Learnability of DNF with Representation-Specific Queries

Liu Yang
Machine Learning Department
Carnegie Mellon University
liuy@cs.cmu.edu

Avrim Blum
Department of Computer Science
Carnegie Mellon University
avrim@cs.cmu.edu

Jaime Carbonell
Language Technologies Institute
Carnegie Mellon University
jgc@cs.cmu.edu

ABSTRACT

We study the problem of PAC learning the class of DNF formulas with a type of natural pairwise query specific to the DNF representation. Specifically, given a pair of positive examples from a polynomial-sized sample, we consider boolean queries that ask whether the two examples satisfy at least one term in common in the target DNF, and numerical queries that ask how *many* terms in common the two examples satisfy. We provide both positive and negative results for learning with these queries under both uniform and general distributions.

For boolean queries, we show that the problem of learning an arbitrary DNF target under an arbitrary distribution is no easier than in the traditional PAC model. However, on the positive side, we show that under the uniform distribution, we can properly learn any DNF formula with $O(\log(n))$ relevant variables, any DNF formula where each variable appears in at most $O(\log(n))$ terms, and any DNF formula having at most $2^{O(\sqrt{\log(n)})}$ terms. Under general distributions, we show that 2-term DNFs are efficiently properly learnable as are disjoint DNFs.

For numerical queries, we show we can learn arbitrary DNF formulas under the uniform distribution; in the process, we give an algorithm for learning a sum of monotone terms from labeled data only. Numerical-valued queries also allow us to properly learn any DNF with $O(\log(n))$ relevant variables under arbitrary distributions, as well as DNF having $O(\log(n))$ terms, and DNF for which each example can satisfy at most $O(1)$ terms.

Other possible generalizations of the query include allowing the algorithm to ask the query for an arbitrary number of examples from the sample at once (rather than just two), or allowing the algorithm to ask the query for examples of its own construction; we show that both of these generalizations allow for efficient proper learnability of arbitrary DNF functions under arbitrary distributions.

Categories and Subject Descriptors

F.2.0 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*General*

ITCS'13, January 9–12, 2012, Berkeley, California, USA.

General Terms

Algorithms, Theory

Keywords

Learning DNF, PAC Learnability, Efficient Learning, Queries

1. INTRODUCTION

Consider a bank aiming to use machine learning to identify new instances of financial fraud. To do so, the bank would ask financial experts to label past transactions as fraudulent or not, and then run a learning algorithm on the resulting labeled data. However, this learning problem might be quite difficult because of the existence of multiple intrinsic types of fraud, with each positive example perhaps involving multiple types of fradulent behavior. For instance, the fraud may involve identity theft, unauthorized account access, security-code violations, overdraft and/or insider security breaches. That is, the target concept would be in the general case a DNF formula, a class for which no efficient algorithms are known.

Yet in such cases, perhaps the experts performing the labeling could be called on to provide a bit more information. In particular, suppose that given two positive examples of fraud, the experts could indicate whether or not the two examples are *similar* in the sense of having at least one intrinsic type of fraud (at least one term) in common, such as both involving unauthorized account access, or both involving internal blockage of account monitoring (inside job). Or perhaps the experts could indicate *how* similar the examples are (how many terms in common they satisfy), stating the number of common ways in which the fraud occurred. This is certainly substantially more information. Can it be used to learn DNF formulas and their natural subclasses efficiently?

Fraud detection is just one of many potential applications of DNF learning in which pairwise queries are natural. Others applications include medical multi-factor medical diagnosis, where patients suffer from several ailments and a full diagnosis is required to select optimal treamtent, and not aggravate one condition while treating another.

In our work, we study the problem of learning DNF formulas and other function classes using such pairwise, representation-dependent queries. Specifically, we consider queries of the form, "Do these two positive examples satisfy at least one term in common in the target DNF formula?" (we call these *boolean similarity queries*) and "How many terms in common do these two positive examples satisfy?" (we call these *numerical similarity queries*).

These queries must be over data in a labeled sample drawn from the distribution; we assume membership queries are *not* allowed. Our motivation comes from scenarios such as fraud detection, where

the boolean-vector representation of an example is just a projection of the actual data object (the fraudulent transaction), so that the human oracle cannot necessarily make sense of artificially-constructed boolean vectors, as they need to observe the actual data objects in order to answer the query. Similarly, in the problem of drug design, the span of constructable examples is highly constrained, so that general membership queries may not be feasible.

We show, for these types of queries, a number of both positive and negative results for learning DNF formulas and interesting subclasses of them. For example, we show that under the uniform distribution, with the boolean-valued queries, we can properly learn any DNF with $O(\log(n))$ relevant variables, any DNF where each relevant variable appears in at most $O(\log(n))$ terms, and any DNF having at most $2^{O(\sqrt{\log(n)})}$ terms. Under arbitrary distributions, we can properly learn 2-term DNF and disjoint DNF, but we find that the general problem of learning an arbitrary DNF target under an arbitrary distribution is no easier than in the traditional PAC model. With numerical queries, under the uniform distribution we can learn arbitrary DNF formulas, and under arbitrary distributions we can can properly learn any DNF with either $O(\log(n))$ relevant variables, $O(\log(n))$ terms, or for which each example can satisfy at most $O(1)$ terms.

Our model can also be viewed in the context of the paradigm of learning with kernels. A kernel is a pairwise numerical similarity function over examples that is presumed to be available at both training and test time. However, suppose this kernel is computationally expensive to run, or even is a human expert, and so will not be available when the learned hypothesis is fielded. Can one use this kernel to produce a good hypothesis defined only over the base features? Our model can be viewed as addressing this question in the context of DNF, for a particular type of kernel function.

1.1 Model

In this setting, we suppose there is a DNF formula $f : \{0,1\}^n \to \{-1,+1\}$, where $f = T_1 \vee \ldots \vee T_t$ for conjunctions $T_i : \{0,1\}^n \to \{-1,+1\}$. There is additionally a distribution D over $\{0,1\}^n$; some of our results deal specifically with D uniform, while others leave D to be arbitrary. The learning algorithm is given parameters ϵ and δ, and is allowed access to a $poly(n)$-sized i.i.d. sample of data points (examples) $x_1, x_2, \ldots, x_{poly(n)}$ drawn from D, along with their target labels $f(x_1), f(x_2), \ldots, f(x_{poly(n)})$.

