Electronic Colloquium on Computational Complexity, Report No. 127 (2005)

Hardness of Approximate Two-level Logic Minimization and PAC Learning with Membership Queries

Vitaly Feldman[*]
Harvard University
vitaly@eecs.harvard.edu

November 4, 2005

Abstract

Producing a small DNF expression consistent with given data is a classical problem in computer science that occurs in a number of forms and has numerous applications. We consider two standard variants of this problem. The first one is *two-level logic minimization* or finding a minimal DNF formula consistent with a given complete truth table (TT-MinDNF). This problem was formulated by Quine in 1952 and has been since one of the key problems in logic design. It was proved **NP**-complete by Masek in 1979. The best known polynomial approximation algorithm is based on a reduction to the SET-COVER problem and produces a DNF formula of size $O(d \cdot \mathrm{OPT})$, where d is the number of variables. We prove that TT-MinDNF is **NP**-hard to approximate within d^γ for some constant $\gamma > 0$, establishing the first inapproximability result for the problem.

The other DNF minimization problem we consider is PAC learning of DNF expressions when the learning algorithm must output a DNF expression as its hypothesis (referred to as *proper* learning). We prove that DNF expressions are **NP**-hard to PAC learn properly even when the learner has access to membership queries, thereby answering a long-standing open question due to Valiant [39]. Finally, we observe that inapproximability of TT-MinDNF implies hardness results for restricted proper learning of DNF expressions with membership queries even when learning with respect to the uniform distribution only.

[*]Supported by grants from the National Science Foundation NSF-CCF-9877049, NSF-CCF-0432037, and NSF-CCF-0427129.

ISSN 1433-8092

1 Introduction

The problem of finding a minimal-size disjunctive normal form expression consistent with a given truth table (TT-MinDNF) is one of the oldest problems in computer science. It was formulated by the famous logician and philosopher Willard Van Quine in his work on mathematical logic [31, 32]. His algorithm for simplifying logical steps was also discovered in 1956 by Edward McCluskey in the context of circuit design [26]. Besides its important role in circuit design (in particular, VLSI design, Programmable Logic Array synthesis, and multi-level logic synthesis [9]) the problem has more recently appeared in reliability analysis [8], IP routing table compaction [23], and high-dimensional data representation [1]. This array of application has lead to an ongoing effort by many researchers to seek efficient heuristic and exact minimization procedures. We direct the interested reader to [9] for an overview of a large number of publications and some software tools. In the original Quine-McCluskey algorithm and in most of the later approaches, after a number of simplification steps the problem is reduced to an instance of the classical SET-COVER problem. Then, either an exact solution is found via the brute-force search, or an approximate solution is found using a certain heuristic. In the former case the size of the search space is not theoretically analyzed and in the latter no guarantees on the quality (i.e. size) of the output are given (both are usually measured empirically).

Far less work has been done on the theoretical side of this problem. Gimpel [15] and Paul [29] showed that Quine-McCluskey method can produce instances of SET-COVER that are **NP**-hard to solve. Then, in 1979, the full truth table version was proven **NP**-complete by Masek [25] (his manuscript was not published but the proof can be found in surveys by Czort [10] and Umans *et al.* [37]). Inapproximability results are only known for a generalization of TT-MinDNF that allows "don't care" values in the truth table (i.e. , the truth table is partial). Allender *et al.* prove that this problem (we denote it by PTT-MinDNF) is **NP**-hard to approximate within any constant factor and cannot be approximated within $\log d$ factor unless $\mathbf{NP} \subseteq \mathbf{RTIME}(n^{polylog(n)})$, where d is the number of variables [3]. They also produced a simpler proof (than Masek's) for **NP**-hardness of TT-MinDNF.

On the approximation side the only known efficient approximating algorithm is the one resulting from using the greedy algorithm to solve the SET-COVER instance obtained in Quine-McCluskey algorithm. It results in $\ln 2^d = O(d)$ approximation factor.

In this paper we present the first result on hardness of approximating TT-MinDNF. More specifically, we prove the following theorem.

Theorem 1 *There exists a constant $\gamma > 0$ such that it is* **NP***-hard to approximate TT-MinDNF to within a factor d^γ, where d is the number of variables of the TT-MinDNF instance.*

This result implies that the approximation factor achieved by the greedy algorithm is at most polynomially larger than the best possible.

Learning is another context where finding a small DNF formula consistent (or almost) with the given data is a fundamental problem. The problem was formulated by Leslie Valiant in his seminal paper introducing the PAC model of learning [39] and has been the subject of numerous subsequent works. A number of questions related to PAC learning of DNF expressions were posed by Valiant [39, 40]. Specifically, he asked whether DNF expressions are learnable from random examples with or without the use of the membership query (MQ) oracle. Valiant's original definition required that the learning algorithm output a DNF expression but this restriction was later relaxed to any efficiently-computable hypothesis with the stricter version being referred to as *proper* learning. All these variants of the DNF learning question remained open until a recent result by Alekhnovich *et al.* that established **NP**-hardness of the hardest variant: proper learning from random examples only [2]. Building on their proof, we resolve one more of Valiant's questions:

Theorem 2 (*informal*) *If* $\mathbf{NP} \neq \mathbf{RP}$ *then there is no polynomial-time PAC learning algorithm for DNF expressions that outputs a DNF expression even when the learning algorithm has access to the membership oracle.*

Besides, we observe that hardness of TT-MinDNF implies hardness of *strongly proper* learning of DNF expressions with MQs even with respect to the uniform distribution, where strongly proper means that the size (number of terms) of a hypothesis has to be upper-bounded by the DNF-size of the target function. Our inapproximability result then translates to hardness even when the size of a hypothesis is $O(\log^\gamma n)$ times larger than the size of the target. We note that, as proved by Jackson, unrestricted DNF expressions are learnable non-properly in this strong model [19], and hence our result highlights the importance of knowledge representation in this model.

Access to membership queries plays an instrumental role in numerous learning algorithms (many of which are proper), but hardness results for learning with MQs are still very scarce. These results are the first to show that PAC learning can be **NP**-hard even when MQs are available.

1.1 Relation to Other Work

Besides the results that we have already mentioned, one of the most significant results in DNF minimization is Umans' proof that finding a minimal DNF formula for a function given by a DNF formula (also called finding a minimal equivalent DNF and denoted MinEquDNF) is Σ_2^p-hard to approximate within N^γ for some constant $\gamma > 0$, where N is the size of the given DNF formula [36]. Despite the same goal in both problems the difference in input makes the nature of the problem (and, eventually, the proof techniques) very different. In particular, the gaps differ exponentially in terms of the size of hard instances.

Our hardness results for learning DNF expressions are contrasted by the fact that monotone DNF expressions are known to be strongly properly PAC learnable with MQs[39]. In addition to that, DNF expressions with k terms are known to be learnable by DNF expressions with 2^k terms when MQs are available [6]. It is also interesting to note that known non-trivial algorithms for learning unrestricted DNF formulae (running in time $2^{\tilde{O}(n^{\frac{1}{3}})}$ [22] and in time $2^{\tilde{O}(\sqrt{n})}$ with DNF hypotheses [2]) use only random examples and it is unknown whether they could be sped-up by using MQs.

Initial hardness results for properly learning DNF formulae due to Pitt and Valiant [30] show that unless $\mathsf{RP} = \mathsf{NP}$, k-term DNF formulae over n variables are not learnable by $2k$-term DNF. These results were strengthened by Nock *et al.* [28] who proved similar hardness even when learning by formulas of size $k^\alpha n^\beta$ (where $\alpha \leq 2$ and β is any constant). Finally, Alekhnovich *et al.* removed any bounds on the size of the hypothesis (other than those naturally imposed by the polynomial running time of the learning algorithm) [2]. Angluin and Kharitonov prove that if non-uniform one-way functions exist then MQs do not help predicting DNF formulae [4]. However, their reduction does not preserve the representation of a hypothesis and therefore cannot be combined with the result by Alekhnovich *et al.* to obtain hardness of proper learning with MQs.

Hardness results for learning of DNF expressions with MQs are only known for the *exact* model of learning (which is weaker than PAC learning) and only for strong proper learning (or slight relaxations similar to the one we prove for PAC learning with respect to the uniform distribution). The strongest results in this model are due to Hellerstein and Raghavan [18] and are based on information-theoretic hardness.

For proper PAC learning without MQs a number of hardness results are known for several other representations [7, 16, 21, 2]. Hardness results for some other variants of DNF minimization can be found in a survey by Umans *et al.* [37].

