New Results on Hardness of Proper Learning and Beyond

Misha Alekhnovich *
IAS, Princeton
misha@math.ias.edu

Mark Braverman†
U. Toronto
mbraverm@cs.toronto.edu

Vitaly Feldman‡
Harvard University
vitaly@eecs.harvard.edu

Adam R. Klivans§
Toyota Technological Institute
klivans@tti-c.org

Toniann Pitassi¶
IAS, Princeton and U. Toronto
toni@cs.toronto.edu

February 7, 2005

Abstract

We consider the complexity of properly learning concept classes, i.e. when the learner must output a hypothesis of the same form as the unknown concept. We present the following new upper and lower bounds on well-known concept classes:

- We show that unless $\mathsf{NP} = \mathsf{RP}$, there is no polynomial-time PAC learning algorithm for DNF formulae where the hypothesis is an OR-of-thresholds. Note that as special cases, we show that neither DNF nor OR-of-thresholds are properly learnable unless $\mathsf{NP} = \mathsf{RP}$. Previous hardness results have required strong restrictions on the size of the output DNF formula. We also prove that it is NP-hard to learn the intersection of $\ell \geq 2$ halfspaces by the intersection of k halfspaces for any constant $k \geq 0$. Previous work held for the case when $k = \ell$.
- Assuming that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{n^\lambda})$ for a certain constant $\lambda < 1$ we show that it is not possible to learn size s decision trees by size s^k decision trees for any $k \geq 0$. Previous hardness results for learning decision trees held for $k \leq 2$. We also show that under the same complexity assumption it is not possible to learn junta functions properly in polynomial time.
- We present the first non-trivial upper bounds on properly learning DNF formulae. In particular we show how to learn size s DNF by DNF in time $2^{\tilde{O}(\sqrt{n \log s})}$.

The hardness results for DNF formulae and intersections of halfspaces are obtained via specialized graph products for amplifying the hardness of approximating the chromatic number as well as applying recent work on the hardness of approximate hypergraph coloring. The hardness results for decision trees, as well as the new upper bounds, are obtained by developing a connection between automatizability in proof complexity and learnability, which may have other applications.

*Supported by CCR grant $\mathcal{N}$CCR-0324906.

†Supported by an NSERC Postgraduate Scholarship

‡Supported by NSF grant CCR-98-77049.

§Work done at Harvard University and supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

¶Supported by NSERC and PREA research grants.

1 Introduction

A fundamental goal of computational learning theory is to establish hardness results for PAC learning concept classes. Seminal work due to Kearns and Valiant [KV94] has shown that under the assumption that certain cryptographic primitives are computationally intractable (e.g. inverting one-way functions), there are no polynomial-time learning algorithms for concept classes which are expressive enough to compute pseudorandom-functions. Subsequent work [Kha93, NR97, JKS02] has shown that even constant depth, polynomial size circuits (often referred to as AC^0) are capable of computing pseudo-random objects and are unlikely to be learnable in polynomial time.

Still, several well-studied concept classes seem too weak to compute cryptographic primitives, such as polynomial-size DNF formulae, intersections of halfspaces, and decision trees. For all of these concept classes the existence of a polynomial-time PAC learning algorithm remains a challenging open problem. The primary contribution of this work is an array of new negative results for learning DNF formulae, intersections of halfspaces, and decision trees. Our hardness results apply to *representation dependent* learning algorithms, algorithms where the output hypothesis is required to be a member of a restricted class of polynomial-time computable functions.

1.1 Previous Work

Previous representation dependent hardness results for learning concept classes applied to *proper* learning algorithms and required strong restrictions on the *size* of the hypothesis output by the learning algorithm [BR92, Gol78, KLPV87, PV88, NJS98]. In each case, the hardness assumption required is not cryptographic, but a worst-case assumption on the complexity of NP (e.g. $\mathsf{NP} \neq \mathsf{RP}$).

Initial hardness results for properly learning DNF formulae due to Pitt and Valiant [PV88] show that unless $\mathsf{RP} = \mathsf{NP}$, k-term DNF formulae over n variables are not learnable by $2k$-term DNF (more specifically the result holds for the case when $k = \Theta(n)$). In other words, it is hard to output a DNF formula whose size is at most twice the size of an unknown DNF formula with $\Omega(n)$ terms. The best result along these lines is due to Nock et al. [NJS98] who have used reductions from generalized coloring problems to show that it is hard to output a DNF formula whose size is at most $O(k^a n^b)$ times the size of the unknown k-term DNF formula for $a \leq 2, b \geq 0$ and $k = \Omega(n^\gamma)$ for any $\gamma > 0$. For $k = O(1)$ the best hardness result is due to Pitt and Valiant [PV88] who show that learning k-term DNF is hard if the output hypothesis is a k-term DNF.

The best hardness result for learning intersections of halfspaces is due to Blum and Rivest [BR92]; implicit in their work (when combined with the hardness results on chromatic number due to Feige and Kilian [FK96]) is a proof that unless $\mathsf{NP} = \mathsf{RP}$ it is hard to learn the intersection of n^γ halfspaces by $n^{1-\gamma}$ halfspaces for any $\gamma > 0$; n is the number of variables (i.e. number of dimensions). For the case of intersections of $k = O(1)$ halfspaces they show the associated learning problem is hard if the output is equal to the intersection of k halfspaces.

For decision trees, Hancock et al. [HJLT95] have shown that it is hard to learn size s decision trees over n variables by size $s \cdot 2^{\log^{1-\gamma} s}$ decision trees for some $\gamma > 0$ unless $\mathsf{NP} \subseteq \mathsf{RTIME}(2^{\log^{O(1)} n})$. The result holds for $s = \Omega(n)$.

We note here that the above hardness results hold for proper *Occam* algorithms, learning algorithms which work by receiving a suitably large set of training examples and outputting a small hypothesis consistent with the examples. It is not known, in general, if the existence of a proper PAC learning algorithm for a concept class implies the existence of a proper Occam algorithm for the class[PB90]. In particular is not known for the classes of DNF formulae and intersections of halfspaces. Our hardness results for DNF formulae and intersections of halfspaces hold for any proper PAC learning algorithm and overcome this limitation.

Several results are known for the hardness of learning DNF in the exact model with membership and equivalence queries (see Hellerstein and Raghavan [HR02] for details).

1.2 Our Results

We provide new hardness results on the complexity of learning DNF formulae, intersections of halfspaces, decision trees, and juntas which place far fewer restrictions on the size and form of the learning algorithm's output hypothesis. We complement these hardness results by describing new algorithms for achieving non-trivial upper bounds on the proper learnability of DNF formulas and decision trees.

1.2.1 Upper Bounds

By making a connection between proper learning and the automatizability of certain propositional proof systems, we give the first non-trivial upper bounds on the complexity of properly learning DNF formulae (we also show how our approach yields Ehrenfeucht and Haussler's result [EH89] for learning decision trees):

Theorem 1 *Decision trees of size s are properly learnable in time $n^{O(\log s)}$. DNF formulae of size s are properly learnable in time $n^{O(\sqrt{n \log s})}$.*

The above upper bound for learning decision trees is originally due to Ehrenfeucht and Haussler [EH89]. The above $2^{\tilde{O}(\sqrt{n})}$ upper bound for properly learning polynomial-size DNF formulas is new and should be contrasted with the $2^{\tilde{O}(n^{1/3})}$-time algorithm for learning DNF due to Klivans and Servedio [KS01]; theirs is the fastest known DNF learning algorithm but does not output a DNF formula as a hypothesis.

1.2.2 Hardness for Learning Decision Trees and Juntas

The parameterized minimum hitting set problem, takes as input a system S of m subsets of $[n]$ and a parameter k, and should output a hitting set of size k for S if one exists. The parameter k is supposed to be a slowly growing function w.r.t. n (like $k = \log n$). This problem is complete for the class W[2] of the parameterized hierarchy ([DF98]).

Theorem 2 *Assume that no randomized algorithm approximates parameterized minimum hitting set to within a factor d in polynomial time, for $k = O(\log n)$ and any constant d. Then there is no algorithm $\mathcal{A}$ such that for every function c with a minimal decision tree representation of size s, distribution $\mathcal{D}$ and error parameter ϵ, $\mathcal{A}$ runs in time $poly(n, s, 1/\epsilon)$ and with probability $2/3$ outputs a decision tree T such that $Pr_{x \in \mathcal{D}}[T(x) = c(x)] \geq 1 - \epsilon$.*

The above theorem combined with a result of [AKP04] implies the following theorem.

Theorem 3 *Decision trees are not properly PAC learnable in polynomial time unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{n^\lambda})$ for some $\lambda < 1$.*

We also show that hardness of approximating the parameterized minimum hitting set problem implies the hardness for proper learning juntas, functions which depend on only a small subset of relevant variables (see Section 4).

1.2.3 Hardness for Learning DNF Formulae

Learning DNF formulae is one of the central challenges in computational learning theory. We give strong evidence that there are no polynomial-time learning algorithms for DNF formulae which output DNF formulae or unions of halfspaces as output hypotheses:

Theorem 4 *If there exists an algorithm $\mathcal{A}$ such that for every function c with a minimal DNF representation of size s, distribution $\mathcal{D}$ and error parameter ϵ, $\mathcal{A}$ runs in time $poly(n, s, 1/\epsilon)$ and with probability $2/3$ outputs an OR-of-thresholds formula f such that if $Pr_{x \in \mathcal{D}}[f(x) = c(x)] \geq 1 - \epsilon$, then* $\mathsf{NP} = \mathsf{RP}$.

