Exact Identification of Read-once Formulas Using Fixed Points of Amplification Functions

Sally A. Goldman
Department of Computer Science
Washington University
St. Louis, MO 63130
sg@cs.wustl.edu

Michael J. Kearns
AT&T Bell Laboratories
Murray Hill, NJ 07974
mkearns@research.att.com

Robert E. Schapire
AT&T Bell Laboratories
Murray Hill, NJ 07974
schapire@research.att.com

March 9, 1992

Abstract

In this paper we describe a new technique for *exactly identifying* certain classes of read-once Boolean formulas. The method is based on sampling the input-output behavior of the target formula on a probability distribution that is determined by the *fixed point* of the formula's *amplification function* (defined as the probability that a 1 is output by the formula when each input bit is 1 independently with probability p). By performing various statistical tests on easily sampled variants of the fixed-point distribution, we are able to efficiently infer *all* structural information about any logarithmic-depth formula (with high probability). We apply our results to prove the existence of short *universal identification sequences* for large classes of formulas. We also describe extensions of our algorithms to handle high rates of noise, and to learn formulas of unbounded depth in Valiant's model with respect to specific distributions.

Most of this research was carried out while all three authors were at MIT Laboratory for Computer Science with support provided by ARO Grant DAAL03-86-K-0171, DARPA Contract N00014-89-J-1988, NSF Grant CCR-88914428, and a grant from the Siemens Corporation. R. Schapire received additional support from AFOSR Grant 89-0506 while at Harvard University. S. Goldman is currently supported in part by a G.E. Foundation Junior Faculty Grant and NSF Grant CCR-9110108.

1 Introduction

In this paper we describe efficient algorithms for *exactly identifying* certain classes of read-once Boolean formulas by observing the target formula's behavior on examples drawn randomly according to a fixed and simple distribution that is related to the formula's *amplification function*. The class of read-once Boolean formulas is the subclass of Boolean formulas in which each variable appears at most once. The amplification function $A_f(p)$ for a function $f : \{0,1\}^n \to \{0,1\}$ is defined as the probability that the output of f is 1 when each of the n inputs to f is 1 independently with probability p. Amplification functions were first studied by Valiant [24] and Boppana [4, 5] in obtaining bounds on monotone formula size for the majority function.

The method used by our algorithms is of central interest. For several classes of formulas, we show that the behavior of the amplification function is *unstable* near the fixed point; that is, the value of $A_f(p)$ varies greatly with a small change in p. This in turn implies that small but easily sampled perturbations of the fixed-point distribution (that is, the distribution where each input is 1 with probability p, where $A_f(p) = p$) reveal structural information about the formula. For instance, a typical perturbation of the fixed-point distribution hard-wires a single variable to 1 and sets the remaining variables to 1 with probability p.

We apply this method to obtain efficient algorithms for exact identification of classes of read-once formulas over various bases. These include the class of logarithmic-depth read-once formulas constructed with NOT gates and three-input majority gates (for which the fixed-point distribution is the uniform distribution), as well as the class of logarithmic-depth read-once formulas constructed with NAND gates (for which the fixed-point distribution assigns 1 to each input independently with probability $1/\phi \approx 0.618$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio). Thus, for these classes, since the fixed point of the amplification function is the same for all formulas in the class, for each class we obtain a simple product distribution under which the class is learnable. As proved by Kearns and Valiant [16, 13], these same classes of formulas cannot be even weakly approximated in polynomial time when no restriction is placed on the target distribution; thus, our results may be interpreted as demonstrating that while there are some distributions that in a computationally bounded setting reveal essentially *no* information about the target formula, there are natural and simple distributions that reveal *all* information.

For Boolean read-once formulas (a superset of the class of formulas constructed from NAND gates) there is an efficient, exact-identification algorithm using membership and equivalence queries due to Angluin, Hellerstein and Karpinski [1, 11]. The class of read-once majority formulas can also be exactly identified using membership and equivalence queries, as proved by Hancock and Hellerstein [9] and Bshouty, Hancock, Hellerstein, and Karpinski [6]. Briefly, in the query model, the learner attempts to infer the target formula by asking questions, or *queries*, of a "teacher." For instance, the learner might ask the teacher what the formula's output would be for a specific assignment to the input variable; this is called a *membership query*. On an *equivalence query*, the learner asks if a given conjectured formula is equivalent to the target formula

Note that our algorithms' use of a *fixed* distribution can be regarded as a form of "random" membership queries, since this fixed and known distribution can be easily simulated by making random membership queries. Thus, our algorithms are the first efficient procedures for exact identification of logarithmic-depth majority and NAND formulas using only membership queries. Furthermore, the queries used are *non-adaptive* in the sense that they do not depend upon the answers received to previous queries. In contrast, all previous algorithms for exact identification, including the algorithms mentioned above, require highly adaptive queries. As a consequence, our results can be applied to prove the existence of polynomial-length *universal identification sequences* for large classes of formulas; these are fixed sequences of instances for which every unique formula in the class induces a different labeling.

We also prove that our algorithms are *robust* against a large amount of *random misclassification noise*, similar to, but slightly more general than that considered by Sloan [23] and Angluin and Laird [2]. Specifically, if η_0 and η_1 represent the respective probabilities that an output of 0 or 1 is misclassified, then a robust version of our algorithm can handle any noise rate for which $\eta_0 + \eta_1 \neq 1$; the sample size and computation time required increase only by an inverse quadratic factor in $|1 - \eta_0 - \eta_1|$. Again regarding our algorithms as using "random" membership queries, these are the first efficient procedures performing exact identification in some reasonable model of *noisy queries*. Our algorithms can also tolerate a modest rate of *malicious noise*, as considered by Kearns and Li [14].

Finally, we present an algorithm that learns *any* (not necessarily logarithmic-depth) read-once majority formula in Valiant's model against the uniform distribution. To obtain this result we first show that the target formula can be well approximated by truncating the formula to have only logarithmic depth. We then generalize our algorithm for learning logarithmic-depth read-once formulas to handle such truncated formulas. A similar result also holds for read-once NAND formulas of unbounded depth.

The problem of learning Boolean formulas against special distributions has been considered by a number of other authors. In particular, our technique closely resembles that used by Kearns et al. [15] for learning the class of read-once formulas in disjunctive normal form (DNF) against the uniform distribution. A similar result, though based on a different method, was obtained by Pagallo and Haussler [18]. These results were extended by Hancock and Mansour [10], and by Schapire [21] as described below.

Also, Linial, Mansour and Nisan [17] used a technique based on Fourier spectra to learn the class of constant-depth circuits (constructed from gates of unbounded fan-in) against the uniform distribution. Furst, Jackson and Smith [7] generalized this result to learn this same class against any *product distribution* (i.e., any distribution in which the setting of each variable is chosen independently of the settings of the other variables). Verbeurgt [25] gives a different algorithm for learning DNF-formulas against the uniform distribution. However, all three of these algorithms require quasi-polynomial ($n^{\mathrm{polylog}(n)}$) time, though Verbeurgt's procedure only requires a polynomial-size sample.

Finally, Schapire [21] has recently extended our technique to handle a probabilistic generalization of the class of all read-once Boolean formulas constructed from the usual basis $\{\textsc{and}, \textsc{or}, \textsc{not}\}$. He shows that an arbitrarily good approximation of such formulas can be inferred in polynomial time against any product distribution.

2 Preliminaries

Given a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, Boppana [4, 5] defines its *amplification function* A_f as follows: $A_f(p) = \mathbf{Pr}[f(X_1, \ldots, X_n) = 1]$, where $X_1, \ldots, X_n$ are independent Bernoulli variables that are each 1 with probability p. The quantity $A_f(p)$ is called the *amplification of f at p*. Valiant [24] uses properties of the amplification function to prove the existence of monotone Boolean formulas of size $O(n^{5.3})$ for the majority function on n inputs. We denote by $D^{(p)}$ the distribution over $\{0,1\}^n$ induced by having each variable independently set to 1 with probability p.

For $q_j \in \{0,1\}$ and $i_j \in \{1, \ldots, n\}$, $1 \leq j \leq r$, we write $f|x_{i_1} \leftarrow q_1, \ldots, x_{i_r} \leftarrow q_r$ to denote the function obtained from f by fixing or hard-wiring each variable x_{i_j} to the value q_j. If each $q_j = q$ for some value q, we abbreviate this by $f|x_{i_1}, \ldots, x_{i_r} \leftarrow q$.

In our framework, the learner is attempting to infer an unknown *target concept* c chosen from some known *concept class* $\mathcal{C}$. In this paper, $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$ is parameterized by the number of variables n, and each $c \in \mathcal{C}_n$ represents a Boolean function on the domain $\{0,1\}^n$. A polynomial-time learning algorithm achieves *exact identification* of a concept class (from some source of information about the target, such as examples or queries) if it can infer a concept that is equal to the target concept on all inputs. A polynomial-time learning algorithm achieves *exact identification with high probability* if for any $\delta > 0$, it can with probability at least $1 - \delta$ infer a concept that is equal to the target concept on all inputs. In this setting polynomial time means polynomial in n and $1/\delta$. Our algorithms achieve exact identification with high probability when the example source is a particular, fixed distribution.

In the *distribution-free* or *probably approximately correct* (PAC) learning model, introduced by Valiant [24], the learner is given access to labeled (positive and negative) examples of the target concept, drawn randomly according to some unknown *target distribution* D. The learner is also given as input positive real numbers ϵ and δ. The learner's goal is to output with probability at least $1-\delta$ a *hypothesis* h that has probability at most ϵ of disagreeing with c on a randomly drawn example from D (thus, the hypothesis has *accuracy* at least $1-\epsilon$). In the case of a randomized hypothesis h, the probability that h disagrees with c is taken over both the random draw from D and the random coin flips used by h. If such a learning algorithm exists (that is, a polynomial-time algorithm meeting the goal for any $n \geq 1$, any $c \in \mathcal{C}_n$, any distribution D, and any ϵ, δ), we say that $\mathcal{C}$ is *PAC-learnable*. In this setting, polynomial time means polynomial in n, $1/\epsilon$ and $1/\delta$. In this paper, we are primarily interested in a variant of Valiant's model in which the target distribution is known a priori to belong to a specific restricted class of distributions. This distribution-specific model has also been studied by Benedek and Itai [3].