1.2 Outline and Organization

The proof of the TT-MinDNF hardness result is based on first reducing TT-MinDNF to a more general problem of covering a subset of the Boolean hypercube with a given set of subcubes (we denote it by PHC-COVER), a problem that can be seen as a geometric version of the general SET-COVER problem. This reduction substantially simplifies the second step of the proof, which is a reduction from a multi-prover proof system with certain simple properties to PHC-COVER. This reduction follows the key ideas of the inapproximability result for SET-COVER by Lund and Yannakakis [24]. Finally, a low-error PCP by Raz and Safra [33] is used to obtain a multi-prover proof system with the desired properties, yielding the inapproximability result for TT-MinDNF.

Besides the main reduction in Appendix B we show a simple reduction from hypergraph vertex cover problem to PHC-COVER. The reduction is based on families of sets in which none of the sets is covered by k others. This reduction together with a recent result by Dinur *et al.* [11] implies analogous hardness of approximation for TT-MinDNF under a stronger assumption $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{\log(n)})$.

The hardness of learning DNF expressions result is based on the proof by Alekhnovich *et al.* [2] that is, in turn, based on hardness of approximating the chromatic number of a graph by Feige and Kilian [14]. In essence, we augment the reduction from coloring to finding a small consistent DNF by providing a way to efficiently define the value of the target function on the whole hypercube without revealing any additional information about coloring and without changing the DNF-size of the target function substantially. This allows for simulation of the membership oracle in the DNF hardness reduction.

The rest of the paper is organized as follows. In Section 3 we show that TT-MinDNF and two other covering problems on athe hypercube can be reduced (in an approximation-preserving way) to PHC-COVER. Then, in Section 4, PHC-COVER is reduced to the low-error PCP by Raz and Safra [33] giving the desired hardness of approximation result. In Section 5 we prove the above-mentioned hardness results for proper learning with MQs.

2 Preliminaries

A Boolean partial function is a function $f : \{0,1\}^d \rightarrow \{0,1,*\}$. We say that a Boolean function g is consistent with a partial function f, if for every $a \in \{0,1\}^d$ such that $f(a) \neq *$,

$g(a) = f(a)$. A subcube of a Boolean hypercube is a set $I_1 \times I_2 \times \cdots \times I_d$ where for each j, $I_j \subseteq \{0,1\}$. We identify each subcube with a term whose satisfying assignments are exactly the elements of the subcube.

The *size* of a DNF formula is the number of terms in it. The DNF-size of a function is the size of a minimal DNF formula equal to the function. Given the truth table of a function f the problem of finding the DNF-size of f is denoted TT-MinDNF. When f is a partial function the problem of finding the size of a minimal DNF consistent with f is referred to as PTT-MinDNF.

The problem of finding the size of a minimal cover of the d-dimensional Boolean hypercube with subcubes represented by the terms in $\mathcal{T} = \{T_i\}_{i=1}^m$ is referred to as HC-COVER. We also consider the following generalization of HC-COVER. Given a set of terms as above and a set of points $S \subseteq \{0,1\}^d$ find the size of a minimal cover of S by terms in $\mathcal{T}$. We refer to this generalized version as PHC-COVER. We say that PTT-MinDNF$(f) = C$ (HC-COVER$(\mathcal{T}) = C$, or PHC-COVER$(S, \mathcal{T}) = C$) if the size of a minimal DNF formula consistent with f (or respective cover for an instance $\mathcal{T}$ or $(S, \mathcal{T})$) equals C.

In all the above problems, we assume that the input is of size poly(2^d) (it cannot be larger as there are 3^d different terms). For PHC-COVER and HC-COVER the input can, in certain situations, be represented more concisely. However, for consistency with the definition of the usual set cover problem, we assume that all the 2^d points of the cube are given explicitly as part of the input.

We use a dot '$\cdot$' to denote concatenation of bits and bit vectors. Let $\texttt{par}()$ denote the parity function defined for any bit vector. For any Boolean variable v and $b \in \{0,1\}$, let $\ell_v(b) = v$, if $b = 1$ and $\ell_v(b) = \bar{v}$, if $b = 0$. Similarly, for a vector of variables $w \in V^l$ and a vector $a \in \{0,1\}^l$, we define $\texttt{eq}(w, a) = \wedge_{i \le r} \ell_{w_i}(a_i)$, or simply the term that checks if variables of w are set to a.

Besides the usual disjunctions and conjunctions we consider *threshold* or *halfspace* gates equal to $\texttt{sign}(\sum_i w_i x_i - \theta)$ for some real-valued $w_1, \ldots, w_n, \theta$. AND-of-thresholds (OR-of-thresholds) formula is a two-level formula with an AND (respectively OR) gate at the top (output) level and thresholds at the bottom level. The size of such a formula is the number of thresholds gates in it.

2.1 Learning Model

Our learning model is Valiant's well-known Probably Approximately Correct (PAC) learning model [39]. In this model, for a concept c and a distribution D over domain X an *example oracle* $\mathrm{EX}_D(c)$ 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. The membership oracle MEM(c) is the oracle that given any point $x \in X$ returns the value $c(x)$. For $\epsilon \ge 0$ we say that function g ϵ-approximates function f with respect to distribution D if $\mathbf{Pr}_D[f(x) = g(x)] \ge 1 - \epsilon$. 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 over X, $\mathcal{A}(n, \epsilon, \delta)$, runs in time polynomial in n, $1/\delta$, $1/\epsilon$, $|c|$ and, with probability at least $1 - \delta$, outputs an efficiently computable hypothesis h that ϵ-approximates c. In the basic PAC model $\mathcal{A}$ is allowed to use only random example oracle $\mathrm{EX}_D(c)$. We denote the model in which the learner also has access to MEM(c) by PAC+MQ.

When the hypothesis is in the same representation as C the algorithm $\mathcal{A}$ is called *proper*.

If, in addition, h has size at most $|c|$, then the learning algorithm is called *strongly* proper. The distribution specific version of this model requires the learning algorithm to succeed only with respect to some specific distribution (in our case it will be the uniform distribution).

3 Hypercube Reductions

Below we show that the covering problems defined in the previous section have similar approximation complexity by describing efficient reductions from PHC-COVER to PTT-MinDNF, from PTT-MinDNF to TT-MinDNF, and from TT-MinDNF to HC-COVER (in Appendix A.1). Our reductions preserve the approximation ratio and increase the number of variables by a small constant factor. By the recent result of Allender *et al.* for PTT-MinDNF [3], we immediately get that all the discussed problems are **NP**-hard to approximate to within any constant (or within factor $\log d$ assuming $\mathbf{NP} \subseteq \mathbf{RTIME}(n^{polylog(n)})$).

It can be easily seen that PTT-MinDNF is an instance of PHC-COVER. For the other direction our reduction converts an instance of PHC-COVER given by a set $S \subseteq \{0,1\}^d$ and a set of terms $\mathcal{T}$, to an instance of PTT-MinDNF given by a function f where each element of $\mathcal{T}$ corresponds to a prime implicant of f.

Theorem 3 *There exists an algorithm that given an instance $(S, \mathcal{T})$ of PHC-COVER over d variables produces the truth table of partial function g over $2d$ variables such that $(S, \mathcal{T})$ has a cover of size C if and only if there exists a C-term DNF formula consistent with g. The algorithm runs in time $2^{O(d)}$.*

Proof: For a point $x \in \{0,1\}^d$, let $p[x]$ denote a point in $\{0,1\}^{2d}$ equal to $x \cdot \bar{x}$ (that is, x on first d coordinates and the bit complement of x on coordinates from $d+1$ to $2d$). For a term T over d variables, let $p[T]$ denote a term over $2d$ variables in which all the positive literals are the same as in T while each negative literal $\bar{x}_i$ is replaced by literal x_{d+i}. Let $g(y) = \bigvee_{T \in \mathcal{T}} p[T](y)$. Then we define

$$f(y) = \begin{cases} 0 & \text{if } g(y) = 0 \\ 1 & \text{if } y = p[x] \text{ and } x \in S \\ * & \text{otherwise} \end{cases}$$

Let $\mathcal{S} \subseteq \mathcal{T}$ be a set of C terms such that $S \subseteq \cup_{T \in \mathcal{S}} T$. We claim that $h(y) = \bigvee_{T \in \mathcal{S}} p[T](y)$ is consistent with f. Let y be a point in $\{0,1\}^{2d}$. If $f(y) = 0$, then $g(y) = 0$ and so $h(y) = 0$. If $f(y) = 1$ then there exists x such that $y = p[x]$ and $x \in S$. Therefore, there exists $T \in \mathcal{S}$ such that $T(x) = 1$, which is equivalent to $p[T](p[x]) = 1$. In particular, $h(y) = 1$, which completes the proof of the claim.