1.2.4 Hardness for Learning Intersections of Halfspaces

Let $h = \mathsf{sign}(\sum_{i=1}^{n} w_i x_i - \theta)$ where each w_i and θ are integers; h naturally induces two halfspaces: the set of points in $\{0,1\}^n$ which make h positive and the set of points which make h negative (h is often referred to as a linear threshold function). Although several polynomial-time algorithms for learning halfspaces are known (e.g. [BEHW87]), a longstanding open problem in learning theory is to develop polynomial-time algorithms for learning *intersections* of halfspaces (i.e. functions of the form $h = \wedge_{i=1}^{k} h_i$ where each h_i is a linear threshold function).

The above theorem proves as a special case that intersections of halfspaces are not properly learnable unless $\mathsf{NP} = \mathsf{RP}$. If we wish to restrict the concept class to intersections of just two halfspaces (even for this case no polynomial-time learning algorithms are known), we can prove the following hardness result:

Theorem 5 *Let $\mathcal{C}$ be the concept class of intersections of two halfspaces. If there exists an algorithm $\mathcal{A}$ such that for every $c \in \mathcal{C}$, distribution $\mathcal{D}$ and error parameter ϵ, $\mathcal{A}$ runs in time $poly(n, 1/\epsilon)$ and with probability $2/3$ outputs f, an intersection of k halfspaces for any constant k such that $Pr_{x \in \mathcal{D}}[f(x) = c(x)] \geq 1 - \epsilon$, then* $\mathsf{NP} = \mathsf{RP}$.

1.3 Our Approach

Our techniques can be divided into two categories: 1) negative results based on the intractability of approximate graph and hypergraph coloring and 2) positive and negative results obtained by establishing a connection between automatizability of propositional proof systems and proper learnability.

1.3.1 Amplifying Hardness Results for Approximate Graph Coloring

For proving hardness results for properly learning DNF and intersections of halfspaces we amplify known hardness results for the problem of distinguishing between graphs with small and large chromatic number. Feige and Kilian [FK96] have proved that for any $\gamma > 0$ it is NP-hard (under randomized reductions) to distinguish between graphs with chromatic number $O(n^{\gamma})$ and graphs with chromatic number $\Omega(n^{1-\gamma})$. This result combined with known reductions from graph coloring to properly learning DNF formulae (e.g. [PV88]) imply that it is NP-hard to distinguish between distributions induced by n^{γ}-term DNF formulae and $n^{1-\gamma}$-term DNF formula.

We wish to amplify this $n^{1-\gamma}$ bound and prove hardness results for n^a-term DNF formulae (and intersections of n^a halfspaces) for any $a \geq 0$. To do this we apply specialized graph products (along the lines of Linial and Vazirani [LV89]) to create distributions which amplify the size of the underlying chromatic number. In addition, we provide an accompanying transformation of DNF formulae and intersections of halfspaces into "normal forms" which satisfy only examples derived from subsets of independent sets from the product. Many terms or halfspaces are required for a good approximation to these distributions if and only if the original graph had large chromatic number.

For proving hardness results for learning the intersection of two halfspaces, we make critical use of recent hardness results due to Dinur et al. [DRS02] on the hardness of coloring 2-colorable, 3-uniform hypergraphs. We give a reduction from ℓ-coloring k-colorable, 3-uniform hypergraphs to properly learning intersections of k halfspaces by ℓ halfspaces.

1.3.2 Automatizability and Proper Learning

A propositional proof system S is said to be *automatizable* if there is an algorithm A which takes as input a CNF formula f, and returns a proof of f, in time polynomial in the size of the shortest S-proof of f. Automatizability is an important concept; while a proof system may be extremely powerful and admit short proofs of many hard statements, if it is impossible to find these proofs quickly, then for all practical purposes we are no further ahead than we were with a naive exhaustive proof method.

There are two types of automatizability for any proof system S. The first type (called automatizability) requires that the automatizing algorithm return an S-proof of f. The second type (called weak automatizability) only requires that the algorithm returns *any* polynomially-verifiable proof, and not necessarily an S-proof. Informally, we have the following relationship. Let C be a circuit class, and let $P(C)$ be a proof system which manipulates formulas from C. Three important examples are: (i) When C is the class of decision trees, the corresponding proof system is DPLL; (ii) When C is the class of DNF formulae, the corresponding proof system is Resolution, and (iii) When C is the class of intersections of threshold formulae, a corresponding proof system is Cutting Planes. Then automatizability of proof system $P(C)$ corresponds to proper PAC learning of C and weak automatizability of $P(C)$ corresponds to learnability of C. In both cases (automatizability and learnability), the desired algorithm is searching for an object over C. We will see that techniques used to obtain positive and negative results for automatizability of various proof systems can be exploited to obtain new learnability results.

2 Preliminaries

2.1 Learning models

Our learning model is Valiant's well known Probably Approximately Correct (PAC) learning model [Val84]. In this model for a concept c and distribution D over X an *example oracle* $EX(c, D)$ is an oracle that upon request returns an example $(x, c(x))$ where x is chosen randomly with respect to D independently of any previous examples. For $\epsilon \geq 0$ we say that function g ϵ-approximates function f with respect to distribution D if $\mathbf{Pr}_D[f(x) = g(x)] \geq 1 - \epsilon$. We say that an algorithm $\mathcal{A}$ efficiently learns concept class C if for every $\epsilon > 0$, $\delta > 0$, n, $c \in \mathcal{C}$, and distribution D_n over X_n, $\mathcal{A}(n, \epsilon, \delta)$, runs in time polynomial in n, $1/\delta$, $1/\epsilon$, $|c|$ and outputs, with probability at least $1 - \delta$, an efficiently computable hypothesis h from some class of functions H that ϵ-approximates c. When $H = C$ the algorithm $\mathcal{A}$ is a *proper* PAC learning algorithm. Frequently we will prove hardness results for cases where H is actually a larger class of functions than C; such results are thus stronger than traditional hardness results for proper learnability.

2.2 Propositional Proof Complexity

The *resolution principle* says that if C and D are clauses and x is a variable, then any assignment that satisfies both of the clauses $C \vee x$ and $D \vee \neg\, x$ also satisfies $C \vee D$. A *resolution refutation* for a CNF formula F consists of a sequence of clauses $C_1, C_2, \ldots, C_s$ where (i) each clause C_i is either a clause of F, or is a resolvent of two previous clauses and (ii) C_s is the empty clause, denoted Λ. A tree-like Resolution refutation is a Resolution refutation where the underlying directed acyclic graph is a tree. A DPLL refutation of an unsatisfiable formula F is a decision tree for f with the additional property that for every path p in the decision tree and corresponding partial truth assignment ρ, the leaf of p is labelled by a clause in f that is falsified by ρ. It is well known that tree-like Resolution refutations and DPLL refutations are equivalent. The *automatizability problem* for proof systems, formalized in [BPR97], is to find effective algorithms for constructing refutations whose size is close to optimal:

Definition 6 *For a propositional proof system $\mathcal{S}$, let $s(F)$ denote the size of the smallest refutation of formula F in $\mathcal{S}$. $\mathcal{S}$ is automatizable if there exists an algorithm that on input F (on n variables and m clauses), outputs an $\mathcal{S}$-refutation of f in time polynomial in $s(F)$ and n and m. More generally $\mathcal{S}$ is $q(s,n,m)$-automatizable if there exists an algorithm that runs in time $q(s(F),n,m)$ and outputs an $\mathcal{S}$-refutation of F.*

3 Upper bounds for Properly Learning Decision Trees and DNF

In [BKPS98] (see also [BSW99, CEI96]), algorithms were presented for automatizability of DPLL and Resolution. In this section, we will show how these algorithms can be used to prove Theorem 1. We first present a proof of our theorem and then discuss how it can be viewed as a modification of the algorithm presented in [BKPS98].

Proof: (of Theorem 1)

Let $\mathcal{P}$ be a DNF formula. $\mathcal{P}$ is *b-bounded* if all terms appearing in it have size at most b. Fix ϵ, δ, n, and s. The algorithm will begin by obtaining a set S of m labelled examples chosen at random according to the underlying distribution D. (The value of m will be chosen later.) The algorithm will then produce a hypothesis consistent with S. Then using a standard argument, it can be shown that any algorithm that produces a hypothesis from a relatively small set of hypotheses, that is consistent with a set of m examples (m sufficiently large), will also satisfy the requirements of PAC learning, with high probability.

First, we need a subroutine, called **Bounded-search**, which takes as input a set of labelled examples over n variables, S, $|S| = m$, and an integer parameter b, and finds a b-bounded DNF consistent with S, if one exists. The subroutine works by learning a single disjunction over a new set of n^b variables (each variable corresponds to one of the n^b different terms of the unknown DNF of length b). It is well-known that disjunctions over N variables can be learned in time $O(N)$ using $O(N/\epsilon)$ examples. The output of the subroutine can be converted to a DNF with at most n^b terms. In our context, this subroutine runs in time $T_0(n,m,b) = O(n^b + m)$.