The algorithm can additionally make certain *queries*, which vary in type among the results below. The two main types of queries we study are boolean-valued queries and numerical-valued queries, defined as follows. For any x_i, x_j in the data set, the boolean-valued query $k(x_i, x_j)$ takes value 1 if $\exists \ell \leq t$ such that $T_\ell(x_i) = T_\ell(x_j) = 1$, and otherwise takes value 0. Similarly, for any x_i, x_j in the data set, the numerical-valued query

$$K(x_i, x_j) = \sum_{\ell \leq t} I[T_\ell(x_i) = T_\ell(x_j) = 1].$$

Thus, the boolean-valued query indicates whether the two examples satisfy some term in common, while the numerical-valued query indicates the precise number of terms they satisfy in common.

We say it is possible to learn a family of DNF formulas H with a given type of query if there exists an algorithm that uses the random data and the given type of query and, for any $f \in H$, in time polynomial in n, $1/\epsilon$, and $\log(1/\delta)$, produces a classifier $h : \{0,1\}^n \to \{-1,+1\}$ such that, with probability at least $1-\delta$, $err(h) = P_{x \sim D}(h(x) \neq f(x)) \leq \epsilon$. We further say it is possible to *properly* learn H with the given type of query if there exists such an algorithm for which h is guaranteed to be an element of H.

Note that we do not allow membership queries in our main results below (membership queries allow an algorithm to generate its own example and ask the oracle the label of it). We do have one general result (learning arbitrary DNF under arbitrary distributions) based on an analogue of membership queries in this model (see Appendix A).

1.2 Related Work

There have been many positive results on learning $poly(n)$-term DNF formulas with membership queries. Two landmark results are [Ang88] showing that monotone DNF formulas can be properly learned under arbitrary distributions using membership queries, and [Jac94] giving an efficient membership-query algorithm for the (improper) learning of DNF formulas under the uniform distribution.

The problem of learning DNF from random examples without queries has remained open over decades since Valiant introduced the PAC (Probably Approximately Correct) learning model [Val84]. It is not known whether we can efficiently learn DNF from random examples alone and there are known hardness results for properly learning DNF. In fact, [PV88] showed that even 2-term DNF formulas are not properly learnable from labeled data alone unless NP = RP. The fastest algorithms so far for learning $poly(n)$-term DNF formulas in the PAC model requires time $2^{\tilde{O}(n^{1/3})}$ [KS01].

For the problem of PAC learning DNF formulas under the uniform distribution, no polynomial-time algorithms are known. The fastest known algorithm for learning $poly(n)$-term DNF under the uniform distribution requires time $n^{O(\log n)}$ [Ver90]. At the heart of this problem lies the problem of efficiently learning k-juntas (an arbitrary boolean function depending on an unknown set of only k out of n boolean variables) for $k = O(\log n)$. The best known algorithm for learning k-juntas from uniform random examples is given by [Val12], building on [MOS04], which requires time roughly $n^{0.6k}$.

Some partial positive results have been obtained for special cases of monotone DNF formulas. [Ser04] provided an algorithm that learns any $2^{O(\sqrt{\log n})}$-term monotone DNF under the uniform distribution to any constant accuracy in $poly(n)$ time, and [OS07] gave an algorithm that learns any monotone boolean function under the uniform distribution to any constant accuracy, in time polynomial in n and in the *decision tree size* of the target function.

Recent work on restricted access learning [WDYR12] of DNF is related to our work at the high level: both our work and [WDYR12] give the learner more power with the goal of being able to learn DNF using this extra power.

1.3 Our Results

We begin with a somewhat surprising negative result: that learning general DNF formulas under arbitrary distributions from boolean similarity queries is as hard as PAC-learning DNF formulas without them. This result uses the equivalence between group learning, weak learning, and strong learning.

Under the uniform distribution, we show that if we are allowed to ask numerical pairwise queries, we can efficiently learn *arbitrary* DNF formulas. If we are restricted to boolean-valued queries, we can properly learn any DNF formula for which each variable appears in $O(\log(n))$ terms, as well as any DNF formula with $O(\log(n))$ relevant variables. We can also learn any DNF having at most $2^{O(\sqrt{\log n})}$ terms. Specifically, we use Servedio's results [Ser04] in the context of learning the $k(x_i, \cdot)$ DNF, which is a monotone DNF after appropriate transformation of the feature space; if we do this for enough random x_i points, the $\max_i k(x_i, \cdot)$ function will agree with the target DNF with high probability.

For the more general case of learning under arbitrary distributions, we show that with numerical-valued queries we can properly learn any DNF formula having $O(\log(n))$ terms, or any DNF formula having $O(\log(n))$ relevant variables. In this case, with boolean-valued queries, we can easily learn disjoint DNF (a class that contains decision trees). We in addition show that we can properly learn any "parsimonious" DNF formula (a formula for which no term can be deleted without appreciably changing the function) as well as any 2-term DNF, a class known to be NP-Hard to properly learn from labeled data alone.

If we are allowed to ask "Do these k examples satisfy any term in common?" for arbitrary (poly-sized) k, we can even properly learn arbitrary DNF formulas under arbitrary distributions.

2. HARDNESS RESULTS

To illustrate why pairwise similarity queries do not trivialize the DNF learning problem, we begin with two hardness results.

THEOREM 2.1. *Learning DNF from random data under arbitrary distributions with boolean similarity queries is as hard as learning DNF from random data under arbitrary distributions with only the labels (no queries).*

PROOF. [Kea89] and [KLV94] proved that "group learning" is equivalent to "weak learning".

In group learning, at each round we are given $poly(n)$ examples that are either all iid from D^+ or all iid from D^- (i.e. all positive or all negative) and our goal is to identify which case it is. Subsequently, Schapire [Sch90] proved that weak-learning is equivalent to strong-learning. So, if DNF is hard to PAC-learn, then DNF is also hard to group-learn.

Now, consider the following reduction from group-learning DNF in the standard model to learning DNF in the boolean similarity queries model. In particular, given an algorithm $\mathcal{A}$ for learning from a polynomial number of examples in the boolean similarity queries model, we show how to use $\mathcal{A}$ to group-learn as follows:

Given a set S of $m = poly(n)$ examples x_1, x_2, ..., x_m (we will use $m = tn$ where t is the number of terms in the target), construct a new example by just concatenating them together. So overall we now have nm variables. We present this concatenated example to $\mathcal{A}$ with label equal to the label of S. If $\mathcal{A}$ makes a similarity query between two positive examples $[x_1, x_2, ..., x_m]$ and $[x'_1, x'_2, ..., x'_m]$, we simply output *yes* (i.e., that they do indeed share a term in common).