For the other direction, let $h = \bigvee_{T \in \mathcal{S}} T$ be a C-term DNF formula consistent with f. For $T \in \mathcal{S}$, let y be the point with the minimal number of 1's accepted by T. By the consistency with f, we get that $f(y) \neq 0$ and hence $g(y) \neq 0$. Therefore let $m[T]$ be the term of g that covers y. We note that $m[T]$ covers all the points covered by T and denote the term $T' \in \mathcal{T}$ such that $p[T'] = m[T]$ by $p^{-1}[m[T]]$. Define $\mathcal{S}' = \{p^{-1}[m[T]] \mid T \in \mathcal{S}\}$. If $x \in S$, then $f(p[x]) = 1$ and therefore, there exists $T \in \mathcal{S}$ such that $T(p[x]) = 1$. This, in turn, implies that $p^{-1}[m[T]](x) = 1$, that is, $\mathcal{S}'$ is a set of C subsets from $\mathcal{T}$ that covers S. $\square$

The next step is reducing from a partially-specified truth table to a fully-specified one. A part of this reduction is based on Gimpel's reduction from partially to fully specified truth-table [15].

Theorem 4 *There exists an algorithm that given the truth table of a partial function f on d variables and an integer $r \geq 1$ produces the truth table of partial function g over $d+r+2$ variables such that there exists a C-term DNF consistent with f if and only if there exists $(2^{r-1}C + |f^{-1}(*)|)$-term DNF formula equal to g. The algorithm runs in time $2^{O(r+d)}$.*

Proof: The reduction has two components. First component introduces a term for each point x where f equals $*$ in a way that forces any consistent DNF to include all the introduced terms. The introduction of new terms skews the original approximation ratio and therefore the second component replicates f 2^{r-1} times to ensure that the size of the cover is still dominated by the original problem (for large enough r).

For a vector in $\{0,1\}^{d+r+2}$, we refer to its first d variables as $x_1, \ldots, x_d$; its next r variables as $y_1, \ldots, y_r$; and its last two variables as z_1, z_2. We define Boolean function g over $\{0,1\}^{d+r+2}$ as follows:

$$g(xyz) = \begin{cases} \texttt{par}(y) & \text{if } f(x) = 1 \text{ and } z = 11 \\ 1 & \text{if } f(x) = * \text{ and } (z = \texttt{par}(x) \cdot \neg\texttt{par}(x) \text{ or } z = 11) \\ 0 & \text{otherwise} \end{cases}$$

Let $S = f^{-1}(*)$. We claim that there exists a C-term DNF consistent with f if and only if there exists a $(2^{r-1}C + |S|)$-term DNF equal to g. For the simpler direction, let $\mathcal{S} \subseteq \mathcal{T}$ be a set of C terms such that $h(x) = \bigvee_{T \in \mathcal{S}} T(x)$ is consistent with f. Let $Z(1) \equiv z_1$ and $Z(0) \equiv z_2$. By the consistency of h and f, and the definition of g, it is easy to verify that

$$g(xyz) = (z_1 \wedge z_2 \wedge h(x) \wedge \texttt{par}(x)) \bigvee \left(\bigvee_{a \in S} \texttt{eq}(x,a) \wedge Z(\texttt{par}(a)) \right).$$

This expression is not in DNF. To convert it we note that $\texttt{par}(x) = \bigvee_{a \in \{0,1\}^r, \texttt{par}(a)=1} \texttt{eq}(x,a)$ and therefore

$$g(xyz) = \left(\bigvee_{T \in \mathcal{S}, a \in \{0,1\}^r, \texttt{par}(a)=1} z_1 \wedge z_2 \wedge T \wedge \texttt{eq}(x,a) \right) \bigvee \left(\bigvee_{a \in S} \texttt{eq}(x,a) \wedge Z(\texttt{par}(a)) \right),$$

that is, g has a DNF expression with $C2^{r-1} + |S|$ terms.

For the other direction, let $\mathcal{T}$ be a set of $C2^{r-1} + |S|$ terms such that $g(xyz) = \bigvee_{T \in \mathcal{T}} T(xyz)$. For each $a \in S$, let $\tau_a \in \mathcal{T}$ be a term that accepts point $p(a) = a \cdot 0^r \cdot \texttt{par}(a) \cdot \neg\texttt{par}(a)$. We first prove that τ_a contains all the literals of $\texttt{eq}(x,a)$. If τ_a does not contain literal $\ell_{x_i}(a_i)$, then let a^i be the point a with the i-th bit negated. Clearly τ_a will also accept the point $a^i \cdot 0^r \cdot \texttt{par}(a) \cdot \neg\texttt{par}(a)$. But this contradicts the consistency with g, since $\texttt{par}(a) = \neg\texttt{par}(a^i)$. It follows that for each $a \in S$, there is a *distinct* term in $\mathcal{S}$ that can only accept points with x part in S. We denote this set of terms by $\mathcal{T}^*$.

Now let $D = \{p \mid p \in \{0,1\}^r, \texttt{par}(p) = 1\}$, p be any point in D and a be any point such that $f(a) = 1$. Then, by definition of g, there exists a term $\tau_{p,a}$ that accepts the point $a \cdot p \cdot 11$. We claim that $\tau_{p,a}$ contains all the literals of $\texttt{eq}(y,p)$. If $\tau_{p,a}$ does not contain literal

$\ell_{y_i}(p_i)$, then let p^i be the point d with the i-th bit negated. Clearly $\tau_{p,a}$ will also accept the point $a \cdot p^i \cdot 11$. But this contradicts the consistency with g, since $g(a \cdot p^i \cdot 11) = \mathtt{par}(p^i) = 0$. Now let $\mathcal{T}_p = \{\tau_{p,a} \mid f(a) = 1\}$ and let $h_p(x) = \bigvee_{T \in \mathcal{T}_p} T(x \cdot p \cdot 11)$. We claim that $h_p(x)$ is consistent with f. This is true since if $f(a) = 1$ then $\tau_{p,a} \in \mathcal{T}_p$ and $\tau_{p,a}(a \cdot p \cdot 11) = 1$. If $f(a) = 0$ then $g(a \cdot p \cdot 11) = 0$ and since $\mathcal{T}_p \subseteq \mathcal{T}$ then no term in $\mathcal{T}_p$ can accept point $a \cdot p \cdot 11$. As we have shown, all the $\mathcal{T}_p$'s for $p \in D$ are disjoint and they are clearly disjoint from $\mathcal{T}^*$ (since $0^r \notin D$). Therefore $|\mathcal{T}| \geq |S| + \sum_{p \in D} |\mathcal{T}_p|$. As $|D| = 2^{r-1}$ we get that there exists p such that $|\mathcal{T}_p| \leq C$ and hence h_p is a C-term DNF formula consistent with f. $\square$

By suitable choice of r in Theorem 3 one obtains the following corollary:

Corollary 5 *If TT-MinDNF can be approximated within $h(d)$ in time $t(d)$ then PTT-MinDNF can be approximated within $h(2d + \log d) + 1$ in time $t(2d + \log d) + 2^{O(d)}$.*

Proof: We note that the claim is trivial for $h(d) \geq d$ since both problems are instances of SET-COVER with the greedy algorithm giving $\ln 2^d < d$ approximation. Therefore, for simplicity, we restrict our attention to cases when $h(d) \leq d/10$. To obtain the claim we set $r = d + 2\log d - 2$. Then if $\mathtt{DNF\text{-}size}(f) = k$ and $|f^{-1}(*)| = s \leq 2^d$, then $\mathtt{DNF\text{-}size}(g) = 2^{r-1}k + s$ and the approximating algorithm for TT-MinDNF will return a value $l \leq h(d + r + 2)(2^{r-1}k + s)$. This would imply that

$$\mathtt{DNF\text{-}size}(f) \leq \frac{l - s}{2^{r-1}} = \frac{h(d+r+2)(2^{r-1}k + s) - s}{2^{r-1}} = h(d+r+2)k + \frac{(h(d+r+2) - 1)s}{2^{r-1}}$$

But $h(d + r + 2) \leq (d + r + 2)/10$ and therefore for $r = d + \log d - 2$,

$$\frac{h(d+r+2)(2^{r-1}c + s) - s}{2^{r-1}} < \frac{s(d + \log d)/10}{2^d d/8} < 1$$

for large enough d. Hence $\frac{l-s}{2^{r-1}} \leq h(2d + \log d)k + 1 \leq (h(2d + \log d) + 1)k$. $\square$

We summarize the reductions in this section by the following equivalence theorem:

Theorem 6 *If there exists a constant $0 < \gamma \leq 1$ such that there is no polynomial-time algorithm approximating PHC-COVER, to within a factor d^γ then there is no polynomial-time algorithm approximating TT-MinDNF, PTT-MinDNF and HC-COVER to within a factor $\Omega(d^\gamma)$.*

Proof: Each of the reductions in Theorems 3,4, 22 multiplies the number of variables by at most $2 + \log d/d < 2 + \delta$ (for any constant $\delta > 0$) while preserving (up to additive 1) the approximation factor. Therefore the gap in terms of the new number of variables $d' = O(d)$ is $\Omega(d'^\gamma)$. $\square$

4 Hardness of Approximation

Below we prove hardness of approximating PHC-COVER by presenting a direct reduction from one-round multi-prover proof systems with certain properties to PHC-COVER. We then obtain the claimed result by coupling our reduction with the low-error PCP for **NP** due to Raz and Safra [33]. The reduction simulates a generalization of the reduction to SET-COVER by Lund and Yannakakis [24] on the Boolean hypercube.

