Parameterized Learning Complexity

Rodney G. Downey

Mathematics Department
Victoria University
Wellington, New Zealand
downey@math.vuw.ac.nz

Patricia A. Evans and Michael R. Fellows

Computer Science Department
University of Victoria
Victoria, B.C. V8W 3P6, Canada
pevans (mfellows) @csr.uvic.ca

Abstract

We describe three applications in computational learning theory of techniques and ideas recently introduced in the study of parameterized computational complexity.
(1) Using parameterized problem reducibilities, we show that P-sized DNF (CNF) formulas can be exactly learned in time polynomial in the number of variables by extended equivalence queries if and only if the dominating sets of a graph can be learned in polynomial time by extended equivalence queries. (That is, learning by an arbitary hypothesis class. See Angluin [6].) Since learning dominating sets is a special case of learning monotone CNF formulas, this extends to the exact learning model a result of Kearns, li, Pitt and Valiant in the PAC prediction model [15]. We show that P-sized DNF (CNF) formulas can be learned exactly in polynomial time by extended equivalence and membership queries if and only there is an algorithm running in time polynomial in n and k to learn the k element dominating sets of an n vertex graph. We also prove related results concerning the problem of learning the truth assignments of weight k for DNF (CNF) formulas (that is, assignments that set exactly k variables to *true* and the rest to *false*).
(2) We describe a number of learning algorithms for both parameterized and unparameterized graph-theoretic learning problems, such as learning the independent sets, vertex covers or dominating sets of a graph.
(3) We show that computing the Vapnik-Chervonenkis dimension of a family of sets is complete for the parameterized complexity class $W[1]$.

1. Introduction

Whether DNF formulas can be learned in polynomial time has remained for many years one of the outstanding open questions in the field of learning theory. There are four results on this problem which form the context of our first theorem.
(1) DNF formulas are PAC-learnable in polynomial time if and only if monotone DNF formulas are PAC-learnable in polynomial time [15].
(2) Modulo a plausible cryptographic assumption, in the PAC model, DNF formulas can be learned in polynomial time by extended equivalence queries if and only if they can be learned in polynomial time by extended equivalence and membership queries [2].
(3) Monotone DNF formulas can be exactly learned by equivalence queries and membership queries in polynomial time [4], [18].
(4) Monotone DNF formulas are not learnable in polynomial time by equivalence queries if the hypotheses are required to be represented as DNF formulas [5].

Theorem 1. *Arbitrary DNF formulas can be learned in polynomial time if and only if monotone DNF formulas can be learned in polynomial time in the model of exact learning by means of extended equivalence queries (without membership queries).*

In fact, we show that the problem of learning DNF or CNF formulas reduces to the problem of learning the dominating sets of a graph in the model of exact learning by extended equivalence queries. (In this learning problem, the concept being taught is: the sets of vertices that are dominating sets in a particular graph. Note that a graph on n vertices may have as many as $2^n - 1$ distinct dominating sets.)

The significance of Theorem 1 depends upon how one feels about the learnability of DNF formulas. If DNF formulas cannot be learned in P-time, then presumably this will be easier to show in the exact model and the reduction to the monotone case may be useful in this direction (see (4) above). In view of (3), we would then have an indication of the importance of membership queries in the exact learning model, in contrast with the results of (2) for the PAC model. In any case, our theorem directly improves on (1) and contributes to our understanding of the relative power of the various models of learning.

Theorem 2. *Arbitrary DNF (CNF) formulas can be exactly learned in polynomial time by extended equivalence and membership queries if and only if there is an algorithm for learning the k-element dominating sets of a graph in time polynomial in k and the number of vertices of the graph.*

We also consider the problem of learning the truth assignments of *weight k* to DNF (CNF) formulas. A truth assignment has *weight k* if it assigns exactly k variables the value *true*. By analogy with the study of

ACM COLT '93 /7/93/CA, USA

parameterized computational complexity introduced in [1], [8] [9], [10], [11], [12], [13], the interesting question is whether the truth assignments of weight k can be learned in time $f(k)n^{\alpha}$, where f is an arbitrary function and α is a constant independent of k. We term this *fixed parameter learnability*; see Definition 6.

Theorem 3. *The weight k truth assignments to arbitrary DNF (CNF) formulas are fixed parameter learnable if and only if the weight k truth assignments to monotone DNF (CNF) formulas are fixed parameter learnable in the model of exact learning by means of extended equivalence queries.*

Theorem 4. *The weight k truth assignments to arbitrary DNF (CNF) formulas are fixed parameter learnable by extended equivalence and membership queries if and only if the k-element dominating sets of a graph are fixed parameter learnable in this model.*

Because the study of parameterized computational complexity employs a finely resolved measure of the complexity of checking solutions, it seems that it may provide an interesting window on the boundary between what can and what cannot be learned in polynomial time. With this possibility in mind and motivated by the connections shown in Theorems 2 and 4, we report in §4 on some results concerning the learning complexity of vertex sets in graphs.

In §5 we directly apply the theory of parameterized computational complexity to the problem of computing the VC dimension of a family of sets.

Theorem 5. *Determining whether the VC dimension of a family of sets is at least k is complete for the parameterized complexity class* $W[1]$.

A concrete interpretation of Theorem 5 is that we can determine whether the VC dimension of a family $\mathcal{F}$ of sets is at least k in time $f(k)|\mathcal{F}|^{\alpha}$, where f is an arbitrary function and α is independent of k, *if and only if* we can obtain an analogous result for the problem of determining whether a graph has a k-clique.