In this paper we give efficient algorithms that with high probability exactly identify the following classes of formulas:

- **The class of logarithmic-depth read-once majority formulas:** This class consists of Boolean formulas of logarithmic depth constructed from the basis {MAJ, NOT} where a MAJ gate computes the majority of three inputs, and each variable appears at most once in the formula. Without loss of generality, we assume all NOT gates are at the input level.

- **The class of logarithmic-depth read-once positive NAND formulas:** This is the class of read-once formulas of logarithmic depth constructed from a basis {NAND} where a NAND gate computes the negation of the logical AND of two inputs. Note that each input appears at most once in the formula and all variables are positive (unnegated). Observe also that the class of read-once positive NAND formulas is equivalent to the class of read-once formulas constructed from alternating levels of OR/AND gates, starting with an OR gate at the top level, with the additional condition that each variable is negated if and only if it enters an OR gate. This observation is easily proved by repeated application of DeMorgan's law.

  Note that by a symmetry argument we can also handle read-once formulas constructed over the {NOR} basis.

In both cases above, our results depend on the restriction on the fan-in of the gates, as well as the requirement that only variables (and not constants) appear in the leaves of the target formula.

Note that because we consider only read-once formulas, there is a unique path from any gate or variable to the output. We define the *level* or *depth of a gate* λ to be the number of gates (not including λ itself) on the path from λ to the output. Thus, the output gate is at level 0. Likewise, we define the *level* or *depth of an input variable* to be the number of gates on the path from the variable to the output. The *depth* of the entire formula is the maximum level of any input, and the *bottom level* consists of all gates and variables of maximum depth.

An input x_i, or a gate λ, *feeds* a gate λ' if the path from x_i or λ to the output goes through λ'. If x_i or λ is an input to λ', then we say that x_i or λ *immediately feeds* λ'. For any two input bits x_i and x_j we define $\Gamma(x_i, x_j)$ to be the deepest gate λ fed by both x_i and x_j. Likewise, $\Gamma(x_i, x_j, x_k)$ is the deepest gate λ fed by x_i, x_j, and x_k. We say that a pair of variables x_i and x_j *meet* at the gate $\Gamma(x_i, x_j)$. Also, if $\Gamma(x_i, x_j) = \Gamma(x_i, x_k) = \Gamma(x_j, x_k) = \Gamma(x_i, x_j, x_k)$, then we say that the variables x_i, x_j and x_k *meet* at gate $\Gamma(x_i, x_j, x_k)$; otherwise, the triple does not meet in the formula. (Note that this only makes sense if there are gates with more than two inputs, such as a three-input majority gate.)

All logarithms in this paper are base 2.

3 Exact identification of read-once majority formulas

In this section we use properties of amplification functions to obtain a polynomial-time algorithm that with high probability exactly identifies any read-once majority formula of logarithmic depth from random examples drawn according to a uniform distribution.

This type of formula is used by Schapire [20] in his proof that a concept class is weakly learnable in polynomial time if and only if it is strongly learnable in polynomial time. That is, the hypothesis output by his boosting procedure can be viewed as a majority formula whose inputs are the hypotheses output by the weak learning algorithm. We also note that a read-once majority formula cannot in general be converted into a read-once Boolean formula over the usual $\{\text{AND}, \text{OR}, \text{NOT}\}$ basis. (For example, a single three-input majority gate cannot be converted into a read-once Boolean formula over this basis; this can be proved, for instance, by enumerating all three-input read-once Boolean formulas.)

It can be shown that the class of logarithmic-depth read-once majority formulas is not learnable in the distribution-free model, modulo assorted cryptographic assumptions.[1] Briefly, this can be proved using a Pitt and Warmuth-style "prediction-preserving reduction" [19] to show that learning read-once majority formulas is at least as hard as learning general Boolean formulas. Our reduction starts with a given Boolean formula which we can assume without loss of generality has been converted using standard techniques into an equivalent formula of logarithmic depth. The main idea of the reduction is to replace

[1] Namely, these hardness results depend on the assumed intractability of factoring Blum integers, inverting RSA functions, and recognizing quadratic residues. See Kearns and Valiant [16] for details.

each OR gate (respectively, AND gate) occurring in this formula with a MAJ gate, one of whose inputs is wired to a distinct variable that, under the target distribution, always has the value 1 (respectively, 0). The resulting majority formula can further be reduced to one that is read-once using the substitution method of Kearns et al. [15]. Finally, combined with Kearns and Valiant's result that Boolean formulas are not learnable (modulo cryptographic assumptions), this shows that majority formulas are also not learnable.

Despite the hardness of this class in the general distribution-free framework, we show that the class is nevertheless exactly identifiable when examples are chosen from the uniform distribution. The algorithm consists of two phases. In the first phase, we determine the relevant variables (i.e., those that occur in the formula), their signs (i.e., whether they are negated or not), and their levels. To achieve this goal, for each variable, we hard-wire its value to 1 and estimate the amplification of the induced function at $\frac{1}{2}$ using examples drawn randomly from the uniform distribution on the remaining variables. Here, by "hard-wiring" a variable to 1, we really mean that we apply a *filter* that only lets through examples for which that variable is 1. We prove that if the variable is relevant, then with high probability this estimate will be significantly smaller or greater than $\frac{1}{2}$, depending on whether the variable occurs negated or unnegated in the formula; otherwise, this estimate will be near $\frac{1}{2}$. Furthermore, the level of a relevant variable can be determined from the amount by which the amplification of the induced function differs from $\frac{1}{2}$.

In the second phase of the algorithm, we construct the formula. More precisely, we first construct the bottom level of the formula, and then recursively construct the remaining levels. To construct the bottom level of the formula, we begin by finding triples of variables that are inputs to the same bottom-level gate. To do this, for each triple of relevant variables that have the largest level number, we hard-wire the three variables to 1 and again estimate the amplification of the induced function from random examples. We show that we can determine whether the three variables all enter the same bottom-level gate based on this estimate.

Briefly, the recursion works as follows. When constructing level t of the formula, if we find that x_i, x_j, and x_k are inputs to the same level-t gate, then in the recursive call we replace x_i, x_j, and x_k by a level-t meta-variable $y \equiv \text{MAJ}(x_i, x_j, x_k)$. Since y is a known subformula, its output on any example can be easily computed and y can be treated like an ordinary variable. Furthermore, since $\frac{1}{2}$ is the fixed point for the amplification function of any read-once majority formula, it follows that y is 1 with probability $\frac{1}{2}$. Thus, for the recursive call we replace all triples of variables that enter level-t gates with meta-variables, and we easily obtain our needed source of random examples drawn according to the uniform distribution on the new variable set from the original source of examples.

For the remainder of this section, we explore some of the properties of the amplification function of read-once majority formulas, leading eventually to a proof of the correctness of this algorithm.

Lemma 3.1 *Let X_1, X_2 and X_3 be three independent Bernoulli variables, each* 1 *with probability p_1, p_2 and p_3, respectively. Then* $\mathbf{Pr}[\textsc{maj}(X_1, X_2, X_3) = 1] = p_1p_2 + p_1p_3 + p_2p_3 - 2p_1p_2p_3$.

Proof: The stated probability is exactly the chance that at least two of the three variables are 1. ■

Lemma 3.1 implies that $A_f(\frac{1}{2}) = \frac{1}{2}$ for any read-once majority formula f. Thus, $\frac{1}{2}$ is a fixed point of A_f.

Our approach depends on the fact that the first derivative of A_f is large at $\frac{1}{2}$, meaning that a slight perturbation of $D^{(1/2)}$ (i.e., the uniform distribution) tends to perturb the statistical behavior of the formula sufficiently to allow exact identification. See Figure 1 for a graph showing the amplification function for balanced read-once majority formulas of various depths.

We perturb $D^{(1/2)}$ by hard-wiring a small number of variables to be 1; such perturbations can always be efficiently sampled by simply waiting for the desired variables to be simultaneously set to 1 in a random example from $D^{(1/2)}$.

We begin by considering the effect on the function's amplification of altering the probability with which one of the variables x_j is set to 1. This will be important in the analysis that follows.

Lemma 3.2 *Let f be a read-once majority formula, and let t be the level of an unnegated variable x_j. Then for $q \in \{0, 1\}$, $A_{f|x_j \leftarrow q}(\frac{1}{2}) = \frac{1}{2} + \left(\frac{1}{2}\right)^t (q - \frac{1}{2})$.*

Proof: By induction on t. When $t = 0$, the formula consists just of the variable x_j, and the lemma holds. For the inductive step, let f_1, f_2 and f_3 be the functions computed by the three subformulas obtained by deleting the output gate of f; thus, f is just the majority of f_1, f_2 and f_3. Note that x_j occurs in exactly one of these three subformulas—assume it occurs in the first. Since x_j occurs at level $t-1$ of this subformula, by the inductive hypothesis, $A_{f_1|x_j \leftarrow q}(\frac{1}{2}) = \frac{1}{2} + \left(\frac{1}{2}\right)^{t-1} (q - \frac{1}{2})$, and since x_j does not occur in the other subformulas, $A_{f_i|x_j \leftarrow q}(\frac{1}{2}) = \frac{1}{2}$ for $i = 2, 3$. From Lemma 3.1, it follows that $A_{f|x_j \leftarrow q}(\frac{1}{2})$ has the stated value, completing the induction. ■

It can now be seen how we use the amplification function to determine the relevant variables of f: if x_j is relevant, then the statistical behavior of the output of f changes significantly when x_j is hard-wired to 1. Similarly, the sign and the level of each variable can be readily determined in this manner.