4.1 Multi-prover Proof Systems

Following the definition by Bellare *et al.* [5] we distinguish five important parameters of one-round multi-prover proof systems and define the class $\mathrm{MIP}_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ as follows:

Definition 7 *$L \in MIP_1(\ell_r(n), p(n), \ell_a(n), \ell_q(n), \epsilon(n))$ if there exists a probabilistic polynomial-time verifier V, communicating with $p(n)$ provers such that for every $x \in \{0,1\}^n$, V performs the following:*

- *tosses $\ell_r(n)$ random coins obtaining $r \in \{0,1\}^{\ell_r}$,*
- *computes $p(n)$ questions $q(r)_1, \ldots, q(r)_{p(n)}$ each of length at most $\ell_q(n)$,*
- *asks the i-th prover question $q(r)_i$ and gets $p(n)$ answers $a_1, \ldots, a_{p(n)}$ each of length at most $\ell_a(n)$,*
- *computes a predicate $V(x, r, a_1, \ldots, a_{p(n)})$ and accepts if and only if it is 1.*

We also require that V has perfect completeness and soundness bounded by $\epsilon(n)$.

Our reduction will rely on three simple properties of V. The *functionality* property requires that for each $x \in \{0,1\}^n$, $r \in \{0,1\}^{\ell_r}$ and each $a_1 \in \{0,1\}^{\ell_a}$ there is at most one vector $(a_2, a_3, \ldots, a_p)$ such that $V(x, r, a_1, a_2, \ldots, a_p) = 1$. The second property, *uniformity*, requires that for each $i \in [p]$, there exists a set $Q_i \subseteq \{0,1\}^{\ell_q}$ such that queries of V to prover i are uniformly distributed over Q_i. The last, *equality of question space sizes*, requires that $|Q_1| = |Q_2| = \cdots = |Q_p|$. Following Bellare *et al.* [5] we call V *canonical* if it has these three properties.

Similarly, we distinguish analogous parameters for a PCP system. We denote the class $\mathrm{PCP}(\ell_r(n), p(n), \ell_a(n), \ell_q(n), \epsilon(n))$ to be the class of languages decidable by a PCP verifier V that uses $\ell_r(n)$ random bits, generates $p(n)$ questions of length $\ell_r(n)$, gets answers of length $\ell_a(n)$, has perfect completeness and soundness $\epsilon(n)$.

4.2 Packing a Proof System into the Boolean Hypercube

The main tool for creating an approximation gap is a set system $\mathcal{B}_{m,l} = (B; C_1, C_2, \ldots, C_m)$ where m, l are positive integers and for each $i \in [m]$, $C_i \subseteq B$. This set system has the property that if $I \subset [m]$ and $|I| \leq l$, then no union $\bigcup_{i \in I} D_i$ covers B, where D_i equals C_i or its complement.

Lemma 8 ([24]) *There exists $\mathcal{B}_{m,l} = (B; C_1, C_2, \ldots, C_m)$ for $|B| = O(2^l m^2)$ and it can be constructed in time polynomial in $|B|$.*

The main construction of this section is given in the following lemma.

Lemma 9 *If $L \in MIP_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ with a canonical verifier V, then there exists an algorithm $\mathcal{A}$ that given x, produces an instance of PHC-COVER $(S_x, \mathcal{T}_x)$ over $d \leq \ell_r + p(\ell_q + 2^{\ell_a})$ variables such that*

- *if $x \in L$ then PHC-COVER$(S_x, \mathcal{T}_x) = p|Q_1|$, where Q_1 is the question space of the first prover.*

- *if $x \notin L$ then PHC-COVER$(S_x, \mathcal{T}_x) \geq \frac{1}{2}(2\epsilon)^{-1/p}|Q_1|$.*

Moreover, $\mathcal{A}$ runs in time polynomial in n and 2^d.

Proof: As before, let $Q_i \subseteq \{0,1\}^{\ell_q}$ denote the set of questions that V asks prover i and let A_i be the answer space of prover i. Set $s_a = |A_2| + |A_3| + \ldots + |A_p|$ and let $\mathcal{B}_{s_a,l} = (B; C_1, C_2, \ldots, C_{s_a})$ be a set system given by Lemma 8 for l to be specified later. We associate each question $a_i \in A_i$ to prover i for $i \geq 2$ with a unique subset from $\mathcal{B}_{s_a,l}$ that we denote C_{i,a_i}. Let $t_a = \sum_{i \in [p]} |A_i|$ and τ be a function mapping each pair (i, a_i) for $i \in [p]$ and $a_i \in A_i$ to a unique element of $[t_a]$. Now for each setting of a random string $r \in \{0,1\}^{\ell_r}$ and $j \in [t_a]$, we assign a set $C(r,j) \subseteq B$ as follows. Let $(i, a_i) = \tau^{-1}(j)$. If $i \geq 2$ let $C(r,j) = C_{i,a_i}$. For $i = 1$, if there exist $a_2, \ldots, a_p$ such that $V(x, r, a_1, a_2, \ldots, a_p) = 1$, then $C(r,j) = \overline{C_{2,a_2}} \cap \cdots \cap \overline{C_{p,a_p}}$, otherwise $C(r,j) = \emptyset$. In both cases $C(r,j)$ is well-defined since V has the functionality property.

Let $d = \ell_r + p\ell_q + t_a$. We refer to the first ℓ_r variables of the Boolean cube $\{0,1\}^d$ as $y_{r,1}, \ldots, y_{r,\ell_r}$, the next $p\ell_q$ variables as $z_{i,j}$ for $i \in [p]$ and $j \in [\ell_q]$, and the last t_a variables as $z_{A,j}$ for $j \in [t_a]$.

For every $r \in \{0,1\}^{\ell_r}$ and $b \in B$ let $z(r,b)$ be a Boolean vector of length t_a such that $z(r,b)_j = 1$ whenever $b \in C(r,j)$. Furthermore, let $(r,b) = r \cdot q(r)_1 \cdots q(r)_p \cdot z(r,b)$. Let $S_x = \{(r,b) \mid r \in \{0,1\}^{\ell_r},\ b \in B\}$. We now proceed to define the terms. For $i \in [p]$, $q_i \in Q_i$, and $a_i \in A_i$, let $T(i, q_i, a_i)$ be the term that checks that variables of i-th question equal to q_i and that the variable corresponding to answer a_i from prover i is set to 1, or formally

$$T(i, q_i, a_i) = \mathsf{eq}(z_{i,1} \cdots z_{i,\ell_q}, q_i) \wedge z_{A,\tau(i,a_i)} \ .$$

Let $\mathcal{T}_x = \{T(i, q_i, a_i) \mid i \in [p],\ q_i \in Q_i,\ a_i \in A_i\}$. It is easy to verify that term $T(i, q_i, a_i)$ covers a points (r,b) if and only if $q_i = q(r)_i$ (V generates query q_i to prover i on input x and random string r) and $b \in C(r, \tau(i, a_i))$. Therefore the set system $(S_x, \mathcal{T}_x)$ corresponds exactly to the set system created in the generalized version of the reduction to SET-COVER by Lund and Yannakakis [24]. Their analysis implies that for $x \in L$, PHC-COVER$(S_x, \mathcal{T}_x) \leq \sum_{i \in P} |Q_i| = p|Q_1|$ and for $x \notin L$, PHC-COVER$(S_x, \mathcal{T}_x) \geq (1 - \epsilon l^p) l \cdot |Q_1|$ [5]. Therefore by setting $l = (2\epsilon)^{-1/p}$ we will get the stated inapproximability gap of $(2\epsilon)^{-1/p}/(2p)$. The analysis is a straightforward generalization of the analysis by Lund and Yannakakis and we include it for completeness in Appendix A.2. □

4.3 Obtaining Proof Systems with Canonical Verifiers

In this section, we show how to derive canonical multi-prover proofs systems from general PCPs for **NP**. The first step is obtaining a multi-prover system from a PCP. As shown by Bellare, Goldreich, and Safra, (their proof appears in Ta-Shma's paper [34]) identity transformation of a PCP to an MIP (that is, just distributing p queries to p different provers) increases the soundness of the proof system by a factor of at most p^p. That is,

Lemma 10 ([34]) *$PCP(\ell_r(n), p(n), \ell_a(n), \ell_q(n), \epsilon(n)) \subseteq MIP_1(\ell_r(n), p(n), \ell_a(n), \ell_q(n), p^p\epsilon(n))$.*

The next step in our transformation is obtaining the functionality property.