The main algorithm called **Search** takes as input a set of m labelled examples over n variables, S, and an auxiliary parameter b. The output of **Search** will be a decision tree with the leaves of the tree labelled by b-bounded DNF formulas. The algorithm is as follows. First, we use **Bounded-search**(S,b) to find a b-bounded DNF formula consistent with S if one exists. If not, then for each of the $2n$ literals l, apply **Search** to the set of labelled examples $S\lceil_{l=1}$, in order to identify the literal l for which **Search**$(S\lceil_{l=1})$ terminates fastest. These $2n$ calls to **Search** are executed in a sequence of parallel rounds; in round i the ith step of each of the $2n$ calls is performed. As soon as the first of the calls terminates, say for literal l^*, all of the other calls are aborted, except the call corresponding to $\neg\ l^*$, which is run to completion. The output of **Search**(S,b) is a decision tree where the leaves of the decision tree are labelled with b-bounded DNF's, the root is labelled by l^*, and the left subtree is a hypothesis consistent with the samples $S\lceil_{l^*=0}$, and the right subtree is a hypothesis consistent with the samples $S\lceil_{l^*=1}$. The analysis of the algorithm will rely on the following technical fact, which is easily proved by induction:

Proposition 7 *Suppose that $T(n, s)$ is a function defined for nonnegative integers n and $s > 0$ that satisfies, for some positive increasing function $h(n)$, positive constant C and $\lambda > 1$:*

$$\begin{array}{lcll} T(0,s) & \leq & h(0) & \\ T(n,s) & \leq & h(n) & \text{if } s \leq 1 \\ T(n,s) & \leq & h(n) + CnT(n-1, s\lambda) + T(n-1,s) & \text{if } n \geq 1 \text{ and } s > 1 \end{array}$$

Then $T(n, s) \leq h(n)(1 + C^{\log_\lambda s} n^{2\log_\lambda s})$.

To prove the first part of the theorem, set $b = 0$, and set $m = (n^{O(\log s)} + \log(1/\delta))/\epsilon$. Since $b = 0$, the output by **Search** will be an ordinary decision tree. We upper bound the running time of the algorithm in terms of s. Let $T_1(n, s; m)$ denote the maximum running time of **Search**$(S, 0)$ over all sets of m labelled examples S, with n underlying variables and such that there is a decision tree consistent with S of size at most s. Let x_i be the splitting variable at the root. The left and right branches of the tree give refutations for $S\lceil_{x_i}$ and $S\lceil_{\neg x_i}$, and the smaller of these is of size at most $s/2$. Hence at least one of the recursive calls terminates after at most $T_1(n - 1, s/2; m)$ steps, and so the literal l^* is found after at most that number of rounds. The time for each round can be bounded above by Cn for some constant C. Once l^* is found, it takes at most $T_1(n - 1, s; m)$ steps to complete the call to **Search**$(S\lceil_{\neg\, l^*})$. Thus, we conclude that $T_1(n, s; m)$ satisfies the recurrence of the above Proposition with $h(n) = T_0(n, m, b)$ and $\lambda = 2$. We conclude that $T_1(n, s; m) = n^{O(\log s)}O(m)$. The algorithm thus produces a hypothesis of size $n^{O(\log s)}$ in time $n^{O(\log s)}O(m)$ that is consistent with the m samples. The probability that any given hypothesis has error more than ϵ but is consistent with all m samples is at most $(1 - \epsilon)^m$. Summing up over all hypotheses of this size, the overall probability that we output a hypothesis consistent with all m samples, but with overall error at least ϵ (with respect to D) is at most δ.

For the second result, let $m = (n^{\sqrt{n \log s}} + \log(1/\delta))/\epsilon$. For a DNF formula F, let $F[b]$ be the subset of terms of F of size greater than b. For a set of labelled examples S, let $DNF(S, b)$ denote the minimum over all DNF formulas F consistent with S, of $|F[b]|$ (so that for $b < 0$, $DNF(S) = DNF(S, b)$). Let $T_2(n, s; m, b)$ denote the maximum time needed by **Search**(S, b) on sets of labelled examples S over n variables, such that $|S| = m$ and $DNF(S, b) \leq s$. Note that $T_2(n, s; m, b) \leq T_0(n, b)$ if $s < 1$ and $T_2(0, s; m, b) = O(1)$. Suppose n and s are both at least 1. Let S be a set of labelled examples over n variables, $|S| = m$ and let F be a DNF consistent with S such that $F[b] \leq s$. For a literal l, let $c(b, l)$ be the number of terms of $F[b]$ containing l. The average of $c(b, l)$ over literals is greater than $|F[b]|b/2n$ and hence there exists a literal l with $c(b, l) > b|F[b]|/2n$. Note that the DNF formula $F\lceil_{l=0}$ for $S\lceil_{l=0}$ has at most $|F[b]|(1 - \frac{b}{2n})$ terms, and hence $T_2(n, s; m, b)$ satisfies the recurrence for T in the proposition with $\lambda = \frac{2n}{2n-b}$ and $h(n) = T_0(n, b)$. Applying the proposition, we conclude that $T_2(n, b) \leq T_0(n, b)n^{O(\frac{n}{b}\log s)}$. Choosing $b = \sqrt{n \log s}$, yields an upper bound of $n^{O(\sqrt{n \log s})} \cdot m$, to complete the proof of the theorem. Recall that the output of **Search** is a decision tree with leaves labelled by DNF formulas. Such a hypothesis is a DNF formula itself, consistent with all of S, and of size $n^{\sqrt{n \log s}}$. Again by our choice of m and the union bound, the probability that there is a hypothesis of this size consistent with S, but with error at least ϵ with respect to D, is at most δ, and thus the second part of the theorem is proven. □

3.1 Discussion: Relationship to Previous Work

We mention here how the above algorithms are variations on results in proof complexity (e.g. [BKPS98]) used to find size s DPLL proofs in time $n^{O(\log s)}$, and size s Resolution proofs in time $n^{O(\sqrt{n \log s})}$. Let f

be an unsatisfiable CNF formula with n variables and m clauses. Modify **Search** to take as input a CNF formula f with n variables and m clauses (rather than a set of examples), and an auxiliary parameter b. The output of modified **Search** produces a decision tree with leaves of the tree labelled by width b Resolution refutations. Similarly modify **Bounded-search** to take as input an unsatisfiable CNF formula f and an integer parameter b, and finds a width b Resolution refutation for f, if one exists. Now if f has as size s DPLL refutation, run modified **Search** with $b = 0$, and if f has a size s Resolution refutation, run modified **Search** with $b = \sqrt{n \log s}$. The same analysis as above yields the automatizability algorithms for DPLL and Resolution, respectively.

It is also interesting to note that a previous algorithm for non-proper learning of DNF due to Bshouty [Bsh96] independently used an almost identical recurrence. His learning algorithm worked by proving a structural theorem about DNF (namely that DNF can be computed by high-rank decision lists) and applying an algorithm for learning a decision list. One interpretation of our results is that automatizability algorithms "construct" or make effective the underlying structural results from [Bsh96].

4 Hardness of Learning of Decision Trees and Juntas

For an unsatisfiable CNF formula f, the search problem associated with f is to find a violated clause, given a truth assignment to the variables underlying f. Because a DPLL refutation for f produces a decision tree for solving the search problem associated with f, automatizability of DPLL is strongly connected to PAC learning decision trees with a respect to a distribution induced by the search problem associated with f. In fact, many of the hardness results of this section were inspired by a paper of Alekhnovich and Razborov [AR01] on non-automatizability of Resolution and DPLL. Our hardness assumptions will center around the following problem:

Definition 8 *The Parameterized Minimum Hitting Set Problem (PMHS), with parameters n, m and k, takes as input a system of m subsets of $[n]$, $\vec{S} = (S_1, ..., S_m)$. The output is a hitting set of size k for $\vec{S}$, i.e. a set I s.t. $\forall j \; I \cap S_j \neq \emptyset$, if one exists.*

This classical optimization problem is equivalent to a more popular Set Cover problem. We added the adjective "parameterized" to stress that the parameter k is supposed to be much smaller than n (typically $k = \log n$ or smaller). This problem is complete for the class W[2] of the parameterized hierarchy [DF98].

4.1 Learning Juntas vs Approximating Minimum Hitting Set

The following construction goes along the lines in [HJLT95].

Definition 9 *Let $\vec{S} = (S_1, ..., S_m)$ be a set system. Let $D_{\vec{S}}$ be a distribution on $\{0,1\}^n$ given by $Pr_{x \sim D_{\vec{S}}}[x = 0^n] = 1/2$ and $\forall j \in [m] \; Pr_{x \sim D_{\vec{S}}}[x = \chi_{S_j}] = 1/2m$, where χ_{S_j} is the characteristic vector of S_j. Define a partial function $h_{\vec{S}} : \{0,1\}^n \to \{0,1\}$ so that*

$$\forall i \in [m] \; h_{\vec{S}}(\chi_{S_i}) = 1, \; h_{\vec{S}}(0^n) = 0.$$

Below we consider the complexity of learning the concept class of juntas. A function $h(x_1, ..., x_n)$ is said to be a k-junta iff its value is completely determined by the input values of some k variables $x_{i_1}, ..., x_{i_k}$. We represent a k-junta as an index set I of its essential coordinates and the truth table of size 2^k on these coordinates. Learning juntas has recently been studied by Mossel et al. [MOS03] who gave a time $n^{.704k}$ algorithm for learning a k-junta with respect to the uniform distribution on inputs.