We now argue that with high probability, the labels and our responses to $\mathcal{A}$ are all fully consistent with some DNF formula of size mt. In particular, we claim they will be consistent with a target function that is just the OR of m copies of the original target function.

First of all, note that the OR of m copies of the original target function will produce the correct labels since by assumption either all $x_i \in S$ are positive or all $x_i \in S$ are negative. Next, we claim that whp, any two of these concatenated positive examples will share a term in common. Specifically, if the original DNF formula has t terms, then for two random positive examples from D^+ there is probability at least $1/t$ that they share a common term. So, the chance of failure for two concatenated examples is at most $(1 - 1/t)^m$. (Because the only way that two of these big concatenated examples $[x_1, x_2, ..., x_m]$ and $[x'_1, x'_2, ..., x'_m]$ can fail to share a term in common is if x_1 and x'_1 fail, x_2 and x'_2 fail, etc.). Setting $m = tn$, the probability of failure for any given query is at most $1/e^n$. Applying the union bound over all polynomially-many pairs of positive examples in $\mathcal{A}$'s sample yields that with high probability all our responses are consistent. Therefore, by assumption, $\mathcal{A}$ will produce a low-error hypothesis under the distribution over concatenated examples, which yields a low-error hypothesis for the group-learning problem. □

We can extend the above result to "approximate numerical" queries that give the correct answer up to a $1 \pm \tau$ factor, for some constant $\tau > 0$ (or even $\tau \geq 1/poly(n)$).

THEOREM 2.2. *Learning DNF from random data under arbitrary distributions with approximate-numerical-valued queries is as hard as learning DNF from random data under arbitrary distributions with only the labels (no queries). Specifically, if C is the number of terms x_i and x_j satisfy in common, the oracle returns a value in the range $[(1-\tau)C, (1+\tau)C]$.*

PROOF. Assume we have an algorithm $\mathcal{A}$ that learns to error $\epsilon/2$ given an oracle for approximate numerical queries.

Now we do the reduction from group learning as before, forming higher-dimensional examples by concatenating groups $x_1, \cdots, x_m$, all of the same class, but this time with $m = 2n(t^4)(1+\tau/2)^2/\tau^2$. Suppose, for now, that we know for the original DNF formula, the expected number of terms α that two random positive examples would have in common (we discharge this assumption later). In that case, when queried by $\mathcal{A}$ for the similarity between two positive examples x, x', we simply answer with the closest integer to αm. As before, we argue that with high probability, our answers are consistent with a DNF formula g consisting of just m shifted copies of the original DNF.

Note that for a random pair of the concatenated examples composed of positive sub-examples, the expected number of terms in common in g is $m\alpha$. Furthermore, the number of terms in common is a sum of m independent samples of the original random variable of mean α, each of which is bounded in the range $[0, t]$. So Hoeffding's inequality implies that with probability

$$1 - 2e^{-2m^2\alpha^2(\tau/2)^2/(m(t^2)(1+\tau/2)^2)} \geq 1 - 2e^{-n}$$

(since $\alpha \geq 1/t$), the number C of terms in common satisfies $|C - m\alpha| \leq m\alpha(\tau/2)/(1+\tau/2)$, which implies $(1-\tau/2)C \leq m\alpha \leq (1+\tau/2)C$.

Thus, for a $poly(n)$-sized sample of data points, with high probability, all of the pairs of positive concatenated examples have the nearest integer to $m\alpha$ within these factors of their true number of terms in common. It therefore suffices to respond to $\mathcal{A}$'s similarity queries with the nearest integer to $m\alpha$.

Now the only difficulty is that we do not know α. So we just try all positive integers i from 1 to mt and then use a validation set to select among the hypotheses produced. That is, we run $\mathcal{A}$ on the constructed data set and respond to all similarity queries with a single value i, getting back a classifier for these concatenated examples, and then repeat for each i. Then we take $O((1/\epsilon)\log(mt/\delta))$ additional higher-dimensional samples (with labels) and choose the classifier among these mt returned classifiers, having the smallest number of mistakes on that validation set. At least one of these mt values of i is the closest integer to $m\alpha$, so at least one of these mt classifiers is $\epsilon/2$-good, and our validation set will identify one whose error is at most ϵ. So we can use this classifier to identify whether a random m-sized group of examples is composed of all positives or all negatives, with error rate ϵ: i.e., we can do group learning.

If the algorithm A only has a "high probability" guarantee of success, we can repeat this several times with independent data sets, to boost the confidence that there will be a good classifier among those we choose from at the end, and slightly increase the size of the validation set to compensate for this larger number of classifiers. □

3. LEARNING DNF UNDER THE UNIFORM DISTRIBUTION

In this section, we investigate the problem of learning DNF under a uniform distribution on $\{0,1\}^n$. We begin with a result showing how to learn general DNF formulas from numerical similarity queries. For boolean similarity queries, we show how to learn several natural classes of DNF formulas, including $2^{O(\sqrt{\log n})}$-term DNF, DNF with $O(\log n)$ relevant variables (juntas), and DNF formulas in which each variable appears in at most $O(\log n)$ terms. First, however, we give an algorithm for learning a sum of monotone terms in the standard (no queries) PAC model under the uniform distribution.

3.1 A useful subroutine: Learning a sum of monotone terms

In this section we give an algorithm for learning a sum of monotone terms over the uniform distribution (without pairwise queries). That is, the target is a (not necessarily boolean) function $f(x) = T_1(x) + T_2(x) + \ldots + T_t(x)$ where the T_i are monotone conjunctions (outputting 1 if x satisfies T_i and 0 if x does not) and $t = poly(n)$. This will then be used in our algorithm for learning general DNF formulas over the uniform distribution from pairwise queries.

THEOREM 3.1. *We can efficiently learn a sum of t monotone terms over the uniform distribution, without pairwise queries, using time and samples $poly(t, n, 1/\epsilon)$.*

PROOF. Let $f(x) = T_1(x) + T_2(x) + \ldots + T_t(x)$ denote the target function. Our algorithm is based on the following facts:

1. The Fourier representation (using the parity basis) for a single term T (viewed as a $\{0,1\}$ function) has a particularly simple form. In particular, $\hat{T}_S = \mathbb{E}[T(x)\phi_S(x)] = 0$ if $S \not\subseteq T$ and $\hat{T}_S = 2^{-|T|}(-1)^{|S|+1}$ if $S \subseteq T$. See, e.g., [BFJ+94].

2. This implies that for any given set S, the associated Fourier coefficient of f is $\hat{f}_S = (-1)^{|S|+1}\sum_{i:S\subseteq T_i} 2^{-|T_i|}$. This further implies that if $S' \subseteq S$, then $|\hat{f}_{S'}| \geq |\hat{f}_S|$. Note that we are using here the fact that the T_i are monotone, so that all terms inside the summation have the same sign.