Lemma 11 *If $L \in MIP_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ with a verifier V, then $L \in MIP_1(\ell_r, p{+}1, p\ell_a, p\ell_q, \epsilon)$ with a verifier V' that has the functionality property.*

Proof: To get a verifier V' with the desired property, we add one more prover (which we place first in the enumeration). Given r, V' uses V to generate questions $q_1, \ldots, q_p$, asks all the "old" provers their respective questions, and asks the new prover question $(q_1, q_2, \ldots, q_p)$. Given answers $a_1, \ldots, a_p$ from the "old" provers and an answer $(a'_1, \ldots, a'_p)$ from the new prover, V' accepts if $a'_i = a_i$ for all $i \in [p]$, and $V(x, r, a_1, \ldots, a_p) = 1$. We first observe that, by definition, V' has the functionality property. Next, it is easy to see, that soundness of the new multi-prover system is at most ϵ. Completeness does not decrease since if the first prover answers his questions in the same way as the other p honest deterministic provers, then V' will accept whenever V accepts. Finally, the bounds on the length of queries and answers grow by a factor of at most p. $\Box$

Next we describe how to obtain the last two properties required to get a canonical verifier.

Lemma 12 *If $L \in MIP_1(\ell_r, p, \ell_a, \ell_q, \epsilon)$ with a verifier V, then $L \in MIP_1((p+1)\ell_r, p, \ell_a, \ell_r + \ell_q, \epsilon)$ with a verifier V' that has uniformity and "equality of answer space sizes" properties. Furthermore, if V has the functionality property then V' is canonical.*

Proof: For each $q_i \in Q_i$, let R_{i,q_i} denote the set of random strings for which V generates question q_i for prover i. New verifier V' uses V to generate questions $q_1, q_2, \ldots, q_p$ and then asks questions $((q_1, j_1), (q_2, j_2), \ldots, (q_p, j_p))$ where j_i is an element of $[|R_{i,q_i}|]$ chosen randomly, uniformly, and independently of other choices. It is easy to see that after this modification the sets of possible questions are all of the same size 2^{ℓ_r} and the questions are distributed uniformly. These random bits can be disregarded by honest provers and therefore completeness is not changed. Clearly, randomly and independently chosen bits cannot help dishonest provers and therefore soundness is still bounded by ϵ. Finally, the bound on questions size is at most $\ell_r + \ell_q$ and the number of random bits required is at most $(p+1)\ell_r$. The accepting predicate of V was not changed and thus functionality property is preserved in this transformation. $\Box$

We can now combine these transformations with the following theorem due to Raz and Safra [33],

Theorem 13 ([33]) *For any $\beta \leq 1/4$, $\ell_q(n) \leq \log^\beta n$ there exist fixed positive constants $b_r, b_p, b_q, b_\epsilon$ such that $SAT \in PCP(b_r \log n, b_p, \ell_a(n), b_q \log n, 2^{-b_\epsilon \ell_a(n)})$.*

obtaining the following result:

Lemma 14 *$SAT \in MIP_1(c_r \log n, c_p, \log\log n, c_q \log n, \log^{-c_\epsilon} n)$ with a canonical verifier for some fixed positive constants $c_r, c_p, c_q, c_\epsilon$.*

Proof: We start with the PCP from Theorem 13 and then apply Lemmas 10, 11, and 12 to get that SAT $\in \mathrm{MIP}_1((b_p+2)b_r \log n, b_p+1, b_p\ell_a(n), (b_p b_q + b_r)\log n, b_p^{b_p} 2^{-b_\epsilon \ell_q(n)})$. We now choose $\ell_a(n) = (\log\log n)/b_p$ and obtain the desired result for $c_r = (b_p+2)b_r$, $c_p = b_p + 1$, $c_q = (b_p b_q + b_r)$, and any $c_\epsilon > b_\epsilon / b_p$ ("strictly greater" is to offset the constant factor $b_p^{b_p}$). $\Box$ Hence, by Lemma 9, we get an inapproximability gap of $(2\log n)^{c_\epsilon/c_p}/(2c_p)$, while $d \leq (c_r + c_p c_q + c_p)\log n$ immediately implying Theorem 1.

5 Hardness of Proper PAC Learning with Membership Queries

In this section, we present our hardness results for proper PAC+MQ learning of DNF formulae. We first look at the learning model where the distribution over the input space is not restricted. In this setting our result is based on the hardness result for learning DNF expressions without MQs by Alekhnovich *et al.* [2]. As in their work, we prove a stronger result that shows hardness of learning DNF expressions by a richer[1] class of OR-of-thresholds. Formally restating Theorem 2 we prove the following.

Theorem 15 *If there exists an algorithm $\mathcal{A}$ such that for every Boolean function c, distribution D and ϵ, $\mathcal{A}$, given access to $EX_D(c)$ and $MEM(c)$, runs in time $poly(n, \texttt{DNF-size}(c), 1/\epsilon)$ and with probability at least $3/4$ outputs an OR-of-thresholds formula f such that $\Pr_{x \in D}[f(x) = c(x)] \geq 1 - \epsilon$, then $\mathsf{NP} = \mathsf{RP}$.*

For consistency, we prove an equivalent formulation that CNF expressions are not learnable by AND-of-thresholds. The proof of Alekhnovich *et al.* is based on a reduction to approximating the chromatic number of a graph. Given a graph G, they produce a set of examples such that if the chromatic number of G is "small" then there exists a "small" CNF formula consistent with the examples. Otherwise, if the chromatic number of the underlying graph is "large", then the size of the minimal AND-of-thresholds formula with "small" error on the induced distribution over the examples is "large". Our contribution is to show that we can define (efficiently) values of the target function f on the rest of the hypercube so that in the case of the "small" chromatic number, f can still be represented by a relatively "small" CNF formula. This allows us to answer queries to the membership oracle without any knowledge of a "small" coloring.

5.1 From Coloring to Learning

Given a graph $G = (V, E)$, construct a target function f and a distribution D as follows. Fix some positive integer parameter r. The examples are from $\{0,1\}^{n \times r} = (\{0,1\}^n)^r$.

Definition 16 *Let $G(V, E)$ be a graph with n vertices and m edges. For a vertex v of G, let $i(v)$ denote the index of the vertex (according to some fixed enumeration), let $z(v)$ denote the vector with a 1 in the $i(v)$-th position and 0 everywhere else. For an edge $e = (u, v)$ of G, let $z(e)$ be the vector with a 1 in positions $i(u)$ and $i(v)$.*

For each vector $(v_1, v_2, \ldots, v_r) \in V^r$, we associate a negative example $(z(v_1), \ldots, z(v_r), 0)$. 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

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

Let S^+ denote the positive examples and S^- denote the negative examples. Set $S = S^+ \cup S^-$. The distribution D is uniform over the above set of examples S.

On the rest of the hypercube we define f as follows: let $x = (x^1, \ldots, x^r)$ be a point not in $S^+ \cup S^-$. If for some $i \leq r$, there exist indices j_1 and j_2 such that vertices with indices

[1]Richer in the sense that all size k DNF formulae can be expressed by size k OR-of-thresholds.

j_1 and j_2 are not connected by an edge in G, then $f(x) = 0$. We call this set of points *non-graph* points. Otherwise, let $f(x) = 1$.

We now prove that if the chromatic number $\chi(G)$ is small, then there exists a small CNF formula equal to f.

Lemma 17 *If $\chi(G) \leq n^\lambda$, then there is a CNF formula of size at most $n^{r\lambda} + r|E|$ equal to f.*

Proof: [Sketch] 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 $g(x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{\chi} \bigvee_{j \notin I_i} x_j$. We then define a CNF formula H on $r \cdot n$ variables that rejects all the non-graph points

$$H(x^1, \ldots, x^r) = \bigwedge_{k \leq r;\ (u,w) \notin E} (\overline{x^k_{i(u)}} \vee \overline{x^k_{i(w)}}) \ ,$$

and the CNF formula F that rejects all the points in S^-

$$F(x^1, \ldots, x^r) = \bigvee_{k=1}^{r} g(x^k_1, \ldots, x^k_n) = \bigvee_{k=1}^{r} \bigwedge_{i=1}^{\chi} \bigvee_{j \notin I_i} x^k_j \ .$$

It is not hard to verify that the set S^- and the non-graph points are exactly the points where the target function is negative and therefore $F \wedge H$ is consistent with the above definition of the target function. 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 expression for F yields a CNF formula with at most $\chi(G)^r \leq n^{\lambda r}$ clauses. So $F \wedge H$ can be written as a CNF formula satisfying the conditions of the lemma. $\square$

For the case when the chromatic number is large, we can simply use the original analysis [2] since our definition of the target function on points with weight 0 does not make the task of finding an AND-of-thresholds formula with small error any easier or harder. Namely, the following was proved by Alekhnovich *et al.* :

Lemma 18 ([2]) *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 $\frac{1}{n^{2r\gamma+4}}$ with respect to D.*

Combining the two cases (Lemmas 17 and 18) gives us the following claim.