Theorem 10 *Assume that it is possible to approximate PMHS within factor $f_1(k)$ in time $f_2(k)n^{O(1)}$, where f_1, f_2 are arbitrary functions. Then k-juntas are PAC learnable in time $f(k)n^{O(1)}$ for $f(k) = [f_1(k) + f_2(k)]^{O(1)}$, moreover the hypothesis produced by the algorithm is an $f_1(k)k$-junta.*

Proof: Let D be a distribution on $\{0,1\}^n$. Fix $\epsilon, \delta > 0$. Choose $m = f_1(k)n^2/(\epsilon\delta)$. Given examples from a k-junta $h(x)$ with $x \sim D$ we generate a table of m samples $(x_1, h_1), ..., (x_m, h_m)$, where $h_i = h(x_i)$. Our goal is to find an $f_1(k)k$-junta consistent with all m samples. We write the following CNF $\phi_{\{(x,h_i)\}}(y_1, ..., y_n)$:

$$\phi_{\{(x_i,h_i)\}} = \bigwedge_{h_i \neq h_j} \bigvee_{(x_i)_t \neq (x_j)_t} y_t \tag{1}$$

We claim that $\phi_{\{(x_i,h_i)\}}$ has a satisfying assignment of weight k. Indeed since h is a k-junta there exists a set of coordinates $I \subset [n]$ of size k that completely determine the value of h, thus if $h(x_i) \neq h(x_j)$ there is $k \in I$ for which $(x_i)_k \neq (x_j)_k$. If we set $y = \chi_I$ then we get a satisfying assignment for (1) of weight k. Moreover, given any satisfying assignment y of weight k' for (1) one may construct k'-junta $\hat{h}$ consistent with m samples in time $2^{k'}$. For this it is sufficient to choose a function that depends only on coordinates $I = \{i | y_i = 1\}$ consistent with m samples.

Note that CNF $\phi_{\{(x_i,h_i)\}}(y_1, ..., y_n)$ is monotone w.r.t. y_i thus we may regard it as an instance of the minimum hitting set problem, in which sets correspond to disjunctions. Given an $f_1(k)$-approximation algorithm for the latter problem one may find a hypothesis $\hat{h}$ that depends only upon $f_1(k)k$ variables consistent with all m samples. We finish the proof by the standard computation of the probability of choosing the correct hypothesis. □

Theorem 11 *Assume that no randomized algorithm approximates PMHS within factor c in time $f(k)n^{O(1)}$. Then no algorithm given examples from a k-junta $h(x)$ chosen from distribution D finds a $(1 - 1/n^{O(1)})$-approximation of h by a (ck)-junta h' in time $f(k)n^{O(1)}$.*

Proof: Assume for the sake of contradiction that there exists a learning algorithm $\mathcal{A}$ with the properties described in the statement. Consider an instance of PMHS $\vec{S}, k$. We run the algorithm $\mathcal{A}$ on $h_{\vec{S}}$ w.r.t. the distribution $D_{\vec{S}}$. Because $D_{\vec{S}}$ gives a non-negligible weight to every word in $\{0^n, \chi_{S_1}, ..., \chi_{S_m}\}$ the approximating function that depends only on ck variables ought to compute $h_{\vec{S}}$ on $D_{\vec{S}}$ exactly, thus any such function corresponds to a hitting set of size ck. □

4.2 Lower Bounds on Learnability of Decision Trees

In this section we give the proof of Theorem 2. In the above subsection we outlined the proof that the infeasibility of approximating the parameterized minimum hitting set implies that it is hard to learn a k-junta (on a special distribution) in polynomial time. This result itself implies that given access to examples from a function computable by size S decision tree it is hard to construct an approximating size $c \cdot S$ decision tree in polytime (and this argument is similar to the reduction in [HJLT95]). However we need to obtain a stronger polynomial gap for learning decision trees, thus we use a different type of amplification. We may assume w.l.o.g. (by scaling n appropriately) that $k < \log n/12$.

Definition 12 *For an instance $\vec{S}$ with parameters n, m, k of PMHS problem we build a partial function $g_{\vec{S},k}$ along with the distribution on its instances $D_{\vec{S},k}$ in the following way.*

Fix the maximal ℓ satisfying $2^{\ell k} < n$ (thus $\ell = \lfloor \log n/k \rfloor$). Let y_i^j for $i \in [n], j \in [\ell]$ be random Boolean variables chosen according to the following experiment. Choose $x = (x_1, ..., x_n)$ according to $D_{\vec{S}}$. For every $i \in [n]$ choose a tuple $y_i^1, ..., y_i^\ell$ uniformly from all tuples satisfying $\bigoplus_{j=1}^m y_i^j = x_i$. Denote by $D_{\vec{S},k}$ the resulting distribution on y_i^j. Finally let $g_{\vec{S},k}(y_1^1, ..., y_n^\ell) = h_{\vec{S}}\left(\bigoplus_{j=1}^\ell y_1^j, ..., \bigoplus_{j=1}^\ell y_n^j\right)$.

Thus $g_{\vec{S},k}$ is a function that depends upon $\lfloor n \log n/k \rfloor$ bits. In the next two theorems we show that the decision tree approximation complexity of $g_{\vec{S},k}$ on $D_{\vec{S},k}$ is tightly connected to the minimum hitting set $\gamma(\vec{S})$. These results will imply lower bounds on proper learnability of decision trees modulo the hardness of approximating the minimum hitting set.

Theorem 13 (upper bound) *Assume that $\gamma(\vec{S}) \leq k$. Then there exists a decision tree of size n that computes $g_{\vec{S},k}$ on $D_{\vec{S},k}$ with probability 1.*

Proof: If $\gamma(\vec{S}) \leq k$ then there exists a set of x-variables of size k that determine the value of $h_{\vec{S}}$ and hence there exists a set of ℓk y-variables that determine the value of $g_{\vec{S},k}$. Consider the decision tree that branches upon all $2^{\ell k}$ different assignments to these variables and outputs the value according to that of $h_{\vec{S}}(x)$. This decision trees correctly computes $g_{\vec{S},k}$ on all inputs and by our choice of ℓ, the size of the tree is $2^{\ell k} < n$. □

Theorem 14 (lower bound) *If $\gamma(\vec{S}) > ck$ then any decision tree T that approximates $g_{\vec{S},k}$ with error less than $1/(5m)$ has size $n^{c(1-o(1))}$.*

Proof:

Fix a decision tree T that contains less than $n^{c(1-11/\ell)}$ nodes. We will prove that it has a non-negligible error in computing $g_{\vec{S},k}$. Define a random restriction ρ on y_i^j in the following way. For every $i \in [n]$ choose a random index $v_i \in [\ell]$ and set values to the variables $y_i^1, ..., y_i^{v_i-1}, {y_i}^{v_i+1}, ..., y_i^\ell$ independently at random. Thus ρ is a random restriction that sets all but n variables.

Lemma 15 *Let t be a term over y_i^j of size w. Then*

$$Pr[t|_\rho \neq 0] \leq 2^{-w(1-10/\ell)}.$$

Proof: Denote by t_i the subterm of t that includes all literals in variables $y_i^1, ..., y_i^\ell$ in t, let $w_i = |t_i|$. Since the restrictions on y_i^j are independent for different i and $w = \sum_i w_i$ it is sufficient to prove the lemma for every single t_i. The probability that t_i is not mapped to 0 by ρ is equal to

$$\frac{\ell - w_i}{\ell} \cdot 2^{-w_i} + \frac{w_i}{\ell} \cdot 2^{-(w_i-1)} = (1 + w_i/\ell) \cdot 2^{-w_i} < 2^{-w_i(1-10/\ell)}.$$

□

Corollary 16 *For every computational path π of length w in the tree T the probability that π is chosen on input $\vec{y}$ is less than $2^{-w(1-10/\ell)}$.*

Proof: We describe a different experiment generating the values for y_i^j that leads to the same distribution $D_{\vec{S},k}$. First we pick a random restriction ρ as described above, it assigns $n(\ell-1)$ variables. Next we choose a vector $x \in \{0,1\}^n$ according to $D_{\vec{S}}$. Finally for every i we choose the remaining coordinate not assigned by ρ so that $\bigoplus_{j=1}^{\ell} y_i^j = x_i$. Define for the path π a term t_π that consists of all literals assigned along the path. If t_v is unsatisfied by ρ then the path is not chosen. $\square$

Denote by $T(\vec{y})$ the Boolean function computed by T. We may write

$$\Pr[T(y) \neq g_{\vec{S},k}(y)] = \sum_v \Pr[T(y) \neq g_{\vec{S},k}(y) | \pi_v \text{ is chosen}] \cdot \Pr[\pi_v \text{ is chosen}], \tag{2}$$

where the sum is taken over all computational paths π_v. Denote $k' = \lceil kc \rceil$. By the assumption of the theorem $\gamma(\vec{S}) > k'$.