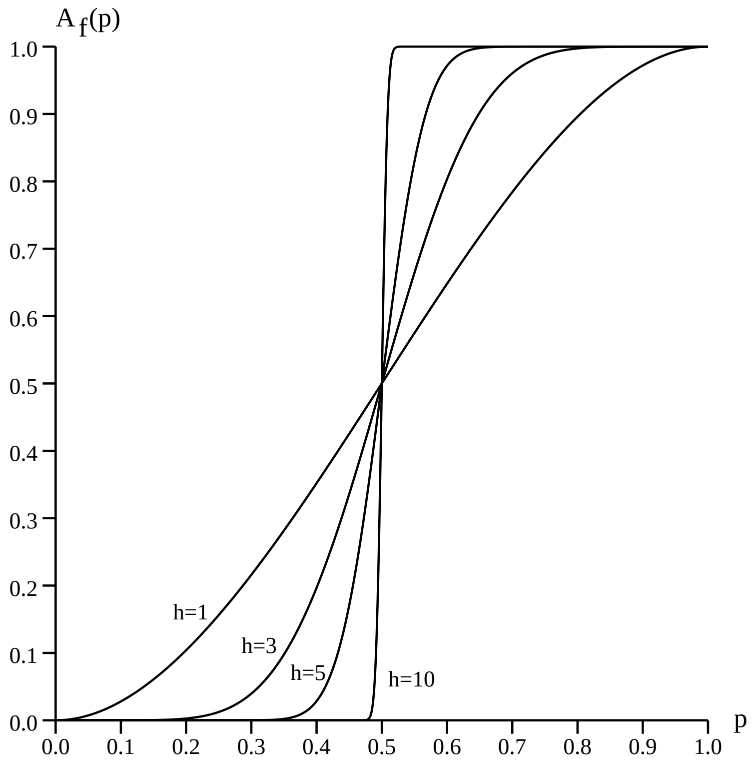

Figure 1: The amplification function for read-once majority formulas for complete ternary trees of depth h.

Theorem 3.3 *Let f be a read-once majority formula of depth h. Let $\hat{\alpha}$ be an estimate of $\alpha = A_{f|x_j \leftarrow 1}(\frac{1}{2})$ for some variable x_j, and assume that $|\hat{\alpha} - \alpha| < \tau \leq \left(\frac{1}{2}\right)^{h+2}$. Then*

- *x_j is relevant if and only if $|\hat{\alpha} - \frac{1}{2}| > \tau$;*
- *if x_j is relevant, then it occurs negated if and only if $\hat{\alpha} < \frac{1}{2}$;*
- *x_j occurs at level t if and only if $\left| |\hat{\alpha} - \frac{1}{2}| - \left(\frac{1}{2}\right)^{t+1} \right| < \tau$.*

Proof: This proof follows from straightforward calculations using Lemma 3.2. ■

Thus, if one estimates the value of the amplification function from a sample whose size is polynomial in 2^h, then with high probability one can determine which variables are relevant, as well as the sign and level of every relevant variable. Specifically, we can apply Chernoff bounds [12] to derive a sample size sufficient to ensure that all the above information is properly computed with high probability. We therefore assume henceforth that the level of every variable has been determined, and that (without loss of generality) all variables are relevant and unnegated.

More problematic is determining exactly how the variables are combined in f. A natural approach is to try hard-wiring *pairs* of variables to 1, and to again estimate the amplification of the induced function in the hopes that some structural information will be revealed. The following lemma, that is useful at a later point, shows that this approach fails.

Lemma 3.4 *Let f be a read-once majority formula, and let x_i and x_j be distinct, unnegated variables that occur at levels t_1 and t_2, respectively. Then*

$$A_{f|x_i,x_j\leftarrow 1}(\tfrac{1}{2}) = \tfrac{1}{2} + \left(\tfrac{1}{2}\right)^{t_1+1} + \left(\tfrac{1}{2}\right)^{t_2+1},$$

regardless of the depth d of $\lambda = \Gamma(x_i, x_j)$.

Proof: By induction on d. Let f_1, f_2 and f_3 be the three subformulas of f that are inputs to the output gate so that $f = \text{MAJ}(f_1, f_2, f_3)$.

If $d = 0$, then λ is the output gate, and x_i and x_j occur in two of the subformulas (say, f_1 and f_2, respectively). From Lemma 3.2, it follows that

$$A_{f_k|x_i,x_j\leftarrow 1}(\tfrac{1}{2}) = \tfrac{1}{2} + \left(\tfrac{1}{2}\right)^{t_k}$$

for $k = 1, 2$, and, since neither x_i nor x_j is relevant to f_3, $A_{f_3|x_i,x_j\leftarrow 1}(\frac{1}{2}) = \frac{1}{2}$. The stated value for $A_{f|x_i,x_j\leftarrow 1}(\frac{1}{2})$ follows then from Lemma 3.1.

If $d > 0$, then λ is a gate occurring in one of the subformulas (say f_1) at level $d - 1$ of the subformula. By inductive hypothesis,

$$A_{f_1|x_i,x_j\leftarrow 1}(\tfrac{1}{2}) = \tfrac{1}{2} + \left(\tfrac{1}{2}\right)^{t_1} + \left(\tfrac{1}{2}\right)^{t_2}.$$

Also, $A_{f_k|x_i,x_j\leftarrow 1}(\frac{1}{2}) = \frac{1}{2}$ for $k = 2, 3$. The stated value for $A_{f|x_i,x_j\leftarrow 1}(\frac{1}{2})$ again follows from Lemma 3.1. ∎

Thus, if two relevant variables are hard-wired to 1, no information is obtained by knowing the value of the amplification function. That is, the amplification function is independent of the level at which the two variables meet.

Therefore, we instead consider what happens when three relevant variables of the same level are fixed to 1. In fact, it turns out to be sufficient to do so for triples of variables all of which occur at the bottom level of the formula. We show that by doing so one can determine the full structure of the formula.

For each triple x_i, x_j and x_k all occurring at level t, there are essentially two cases to consider; either

1. the triple x_i, x_j, x_k does not meet in the formula; or
2. the three variables x_i, x_j and x_k meet at the gate $\Gamma(x_i, x_j, x_k)$. We divide this case into two sub-cases:

(a) x_i, x_j and x_k are inputs to the same gate so that $\Gamma(x_i, x_j, x_k)$ occurs at level $t-1$; or

(b) $\Gamma(x_i, x_j, x_k)$ occurs at some level $d < t-1$.

We are interested in separating Case 2a from the other cases by estimating the amplification of the function when all three variables are hard-wired to 1. This is sufficient to reconstruct the structure of the formula: if we can find three variables that are inputs to some gate λ (and there always must exist such a triple), then we can essentially replace the subformula consisting of the three variables and the gate λ by a new meta-variable whose value can easily be determined from the values of the original three variables. Furthermore, since $\frac{1}{2}$ is a fixed point for all read-once majority formulas the meta-variables' statistics will be the same as those of the original variables. Thus, the total number of variables is reduced by two, and the rest of the formula's structure can be determined recursively.

The following two lemmas analyze the amplification of the function when three variables are hard-wired to 1 in both of the above cases. We begin with Case 2:

Lemma 3.5 *Let f be a read-once majority formula. Let x_i, x_j and x_k be three distinct, unnegated inputs that occur at levels t_1, t_2 and t_3, respectively, and which meet at gate $\lambda = \Gamma(x_i, x_j, x_k)$. Let d be the level of λ. Then*

$$A_{f|x_i,x_j,x_k \leftarrow 1}(\tfrac{1}{2}) = \tfrac{1}{2} + \left(\tfrac{1}{2}\right)^{t_1+1} + \left(\tfrac{1}{2}\right)^{t_2+1} + \left(\tfrac{1}{2}\right)^{t_3+1} - \left(\tfrac{1}{2}\right)^{t_1+t_2+t_3-2d-1}.$$

Proof: By induction on d. As in the preceding lemmas, suppose that $f = \text{MAJ}(f_1, f_2, f_3)$. If $d = 0$, then λ is the output gate of f, and, without loss of generality, x_i, x_j and x_k occur one each in f_1, f_2 and f_3, respectively. From Lemma 3.2, $A_{f_r|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2} + \left(\frac{1}{2}\right)^{t_r}$, for $r = 1, 2, 3$. The stated value for $A_{f|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2})$ follows from Lemma 3.1.

If $d > 0$, then one of the subformulas (say, f_1) contains λ at depth $d-1$. By inductive hypothesis,

$$A_{f_1|x_i,x_j,x_k \leftarrow 1}(\tfrac{1}{2}) = \tfrac{1}{2} + \left(\tfrac{1}{2}\right)^{t_1} + \left(\tfrac{1}{2}\right)^{t_2} + \left(\tfrac{1}{2}\right)^{t_3} - \left(\tfrac{1}{2}\right)^{t_1+t_2+t_3-2d-2},$$

and of course, $A_{f_r|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2}$ for $r = 2, 3$. The proof is completed by again applying Lemma 3.1. ■

So, unlike the situation in which only two variables are hard-wired to 1, here the value of the amplification function depends on the level of the formula at which the three variables meet. However, it may be the case that x_i, x_j, and x_k do not meet at all (i.e., we may be in Case 1). The next lemma considers this case.

Lemma 3.6 *Let f be a read-once majority formula. Let x_i, x_j and x_k be three distinct, unnegated inputs that occur at levels t_1, t_2 and t_3, respectively, and for which $\lambda' = \Gamma(x_i, x_j) \neq \Gamma(x_i, x_j, x_k) = \lambda$. Then*

$$A_{f|x_i,x_j,x_k \leftarrow 1}(\tfrac{1}{2}) = \tfrac{1}{2} + \left(\tfrac{1}{2}\right)^{t_1+1} + \left(\tfrac{1}{2}\right)^{t_2+1} + \left(\tfrac{1}{2}\right)^{t_3+1},$$

regardless of the levels d and d' of gates λ and λ'.

Proof: By induction on d. As before, assume that $f = \text{MAJ}(f_1, f_2, f_3)$. If $d = 0$, then λ is the output gate, λ' occurs (say) in f_1, and x_k in f_2. From Lemma 3.4, $A_{f_1|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2} + \left(\frac{1}{2}\right)^{t_1} + \left(\frac{1}{2}\right)^{t_2}$, and from Lemma 3.2, $A_{f_2|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2} + \left(\frac{1}{2}\right)^{t_3}$. Also, $A_{f_3|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2}$. Lemma 3.1 then implies the stated value for $A_{f|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2})$.