Lemma 19 *Suppose that CNF formulas are learnable by AND-of-thresholds in time $O(n^k \cdot s^k \cdot (\frac{1}{\epsilon})^k)$, where $k > 1$. Then there exists a randomized algorithm for approximating the chromatic number of a graph within a factor of $n^{1-k/10}$ in time $O(n^{9k+1})$. Moreover, the algorithm will always give a valid answer for $\chi \geq n^{1-k/10}$.*

Proof: Fix $\delta = 1/4$, $\epsilon = \frac{1}{n^6}$, and $r = 10k$. For a graph G let the target function f and the distribution D be defined as in Section 5.1. Run the learning algorithm with respect to distribution D and with queries answered according to f. If it does not terminate after n^{9k+1} steps, output "$\chi \geq n^{1-k/10}$". Otherwise, let h be the hypothesis the algorithm outputs. Calculate the error ϵ_h of h with respect to the distribution D. If $\epsilon_h < \frac{1}{n^6}$ output "$\chi \leq n^{k/10}$", otherwise output "$\chi \geq n^{1-k/10}$". We claim that this algorithm works with probability $\geq \frac{3}{4}$ for sufficiently large n's in approximating $\chi \leq n^{k/10}$, and works perfectly for $\chi \geq n^{1-k/10}$.

If $\chi \leq n^{k/10}$, then Lemma 17 implies that $s \leq n^{\frac{1}{10k}10k} + 10k|E| \leq 10k \cdot n^2$. Hence the running time with probability $\geq \frac{3}{4}$ is at most $O((10k \cdot n)^k (10k \cdot n^2)^k (n^6)^k) = O(n^{9k}) < n^{9k+1}$ for sufficiently large n, and the output is supposed to have an error $< \epsilon = \frac{1}{n^6}$. Hence the algorithm outputs "$\chi \leq n^{1/10k}$" with probability $\geq \frac{3}{4}$ in this case.

If $\chi \geq n^{1-k/10}$, then by Lemma 18, 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 $< \epsilon = \frac{1}{n^6} = \frac{1}{n^{2\cdot 10k(k/10)+4}}$. In this case the running time of the algorithm is at least

$$\frac{1}{2\chi r}\left(\frac{\chi-1}{\ln n}\right)^r \geq \frac{1}{20\chi \cdot k}\left(\frac{n^{1-k/10}-1}{\ln n}\right)^{10k} \geq \frac{1}{20nk}\left(\frac{n^{1-k/10}}{2\ln n}\right)^{10k} \geq n^{10k-3}$$

for sufficiently large n. Hence if the algorithm terminates in $n^{9k+1} < n^{10k-3}$ steps, its error will be larger than ϵ, and the algorithm outputs "$\chi \geq n^{1-k/10}$" with probability 1 in this case. □

Finally, we will require the following hardness result due to Feige and Kilian [14]:

Theorem 20 *[14] For any constant $\gamma > 0$, 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 $\chi(G) \leq O(N^\gamma)$ and if f is unsatisfiable then $\chi(G) \geq \Omega(N^{1-\gamma})$. The reduction has zero-sided error.*

Combining Lemma 19 and Theorem 20 gives Theorem 15.

5.2 Proper PAC Learning with Respect to the Uniform Distribution

We now show a simple application of the hardness of TT-MinDNF to the hardness of proper PAC learning of DNF expressions restricted to the uniform distribution over $\{0,1\}^n$. It is a very strong model in which, as proved by Jackson [19], DNF expressions are learnable non-properly.

It has been observed by Allender *et al.* that TT-MinDNF naturally reduces to exact learning of DNF with MQs [3]. We further this observation by reducing TT-MinDNF to PAC+MQ learning with respect to the uniform distribution.

Theorem 21 *There exists a constant $\gamma > 0$ such that, if there exists an algorithm $\mathcal{A}$ that for every Boolean function c and $\epsilon > 0$, $\mathcal{A}$, given access to $EX_D(c)$ and $MEM(c)$, runs in time $poly(n, \texttt{DNF-size}(c), 1/\epsilon)$ and, with probability at least $3/4$, outputs a DNF formula f of size $s \leq \log^\gamma n \cdot \texttt{DNF-size}(c)$ such that $\Pr_{x\in\{0,1\}^n}[f(x) = c(x)] \geq 1 - \epsilon$, then $\mathsf{NP} = \mathsf{RP}$.*

Proof: We reduce from TT-MinDNF and let γ be the constant from Theorem 1. Given the truth table of a function f over $d = \log n$ variables, we let the target concept be $c(x) = f(x_1 \cdots x_d)$. Clearly, $s = \texttt{DNF-size}(c) \leq 2^{\log n} = n$. The definition of $c(x)$ implies that MEM(c) can be efficiently simulated given the truth table of f. We then set $\epsilon = 1/(2n)$ and $\delta = 1/2$. A strongly proper algorithm on this input will (with probability at least 1/2) produce in time polynomial in $n = 2^d$ a DNF formula h of size $s \leq \log^\gamma n \cdot \texttt{DNF-size}(c)$ that

$\frac{1}{2n}$-approximates c. Now we choose a vector y of length $n-d$ randomly and uniformly and let h_y be the projection of h to first d variables with the last $n-d$ variables set to y. We claim that with probability at least 1/2, $f \equiv h_y$. To see this, note that

$$\frac{1}{2n} \geq \mathbf{Pr}_{x \in \{0,1\}^n}[c(x) \neq h(x)] = \mathbf{E}_{y \in \{0,1\}^{n-d}}\left[\mathbf{Pr}_{z \in \{0,1\}^d}[f(z) \neq h_y(z)]\right] ,$$

and therefore for at least 1/2 of y's, $\mathbf{Pr}_{z \in \{0,1\}^d}[f(z) \neq h_y(z)] = 0$, that is $f(z) = h_y(z)$ for all z. For each y, the number of terms in h_y is at most s and therefore with probability at least 1/4, s approximates `DNF-size`(f) within d^γ. $\square$

6 Conclusions and Open Problems

In this work we have conclusively answered the question of proper learning of DNF expressions in the PAC+MQ model. We also made some progress towards answering a similar question when the distribution is restricted to be uniform. It is easy to see that in the uniform case finding a DNF hypothesis $\log(1/\epsilon)$ times larger than the target can be done in time $n^{\log(1/\epsilon)}$ and therefore, is unlikely to be **NP**-hard. This means that substantial improvements of the result will have to be based on different (probably stronger) assumptions. A more important direction would be to reduce restrictions on the output hypothesis in the hardness results (with the final goal being the circuit representation).

It would be also interesting to close the gap between $O(d)$ and d^γ for approximating TT-MinDNF. As it follows from our reduction, stronger hardness results can be obtained by using query and randomness-efficient PCPs. More specifically, γ is at most $\frac{\log(1/\epsilon)}{\log(|Q|+\ell_r)}$ where the soundness ϵ is at least $1/\log^p(n)$ (in a PCP with constant number of queries and perfect completeness). Most known PCPs are not query-efficient and known query-efficient constructions are based on the long code that requires a "large" number of random bits (c.f. [17]). Therefore in the reductions from the known PCPs for **NP** the fraction (and subsequently γ) is at most a small constant.

7 Acknowledgments

I am grateful to Leslie Valiant for his advice and encouragement of this research. I would also like to thank Alex Healy, Shaili Jain, and Emmanuele Viola for careful proofreading and valuable comments on the earlier versions of this paper.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, 11(1):5–33, July 2005.

[2] M. Alekhnovich, M. Braverman, V. Feldman, A. Klivans, and T. Pitassi. Learnability and automizability. In *Proceeding of FOCS '04*, pages 621–630, 2004. Extended version available at `www.eecs.harvard.edu/∼vitaly/papers/properDNF.pdf`.

[3] E. Allender, L. Hellerstein, T. Pitassi, and M.Saks. On the complexity of finding minimal representations of boolean functions. 2004. Unpublished.

[4] D. Angluin and M. Kharitonov. When won't membership queries help? *Journal of Computer and System Sciences*, 50(2):336–355, 1995.

[5] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *STOC '93*, pages 294–304, 1993.

[6] A. Blum and S. Rudich. Fast learning of k-term DNF formulas with queries. *Journal of Computer and System Sciences*, 51(3):367–373, 1995.

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

[8] O. Coudert and J. Madre. METAPRIME, an Interactive Fault-Tree Analyser. *IEEE Transactions on Reliability*, 43(1):121–127, 1994.