Definition 17 *A path π_v in the decision tree* gives value *ϵ to variable x_i iff all variables $y_i^1, ..., y_i^\ell$ are queried on this path and been given values $\epsilon_1, ..., \epsilon_\ell$ so that $\epsilon = \epsilon_1 \oplus ... \oplus \epsilon_\ell$. A path is* flexible *if*

- *it does not give value 1 to any variable.*
- *its length is less than $k'\ell$.*
- *it outputs 0.*

Lemma 18 *Assume that Pr$[T(\vec{y}) \neq g_{\vec{S},k}(\vec{y})] < 1/m$. With probability at least $1/4$ a flexible path is chosen in T.*

Proof: Note that according to $D_{\vec{S}}$, $x = 0^n$ with probability $1/2$ and the value of $g_{\vec{S},k}(y)$ is always 1 when $x = 0^n$. Thus in the assumption $\Pr[T(y) \neq g_{\vec{S},k}(y)] < 1/m$ with probability $1/2 - 1/m$ the path chosen by T on input y outputs 0 and does not give 1 to any variable. Recall that we assumed the size of T be less than $n^{c(1-11/\ell)}$. By the union bound applied to Corollary 16 the probability that a path of length greater or equal than $k'\ell$ is chosen is less than

$$n^{c(1-11/\ell)} \cdot 2^{-k'\ell(1-10/\ell)} < 1/5.$$

Thus a flexible path is chosen with probability $1/2 - 1/m - 1/5$. $\square$

Lemma 19 *Let π_v be a flexible path. Then*

$$Pr[T(y) \neq g_{\vec{S},k}(y) | \pi_v \textit{ is chosen}] \geq 1/(m+1).$$

Proof: Denote by I_v the index set of the variables to which π_v gives a value, thus $|I_v| < k'$. Denote by $p_j = \Pr[\pi_v \text{ is chosen} | x = \chi_{S_j}]$ for $j = 1..m$ and $p_0 = \Pr[\pi_v \text{ is chosen} | x = 0^n]$. It is clear that for every $j = 0..m$ either $p_j = 0$ (in the case π_v gives value to some x_i inconsistent with χ_{S_j}) or $p_j = 2^{-(|\pi_v| - |I_v|)}$. Indeed after the choice of x is fixed one has to specify $|\pi_v| - |I_v|$ independent y-variables to determine whether π_v is chosen. Denote $q = 2^{-(|\pi_v| - |I_v|)}$. Denote by J the set of $j > 0$ for which $p_j = q$. Since $\gamma(\vec{S}) \geq k'$ and $|I_v| < k'$ there exists some S_j for which $S_j \cap I_v = \emptyset$, thus J is non-empty. Since π_v is flexible $p_0 = q$ as well. We write

$$\Pr[T(y) \neq g_{\vec{S},k}(y) | \pi_v \text{ is chosen}] = \Pr[g_{\vec{S},k}(y) \neq 0 | \pi_v \text{ is chosen}] = \Pr[x \neq 0^n | \pi_v \text{ is chosen}] =$$

$$= \frac{\Pr[x \neq 0^n \wedge \pi_v \text{ is chosen}]}{\Pr[\pi_v \text{ is chosen}]} = \frac{(|J|/2m) \cdot q}{(|J|/2m) \cdot q + 1/2 \cdot q} \geq 1/(m+1).$$

The lemma follows. $\square$

We are now ready to finish the proof of Theorem 14. Assume that $\Pr[T(y) \neq g_{\vec{S},k}(y)] < 1/m$ (otherwise the theorem follows). By Lemma 18 we infer that a flexible path is chosen with probability at least $1/4$. Consider the summation in (2).

$$\sum_{\pi_v} \Pr[T(y) \neq g_{\vec{S},k}(y) | \pi_v \text{ is chosen}] \cdot \Pr[\pi_v \text{ is chosen}] \geq$$

$$\Pr[T(y) \neq g_{\vec{S},k}(y) | \pi_v \text{ is chosen} \wedge \pi_v \text{ is flexible}] \Pr[\text{a flexible path is chosen}] \geq$$

$$\frac{1}{m+1} \cdot \frac{1}{4} > 1/(5m).$$

□

By the above two theorems the inappoximability of PMHS implies hardness for learning decision trees (Theorem 2). [AKP04] prove that if PMHS for all c can be solved in polynomial time for $k = \log n$, then $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{n^\lambda})$ for some $\lambda < 1$. Thus Theorem 3 follows.

5 Hardness of Learning DNF and Intersections of Halfspaces

In this section we prove our main hardness result for DNF formulae, namely that an algorithm for learning DNF in polynomial-time by ORs of threshold functions can be used to approximate the chromatic number of a graph. We will actually prove the equivalent hardness result for CNF formulae and ANDs of thresholds (intersections of halfspaces). It is easy to see that this will imply the intractability of properly learning both DNF formulae and intersections of halfspaces. We begin by defining a particular distribution over a set of examples corresponding to taking specialized products of a graph.

5.1 The Distribution

Given a graph $G = (V, E)$ we construct a distribution $\mathcal{D}$ over a set of examples as follows. We fix some positive integer parameter r, which might depend on n. The examples are from $\{0,1\}^{n\times r} = (\{0,1\}^n)^r$.

Definition 20 *Let $G(V, E)$ be a graph with n vertices and m edges. For a vertex v of G, let $z(v)$ denote the vector with a 1 in the ith position if v is the ith vertex of G and 0 everywhere else. For an edge $e = (u, v)$ of G let $z(e)$ be the vector with a 1 in positions i and j if u is the ith vertex of G and v is the jth vertex of G and 0 everywhere else.*

For each vector $(v_1, v_2, \ldots, v_r) \in V^r$ we associate a negative example $(z(v_1), \ldots, z(v_r), -)$. There are a total of $|V^r| = n^r$ negative examples. For each choice of k_1, k_2, such that $1 \leq k_1 \leq r$, $1 \leq k_2 \leq r$, $k_1 \neq k_2$, $e = (u, w) \in E$ and $v_i \in V$ for each $i = 1, 2, \ldots, r$, $i \neq k_1, k_2$ we associate a positive example

$$(z(v_1), \ldots, z(v_{k_1-1}), z(e), z(v_{k_1+1}), \ldots, z(v_{k_2-1}), \overline{0}, z(v_{k_2+1}), \ldots, z(v_r), +)\ .$$

Let S^+ denote the positive examples and S^- denote the negative examples. Set $S = S^+ \cup S^-$.

There are r ways to choose k_1, $r-1$ ways to choose k_2, $|E|$ ways to choose e, and $|V|^{r-2}$ ways to choose the rest of v_i's. Hence there is a total of $r \cdot (r-1) \cdot |E| \cdot n^{r-2}$ positive examples.

Distribution $\mathcal{D}$ is uniform over the above set of examples S, so the probability of each negative example is $\frac{1}{2\cdot n^r}$ and the probability of each positive example is $\frac{1}{2\cdot r\cdot(r-1)\cdot|E|\cdot n^{r-2}}$.

5.2 The Case of Small Chromatic Number

Here we prove that if the chromatic number $\chi(G)$ is small, then there exists a small CNF formula consistent with the examples above. Set $r = g(n)/\lambda = g/\lambda$, for some function g such that $r = g(n)/\lambda \leq \sqrt{n}/2$ and constant $\lambda < 1$. Hence $\lambda = g/r$. Then we have

Lemma 21 *If $\chi(G) \leq n^\lambda = n^{g/r}$, then there is a CNF consistent with the examples with at most n^g clauses, and hence of size n^g.*

Proof: Suppose $V = \bigcup_{i=1}^{\chi} I_i$, where I_i are independent sets. Such sets must exist by the definition of χ. Define the CNF formula $f(x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{\chi} \bigvee_{j \notin I_i} x_j$. We then define a formula on $r \cdot n$ variables, which we claim is consistent with the learning problem: $F((x_1^1, \ldots, x_n^1), \ldots, (x_1^r, \ldots, x_n^r)) = \bigvee_{k=1}^{r} f(x_1^k, \ldots, x_n^k) = \bigvee_{k=1}^{r} \bigwedge_{i=1}^{\chi} \bigvee_{j \notin I_i} x_j^k$.

Note that F above is not written as a CNF formula. It is, however, a disjunction of r CNF formulas, each having at most $\chi(G)$ clauses. Hence expanding the formula yields a CNF formula with at most $\chi(G)^r \leq (n^\lambda)^r = n^g$ clauses. So F can be written as a CNF formula satisfying the conditions of the lemma, and it is not to hard to check that it is consistent with all of the examples. □

5.3 The Case of Large Chromatic Number

In this section we assume that $\chi(G) \geq n^{1-\lambda}$, and we prove that no small AND-of-thresholds formula gives a good approximation to the learning problem.

Theorem 22 *Let G be a graph such that $\chi(G) \geq n^{1-\lambda}$. Let $F = \wedge_{i=1}^{\ell} h_i$ where $\ell < \frac{1}{2\chi r}\left(\frac{\chi-1}{\log n}\right)^r$. Then F has error at least $\frac{1}{n^{2g+4}}$ with respect to D.*

We will need the following covering lemma which was first proved by Linial and Vazirani [LV89] and is a special case of a result due to Feige on randomized graph products (Corollary 2.9 of [Fei95]):

Lemma 23 *[LV89] One needs at least $\left(\frac{\chi-1}{\ln n}\right)^r$ products of the form $I_1 \times I_2 \times \ldots \times I_r$, where the I_i's are independent sets, to cover $V^r = V \times V \times \ldots \times V$.*

Let a product in the above form be called a product of independent sets. At a high level, we will argue that any $h_k \in F$ correctly classifies very few negative examples that lie outside a particular product of independent sets. Then using the above lemma, it will follow that we need many h_k's to cover (correctly classify) all negative examples. We now proceed to the details.