3. Since for each term T_i, its L_1 length in the Fourier representation is exactly 1, we have $L_1(f) \leq t$. This implies that for any given threshold θ there are at most t/θ coefficients of magnitude at least θ.

We can use the above facts to identify all Fourier coefficients of f of magnitude at least $\theta = \epsilon/(8t)$ in time polynomial in n, t, and $1/\epsilon$ as follows.

We begin by examining each parity function of size 1 and estimating its Fourier coefficient from data (up to accuracy $\theta/4$). We place all coefficients of magnitude at least $\theta/2$ into a list L^1.

For $j = 2, 3, \ldots$ we repeat the following: for each parity function ϕ_S in list L^{j-1} and each $x_i \notin S$, estimate the Fourier coefficient of $S_i = S \cup \{x_i\}$. If the estimated value is at least $\theta/2$ then add it to list L^j (if it is not in the list already). Note that by observation (2) above, we maintain by induction that list L^j contains all parity functions of size j whose coefficients have magnitude at least θ.

Lastly, note that by observation (3), each list L^j can have length at most $O(t/\theta)$, so the overall total time is polynomial in n, t, and $1/\theta$.

We now construct a function g consisting of the weighted sum of parities for all coefficients identified in the above procedure. Since g includes all coefficients of f of magnitude at least $\epsilon/(8L_1(f))$, this implies that we have

$$\langle f-g, f-g\rangle \leq \sum_{S:|\hat{f}_S|<\epsilon/(8L_1(f))} (\hat{f}_S)^2 \leq \epsilon/8$$

[Man94]. The above assumes we have measured the large Fourier coefficients precisely; adding in measurement error, by measuring coefficients to $poly(t/\epsilon)$ accuracy, we have $\langle f-g, f-g\rangle \leq \epsilon/4$. In particular, this implies that $\mathbb{E}[(f(x)-g(x))^2] \leq \epsilon/4$. Finally, we output the function $h(x) = [g(x)]$ where "[.]" is rounding to the nearest integer. Each mistake of h on some example x implies that $(f(x)-g(x))^2 \geq 1/4$, so h has error at most ϵ with respect to f. □

Feldman, Kothari, and Vondrak (personal communication) have recently independently proven a result related to this theorem. Note that the above result holds just as well for *unate* sums of terms (a sum of terms in which no variable appears both positively and negatively). In particular, the only place where monotonicity was used in the proof was in observation (2), which applies just as well to unate sums, since they still have the property that $|\hat{f}_S| = \sum_{i:S\subseteq T_i} 2^{-|T_i|}$, implying that if $S' \subseteq S$, then $|\hat{f}_{S'}| \geq |\hat{f}_S|$. Therefore, we have:

THEOREM 3.2. *We can efficiently learn a unate sum of t terms over the uniform distribution, using time and samples $poly(t, n, 1/\epsilon)$.*

In the case that all terms T_i have size at most $s = O(\log(t/\epsilon))$, we can convert the above procedure into a proper learning algorithm by setting the threshold θ to $O(1/2^s)$ and outputting terms for the maximal sets S found having large Fourier coefficients.

3.2 Learning DNF over the uniform distribution from numerical pairwise queries

We now use Theorem 3.2 to learn general DNF formulas over the uniform distribution from numerical pairwise queries.

THEOREM 3.3. *Under the uniform distribution, with numerical pairwise queries, we can learn any $poly(n)$-term DNF.*

PROOF. Let t be the number of terms in the target. Sample $m = O((t/\epsilon)\log(t/\epsilon\delta))$ "landmark" points $x_1, x_2, \ldots, x_m$. If x_i is a positive example, define $F_i(\cdot) = K(x_i, \cdot)$, where K is the pairwise numerical query function. That is, $F_i(x)$ is the sum-of-terms function for the target DNF formula in which all terms not satisfied by x_i have been removed. This sum-of-terms is unate, since every variable that appears must be in agreement with x_i. Therefore, we can use Theorem 3.2 to learn a hypothesis $h_i(x)$ with error at most $\epsilon/(2m)$ with respect to F_i.

We now combine all hypotheses h_i produced to a single boolean function h defined as: $h(x) = 0$ if $h_i(x) = 0$ for all i, else $h(x) = 1$. By the union bound, h has error at most $\epsilon/2$ with respect to the function f' consisting of all terms of the target satisfied by at least one landmark x_i. Finally, the number of landmarks is sufficiently large that f' has error at most $\epsilon/2$ with respect to the target, so by the triangle inequality, h has error at most ϵ with respect to the target. □

3.3 Learning DNF over the uniform distribution from boolean pairwise queries

DEFINITION 3.4. *Fix a constant $c \in (0, \infty)$. We say a term T in the target DNF is "relatively distinct" if it contains a variable v*

which occurs in at most $c\log(n)$ other terms. We say v is a witness to T being relatively distinct.

DEFINITION 3.5. *For a term T in the target DNF, and a variable v in T, we say v is "sometimes nonredundant" for T if, given a random example that satisfies T, there is at least an ϵ probability that every term in the target DNF that the example satisfies also contains v.*

THEOREM 3.6. *Suppose no term in the target DNF is logically entailed by any other term in the target DNF, every term T is relatively distinct, and that some variable v that is a witness to T being relatively distinct is sometimes nonredundant for T. Then we can properly learn any monotone DNF of this type under a uniform distribution on $\{0,1\}^n$ with boolean pairwise queries.*

PROOF. By Lemma 4.1, it suffices to show that every term having at least $\epsilon/(2t)$ probability of being satisfied will, with high probability, have some example satisfying only that term, given a polynomial-sized data set.

Consider a given term T in the target DNF, and choose the v that witnesses relative distinctness which is sometimes nonredundant. Note that every other term in the target DNF contains some variable not present in T, and in particular this is true for the (at most) $c\log(n)$ terms containing v. So under the conditional distribution given that T is satisfied and that v is nonredundant, with probability at least $2^{-c\log(n)} = n^{-c}$, none of these other terms containing v are satisfied, so that T is the only term satisfied. Thus, since T has probability at least $\epsilon/(2t)$ of being satisfied, and v has probability at least ϵ of being nonredundant given that T is satisfied, we have that with probability at least $(\epsilon^2/t)n^{-c}$, a random example satisfies T and no other terms in the target DNF.