[9] O. Coudert and T. Sasao. *Two-level logic minimization.* Kluwer Academic Publishers, 2001.

[10] S. Czort. The complexity of minimizing disjunctive normal form formulas. Master's thesis, University of Aarhus, 1999.

[11] I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal of Computing*, 34(5):1129–1146, 2005.

[12] P. Erdös, P. Frankl, and Z. Furedi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79–89, 1985.

[13] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[14] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proceedings of Conference on Computational Complexity (CCC-96)*, pages 278–289, May 24–27 1996.

[15] J.F. Gimpel. A method for producing a boolean function having an arbitrary prescribed prime implicant table. *IEEE Transactions on Computers*, 14:485–488, 1965.

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

[17] Johan Håstad and Subhash Khot. Query Efficient PCPs with Perfect Completeness. *Theory of Computing*, 1(7):119–148, 2005.

[18] Lisa Hellerstein and Vijay Raghavan. Exact learning of DNF formulas using DNF hypotheses. In ACM, editor, *Proceedings of STOC '02*, pages 465–473, 2002.

[19] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.

[20] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory*, 10:363–377, 1964.

[21] 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.

[22] 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.

[23] H. Liu. Routing table compaction in ternary cam. *IEEE Micro*, 22(1):58–64, 2002.

[24] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[25] W. Masek. Some NP-complete set covering problems. unpublished, 1979.

[26] E. L. Jr. McCluskey. Minimization of Boolean Functions. *Bell Sys. Tech. Jour.*, 35:1417–1444, 1956.

[27] N. Nisan and A. Wigderson. Hardness versus randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[28] 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.

[29] W. J. Paul. Boolesche minimalpolynome und überdeckungsprobleme. *Acta Informatica*, 4:321–336, 1974.

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

[31] W.V. Quine. The problem of simplifying truth functions. *Americal Mathematical Monthly*, 59:521–531, 1952.

[32] W.V. Quine. A way to simplify truth functions. *Americal Mathematical Monthly*, 62:627–631, 1956.

[33] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of STOC '97*, pages 475–484, 1997.

[34] A. Ta-Shma. A Note on PCP vs. MIP. *Information Processing Letters*, 58(3):135–140, 1996.

[35] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of STOC '01*, pages 453–461, 2001.

[36] C. Umans. The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, 63(4):597–611, 2001.

[37] C. Umans, T. Villa, and A. L. Sangiovanni-Vincentelli. Complexity of two-level logic minimization. Technical Report UCB/ERL M04/45, UC Berkeley, October 2004.

[38] S. Vadhan. Lecture notes on pseudorandomness. Available at `http://www.courses.fas.harvard.edu/∼cs225/`, 2004.

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

[40] L. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 560–566, 1985.

A Appendix

A.1 From TT-MinDNF to HC-COVER

We give a simple reduction proving that covering the whole hypercube by terms from a restricted set $\mathcal{T}$ is not substantially easier than covering a subset of the hypercube. Note that this reduction is not required as a step in our main result.

Theorem 22 *There exists an algorithm that, given the truth table of a function f on d variables and an integer $r \geq 1$, produces a set of terms $\mathcal{T}$ over $d+r$ variables such that there exists a C-term DNF expression equal to f, if and only if, $\{0,1\}^{d+r}$ can be covered by $2^r C + |f^{-1}(0)|$ terms from $\mathcal{T}$. The algorithm runs in time $2^{O(r+d)}$.*

Proof: The idea of the proof is to create $\mathcal{T}$ that contains all the terms consistent with f and terms that cover $\neg f$. As in the proof of Theorem 4, we will replicate f many times to preserve the approximation ratio .

Our instance of the HC-COVER problem is over $d+r$ variables where we refer to the first d variables as $x_1, \ldots x_d$ and to the next r variables as $y_1, \ldots, y_r$. Let $\mathcal{T}^f$ be the set of all the terms consistent with f (we can take only prime implicants of f, but this is not essential). For $d \in \{0,1\}^r$ let $\mathcal{T}^f_p = \{T \wedge \mathtt{eq}(y,p) \mid T \in \mathcal{T}^f\}$, let $S = f^{-1}(0)$ and $\mathcal{T}^{\neg f} = \{\mathtt{eq}(x,a) \mid a \in S\}$. Then we define

$$\mathcal{T} = \mathcal{T}^{\neg f} \bigcup \left(\bigcup_{p \in \{0,1\}^r} \mathcal{T}^f_p \right).$$

We claim that there exists a C-term DNF equal to f, if and only if, there exists a set $\mathcal{S} \subseteq \mathcal{T}$ of size $C2^r + |S|$ that covers $\{0,1\}^{r+d}$. Let $\mathcal{S}^f$ be a set of C terms such that $\bigvee_{T \in \mathcal{S}^f} T = f$.

Then it is clear that $\mathcal{S} = \mathcal{T}^{\neg f} \bigcup \{T \wedge \mathtt{eq}(y,p) \mid p \in \{0,1\}^r,\ T \in \mathcal{S}^f\}$ covers $\{0,1\}^{r+d}$, has size $C2^r + |S|$, and includes only terms from $\mathcal{T}$.

For the other direction, let $\mathcal{S} \subseteq \mathcal{T}$ be a cover for $\{0,1\}^{d+r}$. We observe that the only way to cover $S \times \{0,1\}^r$ is by including all the terms of $\mathcal{T}^{\neg f}$ in $\mathcal{S}$. For each $p \in \{0,1\}^r$, let $\mathcal{S}_p = \mathcal{S} \cap \mathcal{T}^f_p$. Only terms in $\mathcal{S}_p$ cover the subset $f^{-1}(1) \times \{p\}$ and therefore $h_p(x) = \bigvee_{T \in \mathcal{S}_p} T(x \cdot p)$ equals exactly $f(x)$. All the $\mathcal{S}_p$'s are disjoint and are disjoint from $\mathcal{T}^{\neg f}$. Therefore if $|\mathcal{S}| \leq C2^r + |S|$, then $\mathtt{DNF\text{-}size}(f) \leq C$. □

As with Theorem 4 we obtain the following corollary.

Corollary 23 *If HC-COVER can be approximated within $h(d) = o(d)$ in time $t(d)$, then TT-MinDNF can be approximated within $h(2d + \log d) + 1$ in time $t(2d + \log d) + 2^{O(d)}$.*

A.2 Analysis for the Proof of Lemma 9

First, let $x \in L$ and let $\bar{P}$ be an honest deterministic prover. For $q_i \in Q_i$ denote by $P_i(q_i)$ the answer given by P_i to query q_i and set $\mathcal{S} = \{T(i, q_i, P_i(q_i)) \mid i \leq p,\ q_i \in Q_i\}$. For every point (r, b) and $i \leq p$, let $a'_i = P_i(q(r)_i)$. By perfect completeness of V, $V(x, r, \bar{a}') = 1$ and therefore $C(r, \tau(a'_1, 1)) = \overline{C_{2,a'_2}} \cap \cdots \cap \overline{C_{p,a'_p}}$, i.e, $\cup_{i \leq p} C(r, \tau(a'_i, i)) = B$. This means that for some j, $b \in C(r, \tau(a'_j, j))$ and therefore $(r, b) \in T(j, q(r)_j, a'_j)$. This means that $\mathcal{S}$ is a collection of size $p|Q_1|$ that covers S_x.

Let $x \notin L$ and $\mathcal{S} \subseteq \mathcal{T}_x$ be a cover for S_x. For a random string r and a set $C \subseteq B$, denote by (r, C) the set $\{(r, b) \mid b \in C\}$ and let $\mathcal{S}_r = \{T(i, q(r)_i, a_i) \mid i \leq p,\ T(i, q(r)_i, a_i) \in \mathcal{S}\}$, in other words $\mathcal{S}_r$ includes the terms from $\mathcal{S}$ that cover (r, B). We say that r is *good* if $|\mathcal{S}_r| \leq l$ and *bad* otherwise. Let δ be the fraction of good r's.