2. Preliminaries and Overview

In discussing the complexity of parameterized problems, we make the convention that in any use of asymptotic ("big O") notation, any hidden constants are independent of the parameter. In the interests of clarity, we will generally write completely explicit expressions for bounds on complexity (as in the discussion of Theorem 5 in the previous section).

Our focus is on the model of *exact learning by equivalence and membership queries* introduced by Angluin [3], [6], which we briefly review for completeness and to fix notation and terminology.

The Learning Model

Learning is modeled as an interaction between two players, the *Teacher* and the *Learner*. The object to be taught is a finite language $c \subseteq \Sigma^*$, where $\Sigma = \{0,1\}$, such that each word $x \in c$ has length n. Such a finite language is termed a *concept*. We refer to n as the *size* of the concept.

A learning problem is described by specifying a family $\mathcal{F}$ of representations r of concepts. We assume that $|r|$ is bounded by some polynomial in the size of the concept $c(r)$ represented by r. Thus, for example, when we discuss the learnability of CNF or DNF expressions we assume that there is a fixed polynomial bound on the number of literals in an expression e as a function of the number of variables; the concept represented is the set of 0-1 vectors corresponding to truth assignments to e. We may view the requirement that the representations have size bounded by some polynomial in the size of the represented concepts to be analogous to the requirement that solutions can be checked in polynomial time in the definition of NP

In the exact learning model, with respect to a given learning problem $\mathcal{F}$, we define three different kinds of queries which the Learner may make to the Teacher.
(1) *Membership Query:* "Is $x \in c$?" (for some word $x \in \Sigma^*$ of length n)
(2) *Equivalence Query:* "Does r represent c?" (for some representation $r \in \mathcal{F}$)
(3) *Extended Equivalence Query:* "Does the n-input boolean circuit h correctly decide membership in c?"

In (2) and (3) the representation r (or the circuit h) is termed an *hypothesis*. The Teacher responds to a membership query with *yes* or *no* (always correctly). The Teacher responds to an equivalence query (of either kind) either with the information that the hypothesis is correct (and consequently the learning process is complete), or by providing a *counterexample* showing that the representation r (or the circuit h) is incorrect. If the counterexample is a word $x \in c$ not represented by the hypothesis r (accepted by circuit h) we say that the Teacher has provided a *positive counterexample*, and if the counterexample provided by the Teacher is a word $x \notin c$ then we say that a *negative counterexample* has been provided.

We measure the running time of a learning algorithm as a function of the concept size n, and we assume that the Teacher provides n to the Learner at the beginning of the interaction. A *polynomial-time exact learning algorithm for the learning problem* $\mathcal{F}$ is an algorithm which can be executed by the Learner, with each query to the Teacher accounted as taking place in unit time, such that for whatever concept $c = c(r)$ for some $r \in \mathcal{F}$ the Teacher may be teaching, the Learner finishes the algorithm in time polynomial in the size n of c with a *correct equivalent representation* of c. Depending on the flavor of exact learning, this will be either a representation $r' \in \mathcal{F}$ such that $c = c(r')$, or an n-input circuit h that accepts precisely c.

Exact learning comes in various flavors depending on which of the 3 kinds of queries to the Teacher are allowed. For example (discussed further below) the dominating sets in a graph can be learned in polynomial time by extended equivalence and membership queries; the independent sets in a graph can be learned in polynomial time by equivalence queries (where the representations r are graphs). In the sequel, we may refer to these flavors as EE+M, EE, E+M, etc., and by EE+M *learnable* we mean exactly learnable in this flavor in polynomial time.

Our analysis of learning algorithms and reductions will generally focus on the number of queries made by the Learner. (P-time generation of the queries will be obvious.)

Problem Reductions for Exact Learning by Extended Equivalence Queries

Theorems 1–4 are concerned with exhibiting reductions between learning problems. The following notion of reduction relevant to Theorem 1 is a slight modification of the reduction introduced by Pitt and Warmuth

in the PAC setting [16]. Another notion of reduction in this setting appropriate for learning with membership queries has been studied by Angluin and Kharitonov [2].

Definition 1. Let $\mathcal{F}$ and $\mathcal{F}'$ be learning problems. A *positive* EE *reduction* from $\mathcal{F}$ to $\mathcal{F}'$ is a triple (α, β, γ) where:
(1) $\alpha : \mathcal{F} \to \mathcal{F}'$ and $\beta : N \to N$ are *reference functions* with $|\alpha(r)|$ polynomial in $|r|$ and β a polynomial, and
(2) $\gamma : \Sigma^n \to \Sigma^{\beta(n)}$ is a function computable in time polynomial in n, such that:
(3) $\forall r \in \mathcal{F}$ and $\forall x \in \Sigma^n$ where n is the size of the concept $c(r)$, we have $x \in c(r)$ if and only if $\gamma(x) \in c(\alpha(r))$.

If in (3) we replace "$\gamma(x) \in c(\alpha(r))$" with "$\gamma(x) \notin c(\alpha(r))$" then we term this a *negative* EE *reduction*. The proof of the following lemma is straightforward.

Lemma 1. *If $\mathcal{F}$ reduces to $\mathcal{F}'$ and $\mathcal{F}'$ is EE learnable, then $\mathcal{F}$ is EE learnable.*

Proof Sketch. We argue for the case of a positive reduction (α, β, γ); the case of negative reductions is similar. The Learner creates a subroutine S which executes the learning algorithm A' for $\mathcal{