If $d > 0$, then λ occurs, say, in f_1. By inductive hypothesis, $A_{f_1|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2} + \left(\frac{1}{2}\right)^{t_1} + \left(\frac{1}{2}\right)^{t_2} + \left(\frac{1}{2}\right)^{t_3}$, and clearly $A_{f_r|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2}) = \frac{1}{2}$ for $r = 2, 3$. An application of Lemma 3.1 completes the induction. ■

Combining these lemmas, we can show that Case 2a can be separated from the other cases by estimating the function's amplification with triples of variables hard-wired to 1.

Theorem 3.7 *Let f be a read-once majority formula. Let x_i, x_j and x_k be three distinct, unnegated, level-t inputs. Let $\hat{\alpha}$ be an estimate of $\alpha = A_{f|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2})$ for which $|\hat{\alpha} - \alpha| < \tau \leq 3\left(\frac{1}{2}\right)^{t+4}$. Then x_i, x_j and x_k are inputs to the same gate of f if and only if $\hat{\alpha} < \frac{1}{2} + \left(\frac{1}{2}\right)^{t} + \tau$.*

Proof: If Case 2a applies, then Lemma 3.5 implies that $\alpha = \frac{1}{2} + \left(\frac{1}{2}\right)^{t}$. Otherwise, if either Case 1 or 2b applies, then Lemmas 3.5 and 3.6 imply that $\alpha \geq \frac{1}{2} + \left(\frac{1}{2}\right)^{t} + 3\left(\frac{1}{2}\right)^{t+3}$. The theorem follows immediately. ■

We are now ready to state the main result of this section:

Theorem 3.8 *There exists an algorithm with the following properties: Given h, n, $\delta > 0$, and access to examples drawn from the uniform distribution on $\{0,1\}^n$ and labeled by any read-once majority formula f of depth at most h on n variables, the algorithm exactly identifies f with probability at least $1 - \delta$. The algorithm's sample complexity is $O(4^h \cdot \log(n/\delta))$, and its time complexity is $O(4^h \cdot (r^3 + n) \cdot \log(n/\delta))$, where r is the number of relevant variables appearing in the target formula.*

Proof: First, for each variable x_i, estimate the function's amplification with x_i hard-wired to 1. (We will ensure that, with high probability, this estimate is within $\left(\frac{1}{2}\right)^{h+2}$ of the true amplification.) It follows from Theorem 3.3 that after this phase of the algorithm, with high probability we know which variables are relevant, and the sign and depth of each relevant variable. (So, we assume from now on that the formula is monotone.)

In the second phase of the algorithm, we build the formula level by level from bottom to top. To build the bottom level, for all triples of variables x_i, x_j, x_k that enter the bottom level, we estimate the amplification with x_i, x_j, and x_k hard-wired to 1. (We will ensure that, with high probability, this estimate is within $3\left(\frac{1}{2}\right)^{h+4}$ of the true amplification.) It follows from Theorem 3.7 that we can determine which variables enter the same bottom-level gates.

We want to recurse to compute the other levels; however, we cannot hard-wire too many variables without the filter requiring too many examples. The key observation is that on examples drawn from the uniform distribution, the output of any subformula is 1 with probability $\frac{1}{2}$. Thus, the inputs into any level are in fact distributed according to a uniform distribution. Since we compute the formula from bottom to top, the filter can just compute the value for the *known* levels to determine the inputs to the level currently being learned. Our algorithm is described in Figure 2.

Given that the estimates for the formula's amplification have the needed accuracy, the proof of correctness follows from Theorems 3.3 and 3.7. To compute the actual sample size needed to make these estimates, we use Hoeffding's inequality [12] (also known as a form of Chernoff bounds) as stated below:

Lemma 3.9 (Hoeffding's Inequality) *Let $X_1, \ldots, X_m$ be a sequence of m independent Bernoulli trials, each succeeding with probability p. Let $S = X_1 + \cdots + X_m$ be the random variable describing the total number of successes. Then for $0 \leq \gamma \leq 1$, the following holds:*

$$\mathbf{Pr}[|S - pm| > \gamma m] \leq 2e^{-2m\gamma^2}.$$

In the first phase of the algorithm, for each variable x_i, we need a good estimate $\hat{\alpha}$ of $\alpha = A_{f|x_i \leftarrow 1}(\frac{1}{2})$; specifically, we require that the chosen sample be sufficiently large that $|\alpha - \hat{\alpha}| < 2^{-(h+2)}$ with probability at least $1 - \delta/2n$. Then every such estimate for the n variables will have the needed accuracy with probability at least $1 - \delta/2$.

Using Hoeffding's inequality it can be shown that a filtered sample (i.e., the sample after all examples are removed for which $x_i = 0$) of size $m_1 > 8 \cdot 4^h \ln(8n/\delta)$ is sufficiently large to ensure that the estimate $\hat{\alpha}$ has the needed accuracy with probability at least $1 - \delta/4n$. Again using Hoeffding's inequality, it can be shown that if we draw an (unfiltered) sample of size at least $\max\{4m_1, 8\ln(4/\delta)\}$ then with probability at least $1 - \delta/4n$, at least $1/4$ of the examples chosen will be such that $x_i = 1$, and thus, the filtered sample will have size at least m_1. So by using a sample of size exceeding $32 \cdot 4^h \ln(8n/\delta)$, all of the estimates satisfy the requirements for the first phase of the algorithm with probability at least $1 - \delta/2$.

In the second phase, we require good estimates of the formula's amplification when triples of variables are hard-wired. In fact, we need such estimates not only when ordinary variables are hard-wired, but also when we hard-wire meta-variables. Note that, assuming all estimates have the needed accuracy, every (meta-)variable added to the set X in Figure 2 in fact

```
LearnMajorityFormula(n, h, δ)
  m ← 228 · 4^h ln(8n^3/δ)
  E ← {m labeled examples from D^(1/2)}
  X ← ∅
  for 1 ≤ i ≤ n
    E' ← examples from E for which x_i = 1
    α̂ ← fraction of E' that are positive
    if |α̂ − 1/2| > (1/2)^(h+2) then
      if α̂ > 1/2 then X ← X ∪ {x_i}
                 else X ← X ∪ {x̄_i}
      t(x_i) ← compute-level(α̂, h)
  BuildFormula(h, X, E)
```

```
BuildFormula(t, X, E)
  if t = 0 then the target formula is the only variable in X
  else
    for all distinct triples x_i, x_j, x_k ∈ X for which t(x_i) = t(x_j) = t(x_k) = t
      E' ← examples from E for which x_i = x_j = x_k = 1
      α̂ ← fraction of E' that are positive
      if α̂ < 1/2 + (1/2)^t + 3 (1/2)^(t+4) then
        let y ≡ MAJ(x_i, x_j, x_k) be a new variable
        t(y) ← t − 1
        X ← (X ∪ {y}) − {x_i, x_j, x_k}
    BuildFormula(t − 1, X, E)
```

Figure 2: Algorithm for exactly identifying read-once majority formulas of depth h. Procedure `compute-level`$(\hat{\alpha}, h)$ computes the level associated with $\hat{\alpha}$ as given by Theorem 3.3.

computes some subformula g of f. Thus, for every triple of subformulas g_1, g_2 and g_3 of f, our algorithm requires an estimate $\hat{\alpha}$ of α, the amplification of f at $\frac{1}{2}$, given that the output of each subformula g_1, g_2 and g_3 is fixed to the value 1. Since a read-once majority formula on n variables has at most $3n/2$ subformulas (since it has at most $n/2$ MAJ gates), we require a sample sufficiently large that $|\alpha - \hat{\alpha}| < 3\left(\frac{1}{2}\right)^{h+4}$ with probability at least $1 - 4\delta/27n^3$. The chance that all of the (at most $(3n/2)^3$) estimates have the needed accuracy is then at least $1 - \delta/2$. The analysis is similar to that given above, yielding the sample size stated in the figure.

Finally, the time complexity follows from the fact that `LearnMajorityFormula` makes n estimates from the sample and `BuildFormula` makes $O(r^3)$ estimates from the sample.

■

Note that our algorithm's sample complexity has only a logarithmic dependence on the number of irrelevant attributes. Also, it follows immediately from Theorem 3.8 that any read-once majority formula of depth $O(\log n)$ can be exactly identified in polynomial time.

Finally, we note that our algorithm can be modified to work without receiving a bound for the height of the formula as input; the time and sample complexity only increase by a factor of two. The idea is to guess an initial value of $h = 1$ and to increment our guess each time the algorithm fails; it can be shown that, if the formula's height is greater than our current guess, then this fact will become evident by our algorithm's inability to successfully construct a formula. (Specifically, the algorithm `BuildFormula` in Figure 2 will reach a point at which there remain level-t variables in X, but no three remaining level-t variables are immediate inputs to the same gate.)

4 Exact identification of read-once positive NAND formulas

In this section we use the properties of the amplification function to obtain a polynomial-time algorithm that with high probability exactly identifies any read-once positive NAND formula of logarithmic depth from $D^{(\psi)}$ where ψ is the constant $(\sqrt{5} - 1)/2 \approx 0.618$. Note that $\psi = 1/\phi = \phi - 1$, where ϕ is the golden ratio.

As noted earlier, the class of read-once positive NAND formulas is equivalent to the class of read-once formulas constructed from alternating levels of OR/AND gates, starting with an OR gate at top level, and with the additional condition that each variable is negated if and only if it enters an OR gate.

We show that this class of formulas is learnable when examples are chosen from a distribution in which each variable is 1 with probability ψ. The basic structure of the algorithm is just like that of the preceding algorithm for identifying read-once majority formulas. In the first phase of the algorithm, we determine the relevant variables and their depths by hard-wiring each variable to 0, and estimating the amplification of the induced function at ψ using random examples from $D^{(\psi)}$. In the second phase of the algorithm, we construct the formula by finding pairs of variables that are direct inputs to a bottom-level gate. Here, we show that this is possible by hard-wiring pairs of variables to 0 and estimating the function's amplification. After learning the structure of the bottom level of the formula, we again are able to construct the remaining levels recursively.