Fix a particular $h_k \in F$. Let $h_k = \sum_{i=1}^{r}\sum_{j=1}^{n} \alpha_j^i x_j^i \geq \beta$. For each $i \leq r$, the i-coefficients in h are the coefficients of the form α_j^i, $j \leq n$. For each $i \leq r$, let I_i be the set of all $j \leq n$ such that there is no edge $(k, j) \in E$ such that α_k^i is less than α_j^i. (That is, we order all i-coefficients in nondecreasing order, and take the coefficients in order that are independent). Note that I_i is an independent set of G. Let $S_1^k = V \times I_2 \times \ldots \times I_r$, $S_2^k = I_1 \times V \times I_3 \times \ldots \times I_r$, and so forth. Let $S^k = \cup_{i=1}^{r} S_i^k$. The following lemma shows that h_k either misclassifies many positive examples, or misclassifies almost all negative examples outside of S^k.

Lemma 24 *Let $\{h_k\}_{k=1}^{l}$ be a family of half-spaces, and S^k as above. Let N denote the number of negative examples outside of $\cup_{k=1}^{l} S^k$ that $\wedge h_k$ classifies correctly. Then the number of positive examples that $\wedge h_k$ misclassifies is at least $N/2n$.*

Proof: Fix h_k and $I_1, I_2, \ldots, I_r$ as above. Let $\alpha = z(j_1), \ldots, z(j_r)$ be a negative example such that α is not in S^k, and $h_k(\alpha) = 0$. Thus $h_k(\alpha) = \alpha^1_{j_1} + \alpha^2_{j_2} + \ldots + \alpha^r_{j_r} < \beta$. Since α is not in S^k, there exist two j_i's, say j_1 and j_2 such that $j_1 \notin I_1$ and $j_2 \notin I_2$. Since j_1 is not in I_1, there is some vertex k_1 in I_1 such that the edge (j_1, k_1) is present in E_1 and similarly there is a vertex k_2 in I_2 such that the edge (j_2, k_2) is in E_2. By the way we chose I_1 and I_2, it follows that $\alpha^1_{k_1} \leq \alpha^1_{j_1}$ and $\alpha^2_{k_2} \leq \alpha^2_{j_2}$. Either (a) $\alpha^1_{j_1} \leq \alpha^2_{j_2}$, or (b) $\alpha^2_{j_2} < \alpha^1_{j_1}$. If (a) holds, then $\alpha^1_{k_1} + \alpha^1_{j_1} + \alpha^3_{j_3} + \ldots + \alpha^r_{j_r} < \beta$. But this corresponds to the positive example $\alpha' = (z(j_1, k_1), \overline{0}, z(j_3), \ldots, z(j_r))$ and thus h_k (and $\wedge h_k$) misclassifies α'. Similarly if (b) holds, then h_k (and $\wedge h_k$) misclassifies the positive example $\alpha' = (\overline{0}, z(j_2, k_2), z(j_3), \ldots, z(j_r)$. Thus we have a mapping from the set of all correctly classified negative examples outside of $\cup_{k=1}^{l} S^k$ to incorrectly classified positive examples. Since each positive example is mapped onto by at most $2n$ negative examples, it follows that the number of positive examples misclassified by $\wedge h_k$ is at least $N/2n$. □

Recall that F is the conjunction of ℓ threshold formulas, $h_1, \ldots, h_\ell$. For each h_k, let S^k be the associated set of cross products. Let the negative examples that h_k correctly classifies be denoted by $In_k \cup Out_k$, where In_k are those correctly classified negative examples in S^k, and Out_k are the remaining correctly classified negative examples.

Lemma 25 *Let S^k, $k \leq \ell$ be defined as above. If $\ell \leq \frac{1}{2\chi r} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$ then $n^r - |\cup_{k=1}^{l} S^k| \geq \frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$.*

Proof: If this were not the case, we would have a collection of $\ell \cdot \chi \cdot r \leq \frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$ products of independent sets which cover all but $m < \frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$ points of V^r. (To see this, replace the cross product $I_1 \times \ldots I_{i-1} \times V \times \ldots \times I_{i+1} \times \ldots \times I_r$ by χ cross products $I_1 \times \ldots I_{i-1} \times J_k \times I_{i+1} \times \ldots \times \ldots \times I_r$, where $k \leq \chi$, and $J_1, J_2, \ldots J_\chi$ is a partition of the vertices in G into χ independent sets.) Then by adding m singletons (which are trivially products of independent sets) we obtain a cover of V^r by $l\chi r + m < \left(\frac{\chi-1}{\ln n}\right)^r$ products of independent sets, which contradicts the above covering lemma (Lemma 23). □

We can now analyze the overall error with respect to D. Let $F = \wedge_{k=1}^{\ell} h_k$, where each h_k is a threshold formula, and $l < \frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r$

Let $R = \frac{1}{4} \cdot \left(\frac{\chi-1}{\ln n}\right)^r$. There are two cases to consider. The first case is when $|\cup_{k=1}^{\ell} Out_k| \geq R$. Then by Lemma 24, the number of positive examples that F misclassifies is at least $\frac{R}{2n}$. Thus the probability of error with respect to D is at least $\frac{R}{4n \cdot r \cdot (r-1) \cdot |E| \cdot n^{r-2}}$ which, for sufficiently large n, is at least:

$$\begin{aligned} R/n^{r+4} &= \frac{\frac{1}{4} \cdot \left(\frac{\chi-1}{\ln n}\right)^r}{n^{r+4}} \geq \frac{\frac{1}{4} \cdot \left(\frac{n^{1-g/r}-1}{\ln n}\right)^r}{n^{r+4}} \\ &> \frac{\left(n^{1-2g/r}\right)^r}{n^{r+4}} = n^{-2g-4} = \frac{1}{n^{2g+4}}. \end{aligned}$$

In the second case, $|\cup_{k=1}^{\ell} Out_k| < R$. But then by Lemma 25, the number of negative examples misclassified is at least $\frac{1}{2} \cdot \left(\frac{\chi-1}{\ln n}\right)^r - R$ which is equal to R. Thus the probability of an error with respect to D is at least $\frac{R}{2n^r}$, which again is at least $\frac{1}{n^{2g+4}}$ for sufficiently large n.

Finally, we have reduced the problem of approximating $\chi(G)$ to learning CNF:

Theorem 26 *Suppose that CNF is efficiently learnable by ANDs-of-thresholds in time $O(n^{kg(n)/2} \cdot s^k \cdot (\frac{1}{\epsilon})^k)$, where $k > 1$, and $1 \leq g(n) \leq \sqrt{n}/28k$. Then there exists a randomized algorithm for approximating the chromatic number of a graph within a factor of $n^{1-1/14k}$ in time $O(n^{14kg(n)+2})$. Moreover, the algorithm will always give a valid answer for $\chi \geq n^{1-1/14k}$.*

Proof: Set $\epsilon = \frac{1}{n^{2g+4}}$ and $r = 14kg$. Let G be a graph and let $\mathcal{D}$ be the distribution induced from G as described previously. Run the learning algorithm with respect to distribution $\mathcal{D}$. If it does not terminate after n^{9kg} steps output "$\chi \geq n^{1-1/14k}$". Otherwise, let h be the hypothesis the algorithm outputs. Calculate the error ϵ_h of h with respect to the distribution $\mathcal{D}$. If $\epsilon_h < \frac{1}{n^{2g+4}}$ output "$\chi \leq n^{1/14k}$", otherwise output "$\chi \geq n^{1-1/14k}$". We claim that this algorithm works with probability at least $3/4$ for sufficiently large n in approximating $\chi \leq n^{1/14k}$ and works perfectly for $\chi \geq n^{1-1/14k}$.

If $\chi \leq n^{1/14k}$, by Lemma 21, $s \leq n^g$. The number of variables in the underlying learning problem is $r \cdot n < n^2$. Hence the running time with probability $\geq 3/4$ is at most $O(n^{2 \cdot kg/2} n^{g \cdot k} n^{(2g+4)k}) \leq O(n^{8kg}) < n^{9kg}$ for sufficiently large n, and the output is supposed to have an error $< \epsilon = \frac{1}{n^{2g+4}}$. Hence the algorithm outputs "$\chi \leq n^{1/14k}$" with probability at least $3/4$ in this case.

If $\chi \geq n^{1-1/14k}$, by Lemma 22 the output of the algorithm must contain at least $\frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r$ terms in order to have an error $< \epsilon = \frac{1}{n^{2g+4}}$. In this case the running time of the algorithm (for n sufficiently large) is at least:

$$\begin{aligned}
\frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r &\geq \frac{1}{n^3}\left(\frac{n^{1-1/14k}-1}{\ln n}\right)^r \\
&\geq \frac{1}{n^3}\left(n^{1-1/13k}\right)^{14kg} \\
&> \frac{1}{n^3} n^{12kg} \geq n^{9kg}.
\end{aligned}$$

Hence if the algorithm terminates in n^{9kg} steps, its error will be bigger than ϵ, and the algorithm outputs "$\chi \geq n^{1-1/14k}$" with probability 1 in this case. □

Remark 27 *By negating the CNFs and the ANDs-of-thresholds in Theorem 26, we obtain the following:*

Suppose that DNF is efficiently learnable by ORs-of-thresholds in time $O(n^{kg(n)/2} \cdot s^k \cdot (\frac{1}{\epsilon})^k)$, where $k > 1$, and $1 \leq g(n) \leq \sqrt{n}/28k$. Then there exists a randomized algorithm for approximating the chromatic number of a graph within a factor of $n^{1-1/14k}$ in time $O(n^{14kg(n)+2})$. Moreover, the algorithm will always give a valid answer for $\chi \geq n^{1-1/14k}$.