Since this is the case for all terms in the target, a sample of size $O((t/\epsilon^2)n^c\log(t/\delta))$ guarantees every term has some example satisfying only that term, with probability at least $1-\delta$. □

We can also consider the class of DNF formulas having only a small number of relevant variables. In this context, it is interesting to observe that if the i^{th} variable is irrelevant, then $P(k(x,y) = 1 \text{ and } x_i \neq y_i) = P(k(x,y) = 1 \text{ and } x_i = y_i)$, where x and y are independent uniformly-distributed samples, and $k(x,y) = 1$ iff x and y are positive examples that satisfy at least one term in common. However, as the following lemma shows, this is not true for relevant variables.

LEMMA 3.7. *For x and y independent uniformly-distributed examples, if the target function has r relevant variables, and the i^{th} variable is relevant to the target function, then $P(k(x,y) = 1$ and $x_i = y_i) - P(k(x,y) = 1$ and $x_i \neq y_i) \geq (1/4)^r$.*

PROOF. For each pair (x,y) with $x_i \neq y_i$, there is a unique corresponding pair (x',y) with $x'_j = x_j$ for $j \neq i$, and $x'_i = y_i$. Let M_i be the number of x,y pairs with $x_i \neq y_i$ and $k(x,y) = 1$. Then note that for every x,y pair with $x_i \neq y_i$ and $k(x,y) = 1$, we also have $k(x',y) = 1$, since whatever term x and y satisfy in common cannot contain variable i anyway, so flipping that feature in x does not change whether x and y share a term or not. In particular, this implies the number of x,y pairs with $x_i = y_i$ and $k(x,y) = 1$ is at least M_i. However, we can also argue it is strictly larger, as follows. By definition of "relevant", each of the 2^r settings of the relevant variables corresponds to an equivalence class of feature vectors, all of which have the same label, and if that label is positive, then all of which have the same profile. Since variable i is relevant, at least one of the 2^r settings of the relevant variables yields an equivalence class of positive examples whose profile contains only terms with variable i in them (these are positive examples such that flipping variable i makes them negative). The probability that both x and y (chosen at random) are in this equivalence class is $(1/4)^r$. Note that for the (x,y) pairs of this type, we have $k(x,y) = 1$; however, if we flip feature x_i, then x would become negative, and hence $k(x,y)$ would no longer be 1; this means this (x,y) pair is not included among those M_i pairs constructed above by flipping x_i starting from some (x,y) with $x_i \neq y_i$ and $k(x,y) = 1$. So $P(k(x,y) = 1 \text{ and } x_i = y_i) - P(k(x,y) = 1 \text{ and } x_i \neq y_i) \geq (M_i/4^n + (1/4)^r) - M_i/4^n = (1/4)^r$. □

THEOREM 3.8. *Under the uniform distribution, with boolean pairwise queries, we can properly learn any DNF having $O(\log(n))$ relevant variables.*

PROOF. We can use the property in Lemma 3.7 to design an algorithm as follows. For each i, sample $\Omega(8^r\log(n/\delta))$ random pairs (x,y), and evaluate $k(x,y)$ for each pair. Then calculate the difference of empirical probabilities (fraction of pairs (x,y) for which $k(x,y) = 1$ and $x_i = y_i$ minus fraction of pairs (x,y) for which $k(x,y) = 1$ and $x_i \neq y_i$). If this difference is $> (1/2)(1/4)^r$, decide variable i is relevant, and otherwise decide variable i is irrelevant. By Hoeffding and union bounds, with probability $1-\delta/2$, this will find exactly the r relevant variables. Now enumerate all $2^r = \text{poly}(n)$ possible conjunctions that can be formed from using all of these r relevant variables. Considering this as a 2^r-dimensional feature space, take $\Omega((2^r/\epsilon)log(1/\delta))$ random labeled data points and learn a disjunction over this 2^r-dimensional feature space; since the VC dimension of this set of disjunctions is 2^r, the usual PAC analysis implies this will learn an ϵ-good disjunction with probability $1-\delta/2$. A union bound implies both stages (finding variables and learning the disjunction) will succeed with probability at least $1-\delta$. □

An alternative approach to the second stage in the proof would be to take $\Omega(2^r\log(2^r/\delta))$ random samples, so that with probability at least $1-\delta/2$, we have at least one data point satisfying each of the 2^r possible conjunctions on the relevant variables; then for each of the conjunctions, we check the label of the example that satisfies it, and if that label is positive, we include that conjunction as a term in our DNF, and otherwise we do not include it. This has the property that, altogether, with probability $1-\delta$, we construct a DNF that has error rate *zero*.

Another family of DNF studied in the literature are those with a sublinear number of terms. Specifically, [Ser04] proved that the class of $2^{O(\sqrt{\log n})}$-term *monotone* DNF are learnable under the uniform distribution from labeled data alone. As the following theorem states, we can extend this result to include general $2^{O(\sqrt{logn})}$-term DNF (including non-monotone) given access to our boolean pairwise queries.

THEOREM 3.9. *Under the uniform distribution, with boolean pairwise queries, we can learn any $2^{O(\sqrt{\log n})}$-term DNF (supposing ϵ to be a constant).*

First, we review some known results from [Ser04]. For any function $g : \{0,1\}^n \to \{-1,+1\}$, define the $g_{i,1}$ and $g_{i,0}$ functions by the property that any x with $x_i = 1$ has $g_{i,1}(x) = g(x)$, and $g_{i,0}(x) = g(y)$, where $y_j = x_j$ for $j \neq i$ and $y_i = 0$. Then define the influence function $I_i(g) = P(g_{i,0}(x) \neq g_{i,1}(x))$. [Ser04] developed a procedure, FindVariable, which uses a $\text{poly}(n, 1/\gamma, \log(1/\eta))$ number of random labeled samples, labeled according to any monotone DNF g having at most t terms, and with probability $1-\eta$, returns a set S of variables (indices in $\{1,\ldots,n\}$) such that

every $i \notin S$ has $I_i(g) \leq \gamma$ and every $i \in S$ has $I_i(g) \geq \gamma/2$ and the i^{th} variable is contained in some term in g with at most $\log \frac{32tn}{\gamma}$ variables in it.

Furthermore, [Ser04] showed that, for any t-term DNF f, if we are provided with a set $S_f \subseteq \{1, \ldots, n\}$ such that every $i \notin S_f$ has $I_i(f) \leq \epsilon/4n$, then we can learn f in time polynomial in n, $|S_f|^{O(\log \frac{t}{\epsilon} \log \frac{1}{\epsilon})}$, and $\log(1/\delta)$. In particular, for $|S_f| = O(t \log \frac{tn}{\epsilon})$ and $t = 2^{O(\sqrt{\log n})}$, this is polynomial in n (though not necessarily in ϵ). Given the set S_f, the learning procedure simply estimates the Fourier coefficients for small subsets of S_f.