Since the techniques used in this section are so similar to those in Section 3, the proofs of the lemmas and theorems have been omitted. Most of the lemmas can be proved by simple induction arguments as before.

We turn now to a discussion of some of the properties of the amplification function of read-once positive NAND formulas; these lead to a proof of the correctness of our algorithm.

Lemma 4.1 *Let X_1 and X_2 be independent Bernoulli variables, each 1 with probability p_1 and p_2, respectively. Then $\mathbf{Pr}[\text{NAND}(X_1, X_2) = 1] = 1 - p_1 p_2$.*

It is easily verified that $1 - \psi^2 = \psi$, and thus that ψ is a fixed point of the amplification function A_f whenever f is a read-once positive NAND formula. Once again, our approach depends on the fact that slight perturbations of $D^{(\psi)}$ tend to perturb the statistical behavior of the formula sufficiently to allow exact identification.

Lemma 4.2 *Let f be a read-once positive* NAND *formula, and let t be the level of some variable x_j. Then $A_{f|x_j \leftarrow q}(\psi) = \psi + (q - \psi)(-\psi)^t$.*

Thus, hard-wiring an even-leveled input to 0 decreases the amplification while hard-wiring an odd-leveled input to 0 increases the amplification. To give some intuition explaining this behavior, consider the correspondence described above between read-once positive NAND formulas and leveled OR/AND formulas. An even-leveled input corresponds to an input to an AND gate and thus hard-wiring that input to 0 clearly decreases the amplification. However, an odd-leveled input corresponds to an input that is first negated and then fed to an OR gate; thus, this case corresponds to hard-wiring the input to an OR gate to 1 which clearly increases the amplification function.

As we saw in the last section, the amplification function can be used to determine the relevant variables of f: if x_j is relevant then the statistical behavior of the output of f changes significantly when x_j is hard-wired to 0. Similarly, the level of each variable can be computed in this manner.

Theorem 4.3 *Let f be a read-once positive* NAND *formula of depth h. Let $\hat{\alpha}$ be an estimate of $\alpha = A_{f|x_j \leftarrow 0}(\psi)$ for some variable x_j, and assume that $|\hat{\alpha} - \alpha| < \tau \le \psi^{h+1}/2$. Then*

- *x_j is relevant if and only if $|\hat{\alpha} - \psi| > \tau$;*

- x_j *occurs at level* t *if and only if* $|\psi + (-\psi)^{t+1} - \hat{\alpha}| < \tau$.

We next consider the effect on the amplification function of hard-wiring two inputs. Unlike the case of majority formulas, measuring the amplification of the function when pairs of variables are hard-wired to 0 reveals a great deal of information about the structure of the formula. In particular, the value of the amplification function when two level-t variables x_i and x_j are hard-wired to 0 depends critically on the depth of $\Gamma(x_i, x_j)$.

Lemma 4.4 *Let* f *be a read-once positive* NAND *formula, and let* x_i *and* x_j *be two distinct variables that occur at levels* t_1 *and* t_2*, respectively, and for which* $\lambda = \Gamma(x_i, x_j)$ *is at level* d*. Then*

$$A_{f|x_i,x_j \leftarrow 0}(\psi) = \psi + (-\psi)^{t_1+1} + (-\psi)^{t_2+1} - (-\psi)^{t_1+t_2-d}.$$

Using the same ideas as in the last section, it can now be proved that, given a good estimate of the amplification function, one can determine which variables meet at bottom-level gates.

Theorem 4.5 *Let* f *be a read-once positive* NAND *formula. Let* x_i *and* x_j *be two level-*t *inputs. Let* $\hat{\alpha}$ *be an estimate of* $\alpha = A_{f|x_i,x_j \leftarrow 0}(\psi)$ *for which* $|\hat{\alpha} - \alpha| < \tau \le \psi^{t+3}/2$*. Then* x_i *and* x_j *are inputs to the same level-*$(t-1)$ *gate of* f *if and only if* $|\psi + (-\psi)^{t+1} - \hat{\alpha}| < \tau$.

We are now ready to state the main result of this section:

Theorem 4.6 *Let* $\psi = 1/\phi = (\sqrt{5}-1)/2$*. Then there exists an algorithm with the following properties: Given* h*,* n*,* $\delta > 0$*, and access to examples drawn from the distribution* $D^{(\psi)}$ *on* $\{0,1\}^n$ *and labeled by any read-once positive* NAND *formula* f *of depth at most* h *on* n *variables, the algorithm exactly identifies* f *with probability at least* $1-\delta$*. The algorithm's sample complexity is* $O(\phi^{2h} \cdot \log(n/\delta))$*, and its time complexity is* $O(\phi^{2h} \cdot (r^2+n) \cdot \log(n/\delta))$*, where* r *is the number of relevant variables appearing in the target formula.*

Our algorithm is obtained by making straightforward modifications to `LearnMajorityFormula` and `BuildFormula`. The proof that this algorithm is correct follows from the preceding lemmas and theorems, and is similar to the proof of Theorem 3.8.

As before, it follows immediately that any read-once positive NAND formula of depth at most $O(\log n)$ can be exactly identified in polynomial time.

5 Short universal identification sequences

In this section we describe an interesting consequence of our results. Observe that if we regard our algorithms' use of a *fixed* distribution as a form of "random" membership queries, then it is apparent that these queries are *non-adaptive*; each query is independent of all previous answers. In other words, our algorithms pick all membership queries before seeing the outcome of any. From this observation we can apply our results to prove the existence of polynomial-length *universal identification sequences* for classes of formulas, that is, fixed sequences of instances that distinguish all concepts from one another.

More formally, we define an *instance sequence* to be an unlabeled sequence of instances, and an *example sequence* to be a labeled sequence of instances. Let $\mathcal{C}_n$ be a concept class. We say an instance sequence S *distinguishes* a concept c if the example sequence obtained by labeling S according to c distinguishes c from all other concepts in $\mathcal{C}_n$; that is, c is the only concept consistent with the example sequence so obtained. A *universal identification sequence* for a concept class $\mathcal{C}_n$ is an instance sequence that distinguishes *every* concept $c \in \mathcal{C}_n$.

In the language used in the related work of Goldman and Kearns [8], a sequence that distinguishes c would be called a *teaching sequence* for c. Thus, in their terminology, a universal identification sequence is one that acts as a teaching sequence for every $c \in \mathcal{C}_n$. See also the related work of Shinohara and Miyano [22].

The following theorem gives general conditions for when a deterministic exact identification algorithm implies the existence of a polynomial-length universal identification sequence. We then apply this theorem to our algorithms of the previous sections.

Theorem 5.1 *Let $\mathcal{C}_n$ be a concept class with cardinality $|\mathcal{C}_n| \leq 2^{p(n)}$, where $p(n)$ is some polynomial. Let A be a deterministic algorithm that, given $\delta > 0$ and random, labeled examples drawn from some fixed distribution D, exactly identifies any $c \in \mathcal{C}_n$ with probability exceeding $1 - \delta$. Furthermore, suppose that the sample complexity of A is $q(n)\log^k(1/\delta)$ for some polynomial $q(n)$ and constant k. Then there exists a polynomial-length universal identification sequence for $\mathcal{C}_n$. Specifically, there exists such a sequence of length $q(n) \cdot (p(n))^k$.*

Proof: The proof uses a standard probabilistic argument. Fix $c \in \mathcal{C}_n$, and let S be the random example sequence drawn by algorithm A. Since algorithm A achieves exact identification of c with probability exceeding $1 - \delta$ we have that the probability, taken over all random example sequences for c, that A fails to exactly identify the *particular* target concept c is less than δ. Letting $\delta = 2^{-p(n)}$ it follows that the probability, taken over all random instance sequences, that A fails to identify *any* target concept in $\mathcal{C}_n$ is less than $|\mathcal{C}_n|\delta \leq 1$. Thus, with positive probability, an instance sequence of length $q(n)(p(n))^k$ drawn randomly from D causes A to exactly identify any $c \in \mathcal{C}_n$. Call such a sequence *good*. Therefore there must exist some good instance sequence S of this length.

We now show that a good S *distinguishes* all $c \in \mathcal{C}_n$. For $c \in \mathcal{C}_n$, let S_c be the example sequence obtained by labeling S according to c. Suppose some $c' \in \mathcal{C}_n$ is consistent with S_c so that $S_c = S_{c'}$. Since S is a good sequence and since A is a deterministic exact identification algorithm, it follows that on input S_c, A must output c and on input $S_{c'} = S_c$, A must output c'. Clearly this can only be true if $c = c'$. Thus c must be the only concept in $\mathcal{C}_n$ that is consistent with S_c. ■

Applying this theorem to our exact identification algorithms, we obtain the following corollary.

Corollary 5.2 *There exist polynomial-length universal identification sequences for the classes of logarithmic-depth read-once majority formulas and logarithmic-depth read-once positive* NAND *formulas.*

6 Handling random misclassification noise

Because the algorithms described in Sections 3 and 4 are statistical in nature, they are easily modified to handle a considerable amount of noise. In this section, we describe a robust version of our algorithm for learning logarithmic-depth read-once majority formulas. Although omitted, a similar (though slightly more involved) algorithm can be derived for NAND formulas.

Our algorithm is able to handle a kind of random misclassification noise that is similar, but slightly more general than that considered by Angluin and Laird [2], and Sloan [23]. Specifically, the output of the target formula is "flipped" with some fixed probability that may depend on the formula's output. Thus, if the true, computed output of the formula is 0, then the learner sees 0 with probability $1 - \eta_0$, and 1 with probability η_0, for some quantity η_0. Similarly, a true output of 1 is observed to be 0 with probability η_1 and 1 with probability $1 - \eta_1$. When $\eta_0 = \eta_1$, this noise model is equivalent to that considered by Angluin and Laird, and Sloan. Note that when $\eta_0 + \eta_1 = 1$, outputs of 0 or 1 are entirely indistinguishable in an information-theoretic sense. Moreover, we can assume without loss of generality that $\eta_0 + \eta_1 \leq 1$ by symmetry of the behavior of the formula f with its negation $\neg f$.