We will require the following hardness result due to Feige and Kilian [FK96]:

Theorem 28 *[FK96] For any constant $\lambda > 0$, there exists a polynomial-time randomized reduction mapping instances f of SAT of length n to graphs G with $N = \mathsf{poly}(n)$ vertices with the property that if f is satisfiable then $\chi(G) \leq O(N^\lambda)$ and if f is unsatisfiable then $\chi(G) \geq \Omega(N^{1-\lambda})$. The reduction has zero-sided error.*

An immediate corollary is that approximating the chromatic number is hard:

Corollary 29 *[FK96] Let $\lambda > 0$ be a constant. Assume there exists an algorithm which approximates the chromatic number of a graph with n vertices within a factor of $n^{1-\lambda}$ in $\mathsf{RPTIME}(t(n))$ (with zero error if $\chi \geq n^{1-\lambda}$). Then $\mathsf{NP} \subseteq \mathsf{RPTIME}(t(n^a))$ for some constant $a \geq 1$.*

Now we can combine Theorem 26 and Corollary 29 to prove Theorem 4.

Proof:(of Theorem 4) If DNF formulae are learnable by ORs-of-thresholds in polynomial-time, we show how to approximate the chromatic number of a graph in polynomial-time to within a factor of n^{λ} for some small constant $\lambda > 0$. Let G be a graph on n vertices. From Remark 27 setting $g = 1$, we can approximate $\chi(G)$ within a factor of $n^{1-1/14k}$ in time $O(n^{14k+2})$ where k is a constant, with zero error for $\chi \geq n^{1-1/14k}$. Hence, by Corollary 29, $\mathsf{NP} \subseteq \mathsf{RPTIME}(n^{O(1)}) = \mathsf{RP}$. □

From the proof of Theorem 26 we can see that it is hard to learn even n^{λ}-term DNF by n^{b}-term OR-of-thresholds in time n^{b} for any constant $b \geq 0$. We can, under a stronger hardness assumption, prove stronger hardness results for learning superpolynomial size DNF formulae (i.e. if we do not restrict our concept class to be polynomial-size DNF formulae):

Corollary 30 *Suppose that* $\mathsf{SAT} \notin \mathsf{RPTIME}(O(n^{n^{\beta}}))$ *for some* β*. Then for any* $k > 0$ *there is* $\alpha > 0$ *such that DNF formulas are not properly learnable in time* $O(n^{n^{\alpha}} \cdot s^{k} \cdot (\frac{1}{\epsilon})^{k})$.

Notice that if we assume $\mathsf{SAT} \notin \mathsf{RPTIME}(2^{n^{\beta}})$ for some β and substitute $k = 1$ in Corollary 30 then we can conclude that DNF formulae are not properly learnable in time $O(n^{n^{\alpha}} \cdot s \cdot \frac{1}{\epsilon})$ for some $\alpha < 1$. Theorem 1 states, however, that DNF formula are properly learnable in time $2^{O((n \log s)^{1/2} \log n)}/\epsilon$, so our lower bound is fairly tight.

6 Hardness Results for Smaller Concept Classes

In the previous two sections our hardness results applied to learning the general class of DNF formulae and intersections of halfspaces. In this section we present new hardness results when we restrict the concept class to intersections of just 2 halfspaces or 2-term DNF formulae.

More specifically, we can show that it is hard to learn the intersection of 2-halfspaces by the intersection of any constant number of halfspaces and that it is hard to learn 2-term DNF formulae by any k-term DNF formula for any constant k. The results for intersections of halfspaces may be especially interesting in light of the fact that it is not known how to learn (even non-properly) the intersection of two n-dimensional halfspaces in time less than $2^{O(n)}$ (there is a simple, non-proper algorithm for learning k-term DNF in time $O(n^{k})$).

The main idea is to apply recent hardness results on the hardness of *hypergraph* coloring. Recently, several researchers [GHS00, Kho02, DRS02] have shown that it is hard to color *uniform* hypergraphs, i.e. hypergraphs where each hyperedge is of equal size.

6.1 Hardness for Learning Intersections of a Constant Number of Halfspaces

In this section we reduce the problem of coloring a 3-uniform hypergraph to a consistency problem for intersections of halfspaces:

Theorem 31 *The problem of* ℓ*-coloring a* k*-colorable hypergraph on* n *vertices reduces to finding an intersection of* l *halfspaces over* n *variables consistent with examples by labelled by an intersection of* ℓ *halfspaces.*

Proof: Let $H = (V, E)$ be a k-colorable hypergraph with n vertices. We construct a set of examples S classified by the intersection of k-halfspaces such that any intersection of ℓ-halfspaces consistent with S can be used to ℓ-color H (the reduction is an extension of the reduction by Blum and Rivest [BR92]).

Denote the vertices of H by $v_1, v_2, \ldots, v_n$. For a vertex $v_i \in V$, let $a(v_i)$ denote the vector of length n with a 1 in the ith position and 0 everywhere else. For an edge $e \subseteq V$ of G let $a(e)$ be the vector equal to $\sum_{v \in e} a(v)$ (that is, the characteristic vector for set e). Let 0^n denote the all zeroes vector of length n. Create the following set S of examples

- The example $(0^n, +)$.
- For every vertex $v \in V$, the example $(a(v), -)$.
- For every edge $e \in E$, the example $(a(e), +)$.

Assume H is colorable by ℓ colors according to the function χ. We construct an intersection of ℓ halfspaces consistent with S. Let $h_i = \mathsf{sign}(w_i \cdot x - \theta_i)$ where $\theta_i = -1/2$ and $w_i = (w_{i,1}, \ldots, w_{i,n})$ such that $w_{i,j}$ equals -1 if $\chi(v_j) = i$ and n otherwise. Set $h = \wedge_{i=1}^{\ell} h_i$.

Checking that $h_1 \wedge h_2, \ldots, \wedge h_\ell$ is consistent with S is straightforward; the example $(0^n, +)$ is satisfied and each $(a(v), -)$ example is consistent with the intersection. Finally given an edge e each $(a(e), +)$ is satisfied by each h_i since there exist two vertices in e which are colored in different colors and hence at least one of two vertices will contribute n to the weighted sum making the total positive.

For the other direction, assume there exists an intersection of ℓ halfspaces $h = h_1 \wedge, \ldots, \wedge h_\ell$ consistent with the examples in S. Construct a coloring for H as follows. Let $\chi(v) = t$ where t is the first halfspace h_t such that $h_t(a(v))$ is false. This assigns a color to each vertex. Now let $e \in E$. If there exists a color c such that $\forall v \in e,\ \chi(v) = c$ then $\forall v \in e,\ h_c(v) = 0$. Since $h(0^n)$ is positive, it implies that $h_c(0^n)$ is positive, and hence its threshold θ_c is negative. But for all i such that $v_i \in e$, $h_c(a(v_i)) = 0$ and thus $w_{c,i} < \theta_c$. This implies $\sum_{v_i \in e} w_{c,i} < \theta_c$ and therefore $h_c(a(e)) = 0$ contradicting the consistency with S. $\square$

We can now apply the following hardness result due to Dinur et al. [DRS02]:

Theorem 32 *[DRS02] It is* NP*-hard to* k*-color a* 2*-colorable* 3*-uniform hypergraph for any constant* $k \geq 0$.

Applying Theorem 31 and Theorem 32 we obtain Theorem 5, our main hardness result for this section.

Proof:(of Theorem 5) Assume there exists a polynomial time algorithm for learning the intersection of two halfspaces which outputs a hypothesis equal to an intersection of k halfspaces. Given a hypergraph H construct a set of examples S as above. Consider a distribution $\mathcal{D}$ which is uniform over this set of examples. Setting the error parameter $\epsilon = 1/(|S| + 1)$, run the algorithm to obtain with probability $3/4$ a hypothesis h equal to the intersection of k halfspaces. The algorithm will run in time polynomial in n and $1/\epsilon = |V| + |E| + 2$, that is, will be polynomial in the size of H. Since $\epsilon < 1/|S|$, h must be consistent with S. Hence from Theorem 31 we can reconstruct a coloring for H. $\square$

Dinur et al. [DRS02] also give the following hardness result under a slightly stronger assumption:

Theorem 33 *[DRS02] There is no polynomial time algorithm for coloring a* 2*-colorable* 3*-uniform hypergraph using* $O((\log \log n)^{1/3})$ *colors unless* $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{\log^{O(1)} n})$.