PROOF OF THEOREM 3.9. To prove Theorem 3.9, we consider the following procedure. First draw m random labeled examples $x^{(1)}, \ldots, x^{(m)}$. Then, for each $j \leq m$, define $k_j(\cdot) = k(x^{(j)}, \cdot)$. Now note that, if we define $\varphi_j(y) = (\varphi_{j1}(y), \ldots, \varphi_{jn}(y))$ by $\varphi_{ji}(y) = 2I[y_i = x_i^{(j)}] - 1$, then we can represent $k_j(\cdot) = (k'_j(\varphi_j(\cdot)) + 1)/2$, where k'_j is a monotone DNF (mapping into $\{-1, +1\}$); specifically, the terms in k'_j correspond to the terms in the target satisfied by $x^{(j)}$, except none of the literals are negated. We then run FindVariable for each of these k'_j, with $\gamma = \epsilon/m$ and $\eta = \delta/2m$. Let S_f denote the union (over $j \leq m$) of the returned sets of variables. It remains only to show this S_f satisfies the requirements for the procedure of [Ser04], including the size requirement.

Taking $m = \Omega(\frac{ct}{\epsilon} \log \frac{t}{\delta})$, with probability at least $1 - \delta/4$, every term in the target having probability at least $\epsilon/2ct$ will have at least one of the m examples satisfying it. Suppose this event happens. In particular, this means $\text{error}(\max_j k_j) < \epsilon/2c$. Note that

$$\begin{aligned} I_i(f) &= P(f_{i,0}(x) \neq f_{i,1}(x)) \\ &\leq 2P(\max_j k_j(x) \neq f(x)) + \\ &\quad P((\max_j k_j)_{i,0}(x) \neq (\max_j k_j)_{i,1}(x)) \\ &< \epsilon/c + \sum_j P((k'_j)_{i,0}(x) \neq (k'_j)_{i,1}(x)) \\ &= \epsilon/c + \sum_j I_j(k'_j). \end{aligned}$$

Thus, by a union bound, with probability $1 - \delta/2$, any variable $i \notin S_f$ has $I_i(f) < \epsilon/c + m\gamma$, and any variable $i \in S_f$ appears in a term in some k'_j of size at most $\log \frac{32tn}{\gamma}$, and therefore also appears in a corresponding term of this size in f. Suppose this happens. Letting $c = 8n$ and $\gamma = \epsilon/8nm$, we have that any $i \notin S_f$ has $I_i(f) < \epsilon/4n$, while any $i \in S_f$ appears in a term of size at most $\log \frac{256tn^2m}{\epsilon} = O(\log \frac{tn \log(1/\delta)}{\epsilon})$. In particular, this implies $|S_f| = O(t \log \frac{tn \log(1/\delta)}{\epsilon})$, and S_f satisfies the requirements of the method of [Ser04].

Thus, running the procedure from [Ser04] with confidence parameter $\delta/4$, a union bound implies the total probability of successfully producing an ϵ-good classifier is at least $1 - \delta$. The above process of constructing S_f is clearly polynomial-time. Then, if $t = 2^{O(\sqrt{\log n})}$, the procedure of [Ser04] runs in time polynomial in n, $\log(1/\delta)$, and $|S_f|^{O(\log(t/\epsilon) \log(1/\epsilon))}$, which is polynomial in n and $\log(1/\delta)$ (though not necessarily in ϵ). □

One interesting observation is that the above is a general reduction. In particular, given any method for learning monotone DNF under the uniform distribution in time poly$(n, 1/\epsilon, \log(1/\delta))$ from random labeled data, we have a method for learning general DNF under the uniform distribution in time poly$(n, 1/\epsilon, \log(1/\delta))$ using boolean-valued queries: simply sample $O(t/\epsilon)$ random examples x_i, and for each learn the monotone (with respect to x_i) DNF $k(x_i, \cdot)$ to error ϵ^2/t, and return the resulting estimate of the hypothesis $\max_i k(x_i, \cdot)$. This points out that one advantage of learning with boolean-valued queries is that it allows us to convert results that hold for monotone DNF into results that hold for general DNF, due to the monotone landmark DNFs that compose to form the target function.

4. LEARNING DNF UNDER GENERAL DISTRIBUTIONS : POSITIVE RESULTS

Our methods are described below, split based on the type of query they use (boolean vs numerical). We will see that the method for boolean queries (called the "neighborhood method" below) is effective for any *parsimonious* DNF: a target DNF for which, on the given data set, removing any term from the target DNF would change the label of some example in the data. Note that ordinarily one can assume without loss of generality that a DNF target is parsimonious, but that is not the case with pairwise queries. The method based on numerical-valued queries (called the "common profiles approach" below) will turn out to be effective under conditions on the target DNF that imply *common profiles*; that is, if we partition the data into equivalence classes based on equality of the set of all terms they satisfy in the target DNF, then each equivalence class has $poly(n)$ examples in it; for instance, this is the case as long as we are guaranteed there are at most a $poly(n)$ number of such equivalence classes (independent of the number of data points).

4.1 Methods

4.1.1 The Neighborhood Method

We refer to the following simple procedure as the "neighborhood method". Take $m = \text{poly}(n, 1/\epsilon, \log(1/\delta))$ samples. First, among the positive examples, query all pairs (with the boolean-valued query) to construct a graph, in which examples are adjacent if they satisfy a term in common. For each positive example, construct a minimal conjunction consistent with that example and all of its neighbors (i.e., the consistent conjunction having largest number of literals in it). Next, discard any of these conjunctions that make mistakes on any negative examples. Then sequentially remove any conjunction c_1 such that some other remaining conjunction c_2 subsumes it (contains a subset of the variables). Form a DNF from the remaining conjunctions. Produce this resultant DNF as the output hypothesis.

LEMMA 4.1. *Suppose the target DNF has $t = \text{poly}(n)$ terms. For an appropriate (t-dependent) polynomial sample size m, the neighborhood method will, with probability at least $1 - \delta$, produce an ϵ-accurate DNF if, for each term T_i in the target DNF having a probability of satisfaction at least $\epsilon/2t$, there is at least a $p = 1/\text{poly}(n, 1/\epsilon)$ probability that a random example satisfies term T_i and no other term (we call such an example a "nice seed" for T_i).*

PROOF. Under these conditions we have that

$$m = O((1/p)\log(t/\delta) + (t/\epsilon)\log(1/\epsilon\delta))$$

samples suffice to guarantee each T_i with probability of satisfaction at least $\epsilon/2t$ has at least one nice seed, with probability at least $1 - \delta/2$.