If we regard our algorithm's use of a fixed distribution as a form of membership query, we can also handle large rates of misclassification noise in the queries. Here the formulation of a meaningful noise model is more problematic. In particular, we wish to disallow the uninteresting technique of repeatedly querying a particular instance in order to obtain its true classification with overwhelming probability. Thus, we consider a model in which noisy labels are *persistent:* for each instance x, on the first query to x, the true output of the target concept is computed and is reversed with probability η_0 or η_1, according to whether the true output is 0 or 1 (as described above). However, on all subsequent queries to x, the label returned is the same as the label returned with the first query to x. A natural interpretation of such persistent noise is that of a teacher who is simply wrong on certain instances, and cannot be expected to change his mind with repeated sampling. This kind of persistent noise is not a problem for our algorithms because, when n is large, the algorithm is extremely unlikely to query the same instance twice.

Our algorithm assumes that $\eta_0 + \eta_1$ is bounded away from 1 so that $\eta_0 + \eta_1 \leq 1 - \rho$ for some known positive quantity ρ. The error rates themselves, η_0 and η_1, are assumed to be unknown. Our algorithm exactly identifies the target formula with high probability in time polynomial in all of the usual parameters, and $1/\rho$.

Our robust algorithm has a similar structure to that of the algorithm described previously for the noise-free case: The algorithm begins by determining the relevance and sign of each variable. However, it is not clear at this point how the level of each variable might be ascertained in the presence of noise. Nevertheless, it turns out to be possible to find three bottom-level variables that are inputs to the same gate. As before, once such a triple has been discovered, the remainder of the formula can be identified recursively.

To start with, note that if p is the probability that a 1 is output by the target formula f under some distribution on $\{0,1\}^n$, then the probability that a 1 is observed by the learner is

$$p(1-\eta_1) + (1-p)\eta_0 = p(1-\eta_0-\eta_1) + \eta_0 \ .$$

Thus, $\tilde{A}_f(p) = A_f(p) \cdot (1-\eta_0-\eta_1) + \eta_0$ is the probability that a 1 is observed when each input is 1 with probability p. Under the uniform distribution, a 1 is observed with probability $\xi = \tilde{A}_f(\frac{1}{2})$. Since η_0 and η_1 are unknown, ξ is unknown as well. However, an accurate estimate $\hat{\xi}$ (say, within $\Theta(\rho/2^h)$ of ξ) can be efficiently obtained in the usual manner by sampling.

The next lemma shows that a variable x_i's relevance and sign can be determined by hard-wiring it to 1 and comparing $\hat{\xi}$ to an estimate of the value $\alpha = \tilde{A}_{f|x_i \leftarrow 1}(\frac{1}{2})$.

Lemma 6.1 *Let f be read-once majority formula of depth h. Let $\hat{\xi}$ and $\hat{\alpha}$ be estimates of $\xi = \tilde{A}_f(\frac{1}{2})$ and $\alpha = \tilde{A}_{f|x_j \leftarrow 1}(\frac{1}{2})$, for some variable x_j. Assume $|\alpha - \hat{\alpha}| < \tau$ and $|\xi - \hat{\xi}| < \tau$ for some $\tau \leq \rho/2^{h+3}$. Then*

- *x_j is relevant if and only if $|\hat{\alpha} - \hat{\xi}| > 2\tau$;*
- *if x_j is relevant, then it occurs negated if and only if $\hat{\alpha} < \hat{\xi}$.*

Proof: Note that $\alpha - \xi = (1-\eta_0-\eta_1)(A_{f|x_j\leftarrow 1}(\frac{1}{2}) - \frac{1}{2})$. The lemma then follows from Lemma 3.2, and by noting that $1-\eta_0-\eta_1 \geq \rho$. ■

More difficult is the problem of determining the level of each variable since η_0 and η_1 are unknown. Nevertheless, it turns out to be possible to identify the formula without first determining the level of each variable. In particular, we can determine a triple of variables that are inputs to the same bottom-level gate. As described in Section 3, once this is done, the three variables can be replaced by a meta-variable, and the rest of the formula can be constructed recursively. Thus, to complete the algorithm, we need only describe a technique for finding such a triple.

From the comments above, we can assume without loss of generality that all variables are relevant and unnegated. The key point, proved below, is the following: $\tilde{A}_{f|x_i,x_j,x_k\leftarrow 1}(\frac{1}{2})$ is minimized over triples x_i, x_j and x_k whenever the three variables are inputs to the same bottom-level gate. Thus, such a triple can be found by estimating $\tilde{A}_{f|x_i,x_j,x_k\leftarrow 1}(\frac{1}{2})$ for each triple and choosing the one with the smallest estimated value.

Lemma 6.2 *Let f be a monotone, read-once majority formula of depth h. For all triples of distinct indices i, j and k, let $\hat{\alpha}_{ijk}$ be an estimate of $\alpha_{ijk} = \tilde{A}_{f|x_i,x_j,x_k\leftarrow 1}(\frac{1}{2})$, and assume that $|\alpha_{ijk} - \hat{\alpha}_{ijk}| < \tau \leq 3\rho/2^{h+4}$. Suppose that $\hat{\alpha}_{qrs} = \min\{\hat{\alpha}_{ijk} : i,j,k \text{ distinct }\}$. Then x_q, x_r and x_s are bottom-level variables that are inputs to the same gate.*

Proof: From Lemma 3.5, if x_i, x_j and x_k are bottom-level variables that are inputs to the same gate, then

$$\alpha_{ijk} = (1-\eta_0-\eta_1)\left(\frac{1}{2} + \left(\frac{1}{2}\right)^h\right) + \eta_0.$$

Otherwise, Lemmas 3.5 and 3.6 imply that

$$\alpha_{ijk} \geq (1-\eta_0-\eta_1)\left(\frac{1}{2} + \left(\frac{1}{2}\right)^h + 3\left(\frac{1}{2}\right)^{h+3}\right) + \eta_0.$$

Furthermore, since $1-\eta_0-\eta_1 \geq \rho$ we get that

$$\alpha_{ijk} \geq (1-\eta_0-\eta_1)\left(\frac{1}{2} + \left(\frac{1}{2}\right)^h\right) + \eta_0 + 3\rho/2^{h+3}.$$

Since each $\hat{\alpha}_{ijk}$ is accurate to within $3\rho/2^{h+4}$, it follows that $\hat{\alpha}_{qrs}$ can be minimal only if x_q, x_r and x_s are bottom-level inputs to the same gate. ■

Thus, Lemma 6.2 gives a technique for finding bottom-level inputs to the same gate, and, as previously mentioned, the remainder of the formula can be constructed recursively as in Section 3. We thus obtain the main result of this section:

Theorem 6.3 *There exists an algorithm with the following properties: Given h, n, $\rho > 0$, $\delta > 0$, and access to examples drawn from the uniform distribution on $\{0,1\}^n$, labeled by* *a read-once majority formula f of depth at most h on n variables, and misclassified with probabilities η_0 and η_1 (as described above) for $\eta_0+\eta_1 \leq 1-\rho$, the algorithm exactly identifies f with probability at least $1-\delta$. The algorithm's sample complexity is $O((4^h/\rho^2)\cdot\log(n/\delta))$, and its time complexity is $O((4^h/\rho^2)\cdot(n+r^3)\cdot\log(n/\delta))$, where r is the number of relevant variables appearing in the target formula.*

Finally, we comment that our algorithms can be extended to handle a modest amount of *malicious noise*. In this model, first considered by Kearns and Li [14], an adversary is allowed to corrupt each example in any manner he chooses (both the labels and the variable settings) with probability η. We can show that the algorithm described in Sections 3 for majority formulas can handle malicious error rates as large as $\Theta(2^{-h})$ where h is the height of the target formula. Thus, for logarithmic-depth formulas, we can handle malicious error rates up to an inverse polynomial in the number of relevant variables. Similar results also hold for NAND formulas.

The extension of the algorithm to handle malicious noise is straightforward. The algorithm of Section 3 depends only on accurate estimates of the amplification of f when some of f's inputs are hard-wired. For instance, in the first phase of the algorithm, we need to estimate, for each input x_i, the amplification $\alpha = A_{f|x_i\leftarrow 1}(\frac{1}{2})$. This quantity is really just the conditional probability

$$\mathbf{Pr}[f=1 \mid x_i=1] = \frac{\mathbf{Pr}[f=1\wedge x_i=1]}{\mathbf{Pr}[x_i=1]} = 2\cdot\mathbf{Pr}[f=1\wedge x_i=1]$$

where the probabilities are computed with respect to the uniform distribution. Thus, we can compute an estimate of α using an estimate for $\beta=\mathbf{Pr}[f=1\wedge x_i=1]$. Note that malicious noise can affect this probability β by at most an additive factor of η. That is, the chance that $f=1$ and $x_i=1$ (in the presence of malicious noise) is at least $\beta-\eta$ and at most $\beta+\eta$.

Thus, using Hoeffding's inequality (Lemma 3.9), an estimate of β that is accurate to within $\eta+\tau$ can be obtained from a sample of size polynomial in $1/\tau$ (with high probability). Such an estimate for β yields an estimate for α that is accurate to within $2(\eta+\tau)$.

A similar argument can be made for computing the estimates required in the second phase of the algorithm. Since Theorems 3.3 and 3.7 show that the required estimates need only be accurate to within $3/2^{h+4}$, it follows that a malicious error rate of, say, $1/16$ this amount can be tolerated without increasing the algorithm's complexity by more than constant factors.

7 Learning unbounded-depth formulas

In this final section, we describe extensions of our algorithms to learn formulas of unbounded depth in Valiant's PAC model with respect to specific distributions. As in the last section, we focus only on majority formulas, omitting the similar application of these techniques to NAND formulas.[2]

For formulas of unbounded depth, exact identification from the uniform distribution in polynomial time is too much to ask: For purely information-theoretic reasons, at least $\Omega(2^h)$ examples must be drawn from the uniform distribution to exactly identify a majority formula of depth h. This can be proved by showing (say, by induction on h) that if x_i occurs at level h of formula f, then 2^{-h} is the probability that an instance is chosen for which the output of f depends on x_i (i.e., for which f's output changes if x_i is flipped). Thus, $\Omega(2^h)$ random examples are needed simply to determine, for example, whether x_i occurs negated or unnegated.