We obtain a corresponding hardness result:

Corollary 34 *Assume* $\mathsf{NP} \neq \mathsf{RPTIME}(2^{\log^{O(1)} n})$. *Then there is no polynomial-time algorithm for learning an intersection of* 2*-halfspaces by an intersection of* $O((\log \log n)^{1/3})$ *halfspaces.*

We can also prove a hardness result for learning two-node neural networks by neural networks with a constant number of origin centered hidden nodes:

Theorem 35 *Coloring a k-colorable hypergraph with 2^ℓ colors reduces to the problem of finding any function of ℓ origin-centered halfspaces consistent with a data set labelled by the intersection of k origin-centered halfspaces.*

Proof: Let $H = (V, E)$ and use the same reduction as in the proof of Theorem 31 to obtain a set of examples S without the 0^n example. We first note that if we define $h = \wedge_{i=1}^{\ell} h_i$ where h_i's are defined as before but with thresholds $\theta_i = 0$ we will get an intersection of k origin-centered halfspaces consistent with S.

For the other direction let f be a Boolean function of ℓ origin-centered halfspaces $h_1, \dots, h_\ell$ consistent with S. Put 2^ℓ colors in correspondence with every subset of ℓ halfspaces. Color v the color corresponding to which subset of the ℓ halfspaces are false on input $a(v)$. Assume e is a monochromatic edge. Then each $v \in e$ is set to false by the same subset of halfspaces. That is for every $j \leq \ell$ and $v_{i_1}, v_{i_2} \in e$, $h_j(e(v_{i_1})) = h_j(e(v_{i_2}))$. h_j is zero centered and hence $\mathsf{sign}(w_{j,i_1}) = \mathsf{sign}(w_{j,i_2})$. This implies that for every $v \in e$,

$$h_j(a(e)) = \mathsf{sign}(\sum_{v_i \in e} w_{j,i}) = h_j(a(v))$$

and thus $f(a(e)) = f(a(v))$ which contradicts the consistency with S. $\square$

For any learning algorithm outputting some representation of a function of ℓ halfspaces in polynomial time we have the following:

Corollary 36 *It is NP-hard to learn two-node neural networks by outputting a neural network with a constant number of hidden, origin-centered nodes.*

6.2 Hardness of Learning 2-term DNF formulae

Here we prove an analogous theorem for learning 2-term DNF formulae by extending the reduction by Pitt and Valiant [PV88] to hypergraphs.

Theorem 37 *Coloring a k-colorable hypergraph $H = (V, E)$ using ℓ colors reduces to learning k-term DNF formulae by outputting an ℓ-term DNF formulae.*

Proof: Let $\mathcal{A}$ be an algorithm for learning k-term DNF formulae by ℓ-term DNF formulae and let $H = (V, E)$ be any k-colorable hypergraph on n vertices. For a vertex $v_i \in V$ let $a(v_i)$ be the a vector of length n which is equal to 0 in position i and 1 elsewhere. For an edge $e \in E$ let $a(e) = \bigwedge_{v \in e} a(v)$ (conjunction is applied bitwise).

We construct a set of examples S as follows:

- Vertex examples: for each $v \in V$, $(a(v), +)$.
- Edge examples: for each $e \in E$, $(a(e), -)$.

We now claim that any k-coloring of H can be efficiently translated into a k-term DNF formula consistent with the given examples and vice versa. Let χ be a k-coloring of H. For every color $c \leq k$ we define

$$t_c = \bigwedge_{\chi(v_i) \neq c} x_i \ ,$$

that is, t_c is the conjunction of all the variables whose corresponding vertices are not colored in color c. We set $h = t_1 \vee t_2 \vee \ldots \vee t_k$. Clearly h is a k-term DNF formula and the translation is efficient. For every vertex example $a(v_i)$, $t_{\chi(v_i)}(a(v_i)) = 1$ and hence $h(a(v_i)) = 1$. For any edge example $a(e)$, vertices in e are colored in at least two different colors and hence every term t_c will contain at least one variable x_i such that $v_i \in e$. This means that h will not satisfy $a(e)$.

Now let $h = t_1 \vee t_2 \vee \ldots \vee t_\ell$ be a DNF expression consistent with the given examples. For every vertex v, we define $\chi(v) = c$ if $a(v)$ is satisfied by t_c (if there are several terms that satisfy $a(v)$ we choose the one with the smallest c). Clearly this defines a mapping of vertices into ℓ colors. Take $e \in E$ and assume that all the vertices in it are colored in color c. That is, for each $v \in e$, $t_c(a(v)) = 1$. This implies that

$$t_c(a(e)) = t_c(\bigwedge_{v \in e} a(v)) = \bigwedge_{v \in e} t_c(a(v)) = 1$$

contradicting the consistency with example $(a(e), -)$. $\square$

Applying the reduction described in Theorem 37 with the hardness result from Theorem 32 we obtain the following hardness result for properly learning DNF formulae:

Theorem 38 *Assuming* $\mathsf{NP} \neq \mathsf{RP}$ *there is no polynomial-time algorithm for learning* 2*-term DNF formulae by* k*-term DNF formulae for any constant* k*.*

To contrast this with known results for learning k-term DNF, note that a k-term DNF is learnable in time $O(n^k)$ where the hypothesis is a CNF of size $O(n^k)$.

7 Acknowledgments

We thank Leslie Valiant for his valuable suggestions and remarks on this research. We are grateful to Rocco Servedio for allowing us to include Theorem 1 which was proved jointly with him. We also thank Subhash Khot, Ryan O'Donnell, and Avi Wigderson for useful conversations.

References

[AKP04] M. Alekhnovich, S. Khot, and T. Pitassi. Inapproximability of fixed parameter problems. 2004. Manuscript.

[AR01] M. Aleknovich and A. Razborov. Resolution is not automatizable unless w[p] is tractable. In *IEEE Symposium on Foundations of Computer Science*, 2001.

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[BKPS98] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability of random k-CNF formulas. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 561–571, Dallas, TX, May 1998.

[BPR97] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *Journal of Symbolic Logic*, 62(3):708–728, September 1997.

[BR92] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127, 1992.

[Bsh96] N. Bshouty. A subexponential exact learning algorithm for DNF using equivalence queries. *Information Processing Letters*, 59:37–39, 1996.

[BSW99] E. Ben-Sasson and A. Wigderson. Short proofs are narrow – resolution made simple. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 517–526, Atlanta, GA, May 1999.

[CEI96] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Gröbner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 174–183, Philadelphia, PA, May 1996.

[DF98] R. Downey and M. Fellows. Parameterized complexity. 1998.

[DRS02] Irit Dinur, Oded Regev, and Clifford D. Smyth. The hardness of 3-Uniform hypergraph coloring. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS-02)*, pages 33–42, Los Alamitos, November 16–19 2002. IEEE COMPUTER SOCIETY.

[EH89] A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

[Fei95] U. Feige. Randomized graph products, chromatic numbers, and the Lovász θ-function. In *stoc95*, pages 635–640, 1995.

[FK96] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity (CCC-96)*, pages 278–289, Los Alamitos, May 24–27 1996. IEEE Computer Society.

[GHS00] V. Guruswami, J. Hastad, and M. Sudan. Hardness of approximate hypergraph coloring. In IEEE, editor, *41st Annual Symposium on Foundations of Computer Science: proceedings: 12–14 November, 2000, Redondo Beach, California*, pages 149–158, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2000. IEEE Computer Society Press.

[Gol78] E. A. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[HJLT95] Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees (extended abstract). In *12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *lncs*, pages 527–538, Munich, Germany, 2–4 March 1995. Springer.

[HR02] Lisa Hellerstein and Vijay Raghavan. Exact learning of DNF formulas using DNF hypotheses. In ACM, editor, *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, Montréal, Québec, Canada, May 19–21, 2002*, pages 465–473, New York, NY, USA, 2002. ACM Press.

[JKS02] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond ac^0. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, 2002.

[Kha93] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual Symposium on Theory of Computing*, pages 372–381, 1993.

[Kho02] Subhash Khot. Hardness results for coloring 3-Colorable 3-Uniform hypergraphs. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS-02)*, pages 23–32, Los Alamitos, November 16–19 2002. IEEE COMPUTER SOCIETY.

[KLPV87] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of boolean formulae. In *Proceedings of the Nineteenth Annual Symposium on Theory of Computing*, pages 285–295, 1987.

[KS01] A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. In *Proceedings of the Thirty-Third Annual Symposium on Theory of Computing*, pages 258–265, 2001.

[KV94] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

[LV89] N. Linial and U. Vazirani. Graph products and chromatic numbers. In IEEE, editor, *30th annual Symposium on Foundations of Computer Science, October 30–November 1, 1989, Research Triangle Park, North Carolina*, pages 124–128, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1989. IEEE Computer Society Press.

[MOS03] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning juntas. In ACM, editor, *Proceedings of the Thirty-Fifth ACM Symposium on Theory of Computing, San Diego, CA, USA, June 9–11, 2003*, pages 206–212, New York, NY, USA, 2003. ACM Press.

[NJS98] R. Nock, P. Jappy, and J. Sallantin. Generalized graph colorability and compressibility of boolean formulae. In *Proceedings of the 9th International Symposium on Algorithms and Computation (ISAAC)*, 1998.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings of the Thirty-Eighth Annual Symposium on Foundations of Computer Science*, pages 458–467, 1997.

[PB90] L. Pitt and R. Board. On the necessity of Occam algorithms. In *Proc. 23th Annu. ACM Sympos. Theory Comput.*, pages 54–63. ACM Press, 1990.

[PV88] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[Val84] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.