In the second phase, we remove any conjunction inconsistent with the negative examples. The conjunctions guarnateed by the above argument survive this pruning due to their minimality, and the fact that they are learned from a set of examples that actually are consistent with some term in the target DNF (due to the nice

seed). The final pruning step, which removes any redundancies in the set of conjunctions, leaves at most t conjunctions.

The terms that do not have nice seeds compose at most $\epsilon/2$ total probability mass, and m is large enough so that with probability at least $1 - \delta/4$, at most a $3\epsilon/4$-fraction of the data satisfy these terms. Thus, since the result of the neighborhood method is a DNF formula with at most t terms, which correctly labels a $1 - 3\epsilon/4$ fraction of the m examples, the standard PAC bounds imply that with probability at least $1 - \delta/4$, the resulting DNF has error rate at most ϵ. A union bound over the above events implies this holds with probability at least $1 - \delta$. □

4.1.2 The Common Profile Approach

In the case of numerical queries, we have some additional flexibility in designing a method. In this context, we refer to the following procedure as the "common profiles approach".

Consider a sample of $m = \text{poly}(n, 1/\epsilon, \log(1/\delta))$ random labeled examples, and for each pair of positive examples x, y, we request the number $K(x, y)$ of terms they satisfy in common; we additionally request $K(x, x)$ for each positive example x. For each positive example x, we identify the set S of examples y such that the numerical value of $K(x, y)$ is equal $K(x, x)$. So these points satisfy at least all the terms x satisfies. For each such set S, we learn a minimal conjunction consistent with these examples. Then for each of these conjunctions, if it is a specialization of some other one of the conjunctions, we discard it. Then we form our hypothesis DNF with the remaining conjunctions as the terms.

For any example x, relative to a particular target DNF, we refer to the "profile" of x as the set of terms T_i in the target DNF satisfied by x.

LEMMA 4.2. *If the target DNF has at most $p = \text{poly}(n)$ possible profiles, then the common profile approach, with an appropriate (p-dependent) sample size m, will with probability at least $1 - \delta$, produce a DNF having error rate at most ϵ.*

PROOF. Note that this procedure produces a DNF that correctly labels the entire data set, since $K(x, y) = K(x, x)$ implies x and y have the same profiles, so that in particular the set S has some term in common to all the examples. If there are only a $\text{poly}(n)$ number of possible profiles, then the above will only produce at most as many distinct terms in its hypothesis DNF, so that a sufficiently large $\text{poly}(n)$-sized data set will be sufficient to guarantee good generalization error. Specifically, $m = O((pn/\epsilon)\log(1/\epsilon\delta))$ examples are enough to guarantee with probability at least $1 - \delta$, any DNF consistent with the data having at most p terms will have error rate at most ϵ, so this is sufficient for the common profile approach. □

4.2 Positive Results

THEOREM 4.3. *If it happens that the target DNF is parsimonious (no redundant terms) for some random $\Omega((tn/\epsilon)\log(1/\epsilon) + (1/\epsilon)\log(1/\delta))$-sized data set (for any distribution), then we can efficiently produce a DNF consistent with it having at most t terms using boolean-valued queries.*

PROOF. Parsimonious, in this case, means that we cannot remove any terms without changing some labels. But this means that every term has some example that satisfies only that term (i.e., a nice seed). So as described in the proof of Lemma 4.1 above, the "neighborhood method," produces a DNF with terms for the neighborhoods of each of these nice seeds, which in the parsimonious case, covers all of the positive examples. □

COROLLARY 4.4. *We can properly learn any 2-term DNF with boolean-valued queries.*

PROOF. Take $O((n/\epsilon)\log(1/\epsilon) + (1/\epsilon)\log(1/\delta))$ random labeled examples and make the boolean query for all pairs of positive examples. First, find a minimal conjunction consistent with all of the positive examples; if this conjunction does not misclassify any negative examples, return it. By classic PAC bounds, a conjunction consistent with this many random labeled examples will, with probabiliy at least $1 - \delta$, have error rate at most ϵ. Otherwise, if this conjunction misclassifies some negatives, then we are assured the target DNF is parsimonious for this data set, and thus Theorem 4.3 guarantees we can efficiently identify a 2-term DNF consistent with it using the boolean-valued queries. Again, the classic PAC bounds imply the sample size is large enough to, with probability at least $1 - \delta$, guarantee that any consistent 2-term DNF has error rate at most ϵ. □

Corollary 4.4 gives a concrete result where using this type of query overturns a known hardness result for supervised learning.

COROLLARY 4.5. *With numerical-valued queries, we can properly learn any DNF having $O(\log(n))$ relevant variables, under arbitrary distributions.*

PROOF. These targets have $\text{poly}(n)$ possible profiles, so the common profiles approach will be successful. □

COROLLARY 4.6. *If the target DNF has only $O(\log(n))$ terms, then we can efficiently properly learn from random data under any distribution using numerical-valued queries.*

PROOF. There are only $\text{poly}(n)$ number of possible profiles, so the "common profiles" approach will work. □

The above result is interesting particularly because proper learning (even for 2-term DNF) is known to be hard from labeled data alone.

COROLLARY 4.7. *If the target DNF has $t = \text{poly}(n)$ terms, and is such that any example can satisfy at most $O(1)$ terms, then we can efficiently properly learn from random data using numerical-valued queries.*

PROOF. There are at most $\text{poly}(t) = \text{poly}(n)$ possible profiles, so the "common profiles" approach will work. □

COROLLARY 4.8. *If the DNF is such that any example can satisfy at most 1 term (a so-called "disjoint" DNF), then we can efficiently properly learn from random data using boolean-valued queries.*

PROOF. A numerical query whose value can be at most 1 is just a boolean query anyway. □

In particular, Decision Trees can be thought of as a DNF where each example satisfies at most 1 term.

5. MORE POWERFUL QUERIES

If we can ask about k-tuples of examples (do they all jointly satisfy a term in common?), we have the following result:

THEOREM 5.1. *If we can use query sets of arbitrary sizes (instead of just 2 points), then under any distribution we can efficiently properly learn DNF using boolean-valued queries from random data.*

PROOF. We take any set of examples and ask the oracle the number of terms all examples in the set have in common. Let S be the query set. The idea is to greedily add the examples to S while keeping some terms in common.