Therefore, to handle arbitrarily deep formulas, we must relax our requirement of exact identification. Instead, we adopt Valiant's criterion of obtaining a good *approximation* of the target concept (with high probability). As before, our algorithms do not work for all distributions, just the fixed-point distribution. We describe an algorithm that, given $\epsilon, \delta > 0$ and access to random examples of the target majority formula drawn from the uniform distribution, outputs with probability $1 - \delta$ an ϵ-*good* hypothesis, that is, one that agrees with the target formula on a randomly chosen instance from the uniform distribution with probability at least $1 - \epsilon$. Furthermore, the running time is polynomial in $1/\delta$, $1/\epsilon$ and the number of variables n.

We begin by briefly discussing the main ideas of the algorithm. First, as noted above, variables that occur deep in the formula are unimportant in the sense that their values are unlikely to influence the formula's output on a randomly chosen instance. Intuitively, we would like to take advantage of this fact by somehow treating such variables as irrelevant. However, they cannot be simply deleted from the formula without leaving "holes" that must in some way be handled.

We therefore introduce the notion of a *partially visible function*. This is a function on a set of *visible* variables whose values can be observed by the learner, and a set of *hidden* variables that are not observable. With respect to a distribution on the set of assignments to the hidden variables, we say that two partially visible Boolean functions are *equivalent* if, for all assignments to the visible variables, the probabilities are the same that each function evaluates to 1 (where the probabilities are taken over random assignments to the hidden variables). In other words, the behaviors of the two functions are indistinguishable with respect to the visible variables.

Thus, we handle all deep variables by regarding them as hidden variables, and the target

[2]Alternatively, the algorithm recently described by Schapire [21] could be used to PAC-learn unbounded-depth NAND formulas. His algorithm is more general but less efficient than the approach described in this section.

formula as one that is partially visible. In particular, *insignificant* variables—those that occur below level $h = \lceil \log(n/2\epsilon) \rceil$—are considered hidden and their actual values ignored. We call the partially visible formula obtained from the target formula f in this manner the *truncated target.*

Our algorithm works by exactly identifying the truncated target, that is, by constructing a partially visible formula f' that is equivalent to it (in the sense described above, with respect to the uniform distribution). It will be shown that f and f' agree on a randomly chosen instance with probability at least $1 - \epsilon$, and therefore f' is an ϵ-good hypothesis satisfying the PAC criterion.

It remains then only to show how f' can be constructed. First, observe that by Lemma 3.2 all significant variables occurring in f can be detected (and their signs and levels determined) in polynomial time. Moreover, by arguments similar to those given in Section 3, it can be shown that if some triple of significant variables meet at a gate, then the level of that gate can be detected from the amplification function by hard-wiring the three variables to 1. We call this information (the level and sign of each significant variable, and the level at which each triple of significant variables meet, if at all) the formula's *schedule*. It turns out that the schedule alone is sufficient to fully re-construct the partially visible formula f', as is shown below.

These then are the main ideas of the algorithm. What follows is a more detailed exposition.

A *partially visible function* $f(x : y)$ is a Boolean function f on a set of *visible variables* $x = x_1 \cdots x_r$, and a set of *hidden variables* $y = y_1 \cdots y_s$. Two partially visible functions $f(x : y)$ and $g(x : z)$ on the same set of visible variables are *equivalent* with respect to distributions D and E on the domains of y and z if, for all x, $\mathbf{Pr}[f(x : Y) = 1] = \mathbf{Pr}[g(x : Z) = 1]$, where Y and Z are random variables representing a random assignment to y and z according to D and E. In the discussion that follows, we will only be interested in uniform distributions.

As described above, our algorithm regards variables that occur deep in the target formula as hidden variables. The next two lemmas show that two partially visible read-once majority formulas that are identical, except for some deep hidden variables, are very likely to produce the same output on randomly chosen inputs.

Lemma 7.1 *Let f be a read-once majority formula on n variables. Let t be the level of x_n in f. Let $X_1, \ldots, X_{n-1}$, Y and Z be independent Bernoulli variables, each* 1 *with probability* $1/2$. *Then* $\mathbf{Pr}[f(X_1, \ldots, X_{n-1}, Y) \neq f(X_1, \ldots, X_{n-1}, Z)] = 2^{-t-1}$.

Proof: By induction on t. If $t = 0$, then f is the function x_n and since $\mathbf{Pr}[Y \neq Z] = 1/2$, the lemma holds. If $t > 0$, then let $f = \text{MAJ}(f_1, f_2, f_3)$, and suppose that f_1 is the subformula in which x_n occurs. Since x_n does not also occur in f_2 or f_3, we will regard these as functions only on the remaining $n-1$ variables. It is not hard to see then that $f(X_1,\ldots,X_{n-1},Y) \neq f(X_1,\ldots,X_{n-1},Z)$ if and only if $f_2(X_1,\ldots,X_{n-1}) \neq f_3(X_1,\ldots,X_{n-1})$ and $f_1(X_1,\ldots,X_{n-1},Y) \neq f_1(X_1,\ldots,X_{n-1},Z)$. Since f_2 and f_3 each output 1 independently with probability 1/2, we have

$$\mathbf{Pr}[f_2(X_1,\ldots,X_{n-1}) \neq f_3(X_1,\ldots,X_{n-1})] = 1/2.$$

Also, by inductive hypothesis,

$$\mathbf{Pr}[f_1(X_1,\ldots,X_{n-1},Y) \neq f_1(X_1,\ldots,X_{n-1},Z)] = 2^{-t}.$$

The lemma then follows by independence. ∎

Lemma 7.2 *Let f be a read-once majority formula on n variables. Let t_i be the level of variable x_i in f. Let $X_1,\ldots,X_n,X_1',\ldots,X_r'$, $r \leq n$, be independent Bernoulli variables, each* 1 *with probability* 1/2. *Then* $\mathbf{Pr}[f(X_1,\ldots,X_n) \neq f(X_1',\ldots,X_r',X_{r+1},\ldots,X_n)] \leq \sum_{i=1}^{r} 2^{-t_i-1}$.

Proof: By induction on r. If $r=0$, then the lemma holds trivially. For $r>0$, we have

$$\begin{aligned}
&\mathbf{Pr}[f(X_1,\ldots,X_n) \neq f(X_1',\ldots,X_r',X_{r+1},\ldots,X_n)] \\
&\leq \mathbf{Pr}[f(X_1,\ldots,X_n) \neq f(X_1',\ldots,X_{r-1}',X_r,\ldots,X_n)] \\
&\quad +\mathbf{Pr}[f(X_1',\ldots,X_{r-1}',X_r,\ldots,X_n) \neq f(X_1',\ldots,X_r',X_{r+1},\ldots,X_n)] \\
&\leq \sum_{i=1}^{r-1} 2^{-t_i-1} + 2^{-t_r-1}
\end{aligned}$$

where the last inequality follows from our inductive hypothesis and the preceding lemma. ∎

As proved below, Lemma 7.2 implies that any partially visible formula is an ϵ-good hypothesis if it is equivalent to the truncated target, the partially visible formula obtained from the target formula by regarding all variables at or below level $h = \lceil \log(n/2\epsilon) \rceil$ as hidden variables. Given an assignment to the visible variables, such a hypothesis is evaluated in the obvious manner by choosing a random assignment to the hidden variables and computing the output of the formula on the combined assignments to the hidden and visible variables. (Thus, the hypothesis is likely to be randomized.)

Lemma 7.3 *Let $\epsilon > 0$, and let f be a read-once majority formula on n variables. Let $x = x_1 \cdots x_r$ be the variables occurring above level $h = \lceil \log(n/2\epsilon) \rceil$, and let $y = y_1 \cdots y_{n-r}$ be the remaining variables. Let $g(x:z)$ be any partially visible formula equivalent to the partially visible formula $f(x:y)$. Then $Pr[f(X:Y) \neq g(X:Z)] \leq \epsilon$, where X, Y and Z are random variables representing the uniformly random choice of assignments to x, y and z. That is, $g(x:z)$ is an ϵ-good hypothesis for f.*

Proof: Let Y' be a random variable representing a random assignment to y, chosen independently of Y. Since $f(x:y)$ is equivalent to $g(x:z)$, we have

$$\mathbf{Pr}[f(X:Y) \neq g(X:Z)] = \mathbf{Pr}[f(X:Y) \neq f(X:Y')].$$

By Lemma 7.2, the right hand side of this equation is bounded by ϵ, since each of the $n-r \leq n$ variables y_i occurs at or below level h in f. ∎

For $\epsilon > 0$ and target formula f, we will henceforth say that variables occurring above level $h = \lceil \log(n/2\epsilon) \rceil$ are *significant.* Note that Theorem 3.3 implies that the significance, sign and level of any variable x_j can be determined by hard-wiring that variable to 1, as usual. More specifically, if $\hat{\alpha}$ is an estimate of $\alpha = A_{f|x_j \leftarrow 1}(\frac{1}{2})$ for which $|\hat{\alpha} - \alpha| < \tau \leq \left(\frac{1}{2}\right)^{h+2}$ then x_j is significant if and only if $\left|\hat{\alpha} - \frac{1}{2}\right| > \left(\frac{1}{2}\right)^{h+1} + \tau$, and, if it is significant, then its sign and level can be determined as in Theorem 3.3.

Similar to Theorem 3.7, we can show that, for any triple of significant variables, we can determine the level of the gate at which the triple meets, if at all. More precisely, if x_i, x_j and x_k are three unnegated variables occurring at levels t_1, t_2 and t_3, and if $\hat{\alpha}$ is an estimate of $\alpha = A_{f|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2})$ for which $|\hat{\alpha} - \alpha| < \tau \leq 2^{-3h}$, then it follows from Lemmas 3.5 and 3.6 that x_i, x_j and x_k meet at a level-d gate if and only if

$$\left| \frac{1}{2} + \left(\frac{1}{2}\right)^{t_1+1} + \left(\frac{1}{2}\right)^{t_2+1} + \left(\frac{1}{2}\right)^{t_3+1} - \left(\frac{1}{2}\right)^{t_1+t_2+t_3-2d-1} - \hat{\alpha} \right| < \tau.$$

As mentioned above, we call the sum total of this information—the significance of each variable, the level and sign of each significant variable, and the level of the gate at which each triple of significant variables meet, if at all—the formula's *schedule.* It remains then only to show how an ϵ-good hypothesis can be constructed from the schedule. Specifically, we show how to construct a partially visible formula that is equivalent to the truncated target $f(x:y)$. (Here, x is the vector of visible (i.e., significant) variables, and y is the vector of hidden (insignificant) variables.)

