.

[38] Dale Schuurmans and Russell Greiner. Learning default concepts. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 519–523, 1994.

[39] Dale Schuurmans and Russell Greiner. *Learning to Classify Incomplete Examples.* In *Computational Learning Theory and Natural Learning Systems*, Volume IV: Making Learning Systems Practical, chapter 6, pages 87–105. MIT Press, 1997.

[40] G. Shackelford and D. Volper. Learning k-DNF with noise in the attributes. In *Proc. 1st Annu. Workshop on Comput. Learning Theory*, pages 97–103, San Mateo, CA, 1988. Morgan Kaufmann.

[41] R. H. Sloan. Four types of noise in data for PAC learning. *Information Processing Letters*, 54:157–162, 1995.

[42] 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.

[43] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.