Algorithm:

0. Input : dataset D
1. Initialize S to be an empty set
2. **Do**{
3. **Do**{
4. $r_{\max} \leftarrow 0$
5. For each example x in the dataset D
6. add x to the set S
7. query the combined set S, and let $r = Oracle(S)$, $r_{\max} \leftarrow \max\{r_{\max}, r\}$
8. If $r = 0$, remove x from S, and otherwise leave it in S and remove x from D
9. } **Until**($r_{max} = 0$)
10. Learn a "most-specific" conjunction from S and add that term to the hypothesis DNF
11. Reset S to empty set
12. }**Until** ($|D| = 0$)

Each time we add a term to the DNF, the examples in S satisfy some term in the target DNF, because we only add each example if by adding it S still has at least one term in common. So the "most-specific" conjunction consistent with S (i.e., the one with most literals in it, still labeling all of S positive) will not misclassify any negative point as positive. Since whenever we add a new term, there were no additional examples in D that could have satisfied a term in common with the examples in S, after adding the term we have removed from D all examples that satisfy the term S has in common. Therefore, the number of terms in our learnt DNF is at most the number of terms T in the true DNF. If the total number of examples is $\gg nT$ (and say T is $poly(n)$), it will get us a DNF that has at most T terms and correctly labels a $poly(n) \gg nT$ sized dataset. Since the training dataset size is much larger than the size of the classifier, by the Occam bound, the learnt DNF will have small generalization error. □

6. OPEN QUESTIONS

- Is it possible to efficiently learn an arbitrary DNF from random data under the uniform distribution with boolean-valued queries?
- Is it possible to efficiently learn an arbitrary DNF from random data under arbitrary distributions with numerical-valued queries? To establish this, in light of the general reduction used to prove Theorem 3.9, it would suffice to show that the class of sums of monotone terms are efficiently learnable from random data under arbitrary distributions. Or, alternatively, can Theorem 2.2 be extended to apply to *exact* numerical queries?

Acknowledgments

Liu Yang would like to extend her sincere thanks to Ryan O'Donnell for his feedback and Steven Rudich for his counseling on Fourier techniques. This work was supported in part by NSF grant IIS-1065251 and a Google Core AI grant.

7. REFERENCES

[Ang88] D. Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–Ð342, 1988.

[BFJ+94] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly Learning DNF and Characterizing Statistical Query Learning using Fourier Analysis. In *Proceedings of the 26th annual ACM symposium on Theory of computing (STOC)*, pages 253–262, 1994.

[Jac94] J. Jackson. An Efficient Membership-Query Algorithm for Learning DNF with Respect to the Uniform Distribution. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 42–53, 1994.

[Kea89] M. Kearns. *The Computational Complexity of Machine Learning*. PhD thesis, Department of Computer Science, Harvard University, 1989.

[KLV94] M. Kearns, M. Li, and L. Valiant. Learning Boolean Formulas. *J. ACM*, 41:1298–1328, 1994.

[KS01] A. Klivans and R. Servedio. Learning DNF in Time $2^{\tilde{o}(n^{1/3})}$. In *Proceedings of the 33rd Symposium on Theory of Computing (STOC)*, pages 258–265, 2001.

[Man94] Y. Mansour. Learning Boolean Functions via the Fourier Transform. pages 391Ð–424, 1994.

[MOS04] E. Mossel, R. O'Donnell, and R. Servedio. Learning Juntas AKA Learning Functions of k Relevant Variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004.

[OS07] R. O'Donnell and R. Servedio. Learning Monotone Decision Trees in Polynomial Time. *SIAM Journal on Computing*, 37:827–844, 2007.

[PV88] L. Pitt and L. Valiant. Computational Limitations on Learning from Examples. *J. ACM*, 35:965–984, October 1988.

[Sch90] R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5:197–227, July 1990.

[Ser04] R. A. Servedio. On Learning Monotone DNF under Product Distributions. *Information and Computation*, 193:57–74, 2004.

[Val84] L. Valiant. A Theory of the Learnable. *Communications of the ACM,*, 27:1134Ð–1142, 1984.

[Val12] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas with noise. In *FOCS*, 2012.

[Ver90] K. Verbeurgt. Learning DNF under the Uniform Distribution in Quasi-Polynomial Time. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory (COLT)*, pages 314–326, 1990.

[WDYR12] A. Wigderson, Z. Dvir, A. Yehudayoff, and A. Rao. Restriction access. In *Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS)*, pages 19–33, 2012.

APPENDIX

A. QUERYING WITH POINTS OF OUR OWN CONSTRUCTION

THEOREM A.1. *If we can construct our own feature vectors in addition to getting random data, then under any distribution we can efficiently properly learn DNF using boolean-valued queries.*

PROOF. Suppose we can adaptively construct our own examples. Suppose the target DNF has $T = poly(n)$ terms. Oracle(x,

x') gives the number of terms that x and x' have in common. For any x, let x_{-i} be x but with the ith bit flipped. Let $\bar{x}$ be the negative of x.

Below is an algorithm. **Move(**x**,** x'**)** moves x' away from x by one bit, while trying to maintain at least one common term. **LearnTerm(**x**)** returns a term in the target function.

0. **Move**(x, x')
1. $x'' \leftarrow \bar{x}$
2. **For** $i = 1, 2, ..., n$ s.t. $x_i = x'_i$
3. **If** (Oracle(x, x'') $\leq$ Oracle(x, x'_{-i}))
4. $x'' \leftarrow x'_{-i}$
5. Return x''

0. **LearnTerm**(x)
1. Replicate x to get x'
2. **While** (Oracle(x, Move(x, x')) ! $= \emptyset$)
3. $x' \leftarrow$ Move(x, x')
4. Let $I \leftarrow \{i : Oracle(x, x'_{-i}) = \emptyset\}$
5. Return x_I (i.e. a conjunction with the literals indexed by I, either positive or negative so that x satisfies it)

0. **LearnDNF**
1. Initialize all-negative DNF $\hat{h}$
2. Take $M = poly(n) \gg nT$ random examples S
3. **For** each $x \in S$
4. **If** Oracle(x,x) > 0 (positive example) and $\hat{h}(x)$ = negative
5. Add term LearnTerm(x) to $\hat{h}$
6. Return $\hat{h}$ (a DNF with at most T terms, consistent with all M examples)

When we reach x' such that we can't flip any more bits (not already flipped) without making it so they don't satisfy any terms in common anymore, then the bits these two have in common must form a term in the target DNF, so LearnTerm(x) should still find a term in the target DNF. $\square$