Suppose first that no three visible variables meet in f. Such a formula is said to be *unstructured.* Although strictly speaking f cannot be unstructured (by our choice of h), this special case turns out nevertheless to be important in handling the more general case since *subformulas* of f may be unstructured.

Lemma 7.4 below shows that an unstructured formula $f(x:y)$ is equivalent to any other unstructured partially visible formula whenever each visible variable occurs at the same level with the same sign in both formulas. Thus, unstructured formulas are not changed when visible variables are moved around within the same level. This fact makes the identification of unstructured formulas from their schedules quite easy.

For any partially visible read-once majority formula $f(x:y)$, let $p_f(x) = \mathbf{Pr}[f(x:Y) = 1]$ where Y represents a random assignment to y.

Lemma 7.4 *Let $f(x:y)$ and $g(x:z)$ be unstructured read-once majority formulas on s visible variables. Suppose that each visible variable x_j is relevant and occurs at the same level t_j with the same sign in both formulas. Then the two partially visible formulas are equivalent.*

Proof: It suffices to prove the lemma when no visible variable is negated since negated variables can simply be replaced by unnegated meta-variables.

To prove the lemma, we show that

$$p_f(x) = \tfrac{1}{2} + \sum_{i=1}^{s} 2^{-t_i}(x_i - \tfrac{1}{2}). \tag{1}$$

Since this statement applies to any unstructured formula, it follows immediately that $p_f(x) = p_g(x)$ and the two partially visible formulas are equivalent.

We prove Equation (1) by induction on the height h of f. If $h = 0$, then f consists of a single visible or hidden variable. If f is the formula x_j, where x_j is some visible variable, then $p_f(x) = x_j$, satisfying (1). If f is the formula y_j, where y_j is a hidden variable, then $p_f(x) = \frac{1}{2}$, also satisfying (1).

If $h > 0$, then let $f = \text{MAJ}(f_1, f_2, f_3)$ where f_1, f_2 and f_3 are partially visible subformulas. Since f is unstructured, one of these (say f_3) contains no visible variables, and thus $p_{f_3}(x) = \frac{1}{2}$. Suppose without loss of generality that $x_1, \ldots, x_r$ are the visible variables relevant to f_1. Then, by inductive hypothesis,

$$p_{f_1}(x) = \tfrac{1}{2} + \sum_{i=1}^{r} 2^{-t_i+1}(x_i - \tfrac{1}{2})$$

and

$$p_{f_2}(x) = \tfrac{1}{2} + \sum_{i=r+1}^{s} 2^{-t_i+1}(x_i - \tfrac{1}{2}).$$

Applying Lemma 3.1, it is easily verified that Equation (1) is satisfied, completing the induction. ∎

Thus, if $f(x:y)$ is unstructured, then an equivalent unstructured formula can be constructed from f's schedule. For instance, here is an efficient algorithm: Let t be the depth of the deepest visible variable in f. Break the set of all level-t variables into pairs. Replace each such pair x_i, x_j by a level-$(t-1)$ visible meta-variable $w \equiv \text{MAJ}(x_i, x_j, y)$, where y is a new hidden variable. If an odd level-t variable x_i remains, replace it with a level-$(t-1)$ visible meta-variable $w \equiv \text{MAJ}(x_i, y, y')$, where y and y' are new hidden variables. Repeat for levels $t-1, t-2, \ldots, 1$. It is not hard to show that this algorithm results in a formula that is unstructured, and that is consistent with f's schedule (and so is equivalent).

With these tools in hand for dealing with unstructured formulas, we are now ready to describe an algorithm for handling the general case, i.e., for reconstructing any (not necessarily unstructured) formula from its schedule.

Let $f(x:y)$ be the truncated target. If f is unstructured, then the previous algorithm applies. Otherwise, we can find from the schedule three visible variables x_i, x_j and x_k that meet at some maximum-depth gate λ of f; that is, they meet at a level-d gate, and no triple of visible variables meet at any gate of depth exceeding d. Then the subformula g subsumed by λ computes the majority of three subformulas g_1, g_2 and g_3, each containing one of x_i, x_j and x_k (say, in that order). Let x_ℓ be some other visible variable. Then it is easily verified that x_ℓ is relevant to g_1 if and only if x_ℓ, x_j and x_k meet at a level-d gate (namely, λ). Thus, all of the visible variables relevant to g_1 (and likewise for g_2 and g_3) can be determined from the schedule. Moreover, note that each of these subformulas is unstructured since λ is of maximum depth. Thus, each subformula can be identified using the previous algorithm for unstructured formulas, and therefore, the entire subformula subsumed by (and including) λ can be identified.

The rest of the formula can be identified recursively: we replace subformula g by a new meta-variable w, and update the schedule appropriately.

This completes the description of the algorithm. We thus have:

Theorem 7.5 *There exists an algorithm with the following properties: Given n, $\delta > 0$, and access to examples drawn from the uniform distribution on $\{0,1\}^n$ and labeled by any read-once majority formula f on n variables, the algorithm outputs an ϵ-good hypothesis for f with probability at least $1-\delta$. The algorithm's sample complexity is $O((n/\epsilon)^6 \cdot \log(n/\delta))$, and its time complexity is $O((n^9/\epsilon^6) \cdot \log(n/\delta))$.*

Proof: As before, let $h = \lceil \log(n/2\epsilon) \rceil$. To implement the procedure outlined above, we must be able to compute from a random sample: (1) each variable's significance, (2) each significant variable's sign and level, and (3) the level of the gate (if any) at which each triple of significant variables meet. As noted above, to compute (1) and (2), we need to find, for each variable x_i, an estimate $\hat{\alpha}$ of $\alpha = A_{f|x_i \leftarrow 1}(\frac{1}{2})$ for which $|\alpha - \hat{\alpha}| < \left(\frac{1}{2}\right)^{h+2}$. Hoeffding's inequality (Lemma 3.9) implies that such estimates can be derived, with high probability, from a sample of size $O(2^{2h}\log(n/\delta))$. We also noted above that we can compute (3) given, for each triple of significant variables x_i, x_j, x_k, an estimate $\hat{\alpha}$ of $\alpha = A_{f|x_i,x_j,x_k \leftarrow 1}(\frac{1}{2})$ for which $|\hat{\alpha} - \alpha| < 2^{-3h}$. Again applying Hoeffding's inequality, we see that such estimates can be computed (with high probability) for all triples of variables from a sample of size $O(2^{6h}\log(n/\delta))$. This proves the stated sample size.

The running time of the procedure is dominated by the computation from the sample of the $O(n^3)$ estimates described above. ■

8 Summary

In this paper, we have described a general technique for inferring the structure of read-once formulas over various bases. We have shown how this technique can be applied to achieve exact identification when the formula is of logarithmic depth, even in the presence of noise. We have also described how to use this technique to infer good approximations of formulas with unbounded depth.

9 Acknowledgements

We are very grateful to Ron Rivest for his comments on this material, and for a careful reading he made of an earlier draft. We also thank Avrim Blum for suggesting the notion of a universal identification sequence. Finally, we thank the anonymous referees for their comments.

References

[1] Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning read-once formulas with queries. Technical Report UCB/CSD 89/528, University of California Berkeley, Computer Science Division, August 1989. To appear, *Journal of the Association for Computing Machinery*.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[3] Gyora M. Benedek and Alon Itai. Learnability by fixed distributions. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 80–90, August 1988.

[4] Ravi B. Boppana. Amplification of probabilistic Boolean formulas. In *26th Annual Symposium on Foundations of Computer Science*, pages 20–29, October 1985.

[5] Ravi Babu Boppana. *Lower Bounds for Monotone Circuits and Formulas*. PhD thesis, Massachusetts Institute of Technology, 1986.

[6] N. Bshouty, T. Hancock, L. Hellerstein, and M. Karpinski. Read-once formulas, justifying assignments, and generic transformations. Unpublished Manuscript, 1991.

[7] Merrick Furst, Jeffrey Jackson, and Sean Smith. Learning AC^0 functions sampled under mutually independent distributions. Technical Report CMU-CS-90-183, Carnegie Mellon University, School of Computer Science, October 1990.

[8] Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 303–314, August 1991.

[9] Thomas Hancock and Lisa Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 326–336, August 1991.

[10] Thomas Hancock and Yishay Mansour. Learning monotone $k\mu$ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 179–183, August 1991.

[11] Lisa Hellerstein. *On Characterizing and Learning Some Classes of Read-Once Formulas.* PhD thesis, University of California at Berkeley, 1989.

[12] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[13] Michael Kearns. *The Computational Complexity of Machine Learning.* MIT Press, 1990.

[14] Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 267–280, May 1988.

[15] Michael Kearns, Ming Li, Leonard Pitt, and Leslie Valiant. On the learnability of Boolean formulae. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 285–295, May 1987.

[16] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 433–444, May 1989.

[17] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science*, pages 574–579, October 1989.

[18] Giulia Pagallo and David Haussler. A greedy method for learning μDNF functions under the uniform distribution. Technical Report UCSC-CRL-89-12, University of California Santa Cruz, Computer Research Laboratory, June 1989.

[19] Leonard Pitt and Manfred K. Warmuth. Prediction-preserving reducibility. *Journal of Computer and System Sciences*, 41(3):430–467, December 1990.

[20] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[21] Robert E. Schapire. Learning probabilistic read-once formulas on product distributions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 184–198, August 1991.

[22] Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Computing*, 8:337–247, 1991.

[23] Robert H. Sloan. Types of noise in data for concept learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 91–96, August 1988.

[24] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[25] Karsten Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, August 1990.