Claim 24 *There exists a prover $\bar{P}$ such that V will accept with probability δ / l^p.*

Proof: We define $\bar{P}$ with the following strategy: prover P_i on query q_i chooses a_i from the set $A^i_{q_i} = \{a \mid T(i, q_i, a) \in \mathcal{S}\}$ randomly and uniformly (this set cannot be empty). Note that for every r, $\sum_i |A^i_{q(r)_i}| = |\mathcal{S}_r|$. If r is good, then $|\mathcal{S}_r| \leq l$ and hence there should exist $a'_1, \ldots, a'_p$ such that for every $i \leq p$, $a'_i \in A^i_{q(r)_i}$ and $V(x, r, a'_1, \ldots, a'_p) = 1$. To prove this, assume that for every $a_1 \in A^1_{q(r)_1}$, there exists $j(a_1)$ such that $V(x, r, a_1, \ldots, a_p) = 1$ but $a_{j(a_1)} \notin A^{j(a_1)}_{q(r)_{j(a_1)}}$. Then

$$T(1, q(r)_1, a_1) \cap (r, B) = (r, C(1, \tau(1, a_1)) = (r, \bigcap_{i \geq 2} \overline{C_{i,a_i}}) \subseteq (r, \overline{C_{j(a_1), a_{j(a_1)}}}) \ .$$

This implies that

$$\left(\bigcup_{a_1 \in A^1_{q(r)_1}} \overline{C_{j(a_1), a_{j(a_1)}}} \right) \bigcup \left(\bigcup_{i \geq 2,\ a_i \in A^i_{q_i}} C_{i, a_i} \right) = B \ .$$

We obtained a union of at most l sets from $\mathcal{B}_{s_a, l}$ that does not include a set and its complement, and covers B. This contradicts the definition of $\mathcal{B}_{s_a, l}$, proving the existence of $a'_1, \ldots, a'_p$ as above. For good r's and each i, $|A^i_{q(r)_i}| \leq l$ and therefore, the probability that each P_i will answer with a'_i is at least l^{-p}. Hence this strategy has success probability at least δ / l^p. □

Claim 25 $|\mathcal{S}| \geq (1-\delta)l|Q_1|$.

Proof: For each bad r, $|\mathcal{S}_r| \geq l$ and therefore $\sum_r |\mathcal{S}_r| = (1-\delta)2^{\ell_r}l$. On the other hand, each term $T(i, q_i, a_i) \in \mathcal{S}$ appears once in all the sets for which $q_i = q(r)_i$. The uniformity property of V implies that each query q_i is asked for exactly $2^{\ell_r}/|Q_i| = 2^{\ell_r}/|Q_1|$ different r's (the last equality follows from the equality of question space sizes property). This implies that

$$|\mathcal{S}| = \frac{|Q_i|}{2^{\ell_r}} \sum_r |\mathcal{S}_r| \geq \frac{|Q_i|}{2^{\ell_r}}(1-\delta)2^{\ell_r}l = (1-\delta)l|Q_1| \ .$$

□

By Claim 24 and soundness of V, we get that $\delta \leq \epsilon \cdot l^p$. By Claim 25, this implies that $|\mathcal{S}| \geq (1-\epsilon l^p)l|Q_1|$.

B Reduction from Hypergraph Vertex Cover to PHC-COVER

Our hardness of approximation result for TT-MinDNF can also be obtained via a reduction from the problem of minimizing the size of vertex cover for k-uniform hypergraphs. This reduction only gives the desired inapproximability factor under the assumption that **NP** is not in quasipolynomial time but is substantially simpler than the reduction from multi-prover systems. The main tool we use in the reduction is families of sets in which no set is covered by k others.

B.1 Vertex Cover for k-uniform Hypergraphs

A k-uniform hypergraph $H = (V, E)$ consists of a set of vertices V and a collection E of k-element subsets of V called hyperedges. A *vertex cover* of H is a subset $S \subseteq V$ such that every hyperedge in E intersects S. The Ek-Vertex-Cover problem is the problem of finding a minimum size vertex cover on a k-uniform hypergraph. The problem is alternatively called the minimum hitting set problem with sets of size k and is equivalent to the set cover problem where each element of the universe occurs in exactly k sets.

The first explicit hardness result shown for Ek-Vertex-Cover was due to Trevisan who showed an inapproximability factor of $k^{1/19}$ [35] (and a comparable result is implicit in Feige's proof of inapproximability of SET-COVER [13]). As we aim at obtaining a large inapproximability factor for covering the hypercube, we are interested in results that hold for large values of k. The first result stating the range of k explicitly is due to Dinur *et al.* [11] who give the following theorem[2].

Theorem 26 ([11]) *There exists some $b > 0$ such that unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log\log(n)})$, *there is no polynomial time algorithm for approximating Ek-Vertex-Cover for $k \leq (\log M)^{1/b}$ to within a factor of $\lfloor k/2 \rfloor - 0.01$, where M is the number of hyperedges in the k-uniform hypergraph.*

Remark 27 *It can be easily seen from the proof of this theorem, that the number of vertices N is smaller than M and therefore the result can be stated with the number of vertices N instead of M.*

[2] In fact, weaker but sufficient for our purposes result is implicit in the above-mentioned work of Feige [13].

B.2 Union-free Families

A family of sets $\mathcal{F}$ is called k-union-free if $A_0 \not\subseteq A_1 \cup A_2 \cup \cdots \cup A_k$ for all distinct $A_0, A_1, \ldots, \mathcal{A}_k \in \mathcal{F}$. They were introduced by Kautz and Singleton [20] (and then rediscovered by Erdös *et al.* [12]). Commonly-used *combinatorial designs* introduced by Nisan and Wigderson [27] are in particular union-free families of sets (for an appropriate limit on the size of intersections). The following theorem is obtained by derandomizing a straightforward randomized construction using the method of conditional probabilities (c.f. [38][Lecture 21]).

Theorem 28 *There exists a k-union-free family of sets over $[d]$ of size m for $d = O(k^2 \log m)$. Moreover, such $\mathcal{F}$ can be constructed in time poly(m, d).*

B.3 Reduction to PHC-COVER

Below we present a reduction from SET-COVER with each point occurring in k sets to PHC-COVER.

Theorem 29 *There exists a polynomial-time algorithm that, given a k-uniform hypergraph $H = (V, E)$ with $|V| = N$, produces an instance $(S, \mathcal{T})$ of PHC-COVER over $d = O(k^2 \log N)$ variables such that H has a vertex cover of size C, if and only if, $(S, \mathcal{T})$ has a cover of size C. The algorithm runs in time $O(N2^d)$.*

Proof: We first transform the k-uniform hypergraph vertex cover problem to its dual set cover problem with each point occurring in k sets. That is, for $v \in V$, let $S_v = \{e \mid e \in E, v \in e\}$ and $\mathcal{S} = \{S_v \mid v \in V\}$. Then $(E, \mathcal{S})$ is an instance of set cover with cover of size C. Let $\mathcal{F} = \{P_1, \ldots, P_N\}$ be a k-union-free family. By Theorem 28, such $\mathcal{F}$ exists and can be efficiently constructed for $d = O(k^2 \log N)$. For each $v \in V$, let $P_v = P_i$ if v is the i-th vertex in V according to some fixed enumeration. For any set $P \subseteq [d]$, let $\chi(P)$ be a characteristic vector of P, that is vector with $\chi(P)_i = 1$ when $i \in P$ and $\chi(P)_i = 0$, otherwise. For each $e \in E$, let $x^e = \chi(\cup_{v \in e} P_v)$. We define $\mathcal{T} = \{\mathsf{eq}(x, \chi(P_v)) \mid v \in V\}$, and define $S = \{x^e \mid e \in E\}$.

To prove the correctness of this reduction all we need to show is that for each $e \in E$ and $v \in V$, $\mathsf{eq}(x, \chi(P_v))$ covers x^e, if and only if, $e \in S_v$ or, in other words, $v \in e$. If $v \in e$ then $P_v \subseteq \cup_{u \in e} P_u$ and, therefore $x^e_i = 1$ for all $i \in P_v$. This implies that $\mathsf{eq}(x, \chi(P_v)) = \wedge_{i \in P_v} x_i$ accepts x^e. On the other hand, if $v \notin e$ for each $u \in e$ then, by the properties of $\mathcal{F}$, $P_v \not\subseteq \cup_{u \in e} P_u$. This implies that $\mathsf{eq}(x, \chi(P_v))$ will not accept x^e. □

Corollary 30 *There exists a constant $\gamma > 0$ such that, unless* **NP** $\subseteq$ **DTIME**$(n^{\log(n)})$, *there is no polynomial-time algorithm approximating PHC-COVER to within a factor d^γ, where d is the number of variables in the PHC-COVER instance.*

Proof: Given an instance of SAT on n variables, the reduction of Theorem 26 produces an instance of Ek-Vertex-Cover on $N = n^{O(\log \log n)}$ vertices for $k = (\log N)^{1/c}$, where $c > 1$. The gap in vertex cover sizes is αk ($\alpha \approx 1/2$). Then reduction in Theorem 29 will produce an instance of PHC-COVER over $d = O(k^2 \log N) = O((\log N)^{1+2/c})$ with the same gap of αk, that in terms of d, is $O(d^{1/(b+2)}) > d^\gamma$ for any constant $\gamma < \frac{1}{2+b}$ (and large enough d). The running time of the reduction and the produced instance are both bounded by

$$2^{O(d)} = 2^{O((\log N)^{1+2/c})} = n^{O((\log n)^{2/c} (\log \log n)^{1+2/c})} = O(n^{\log n})$$

for $c > 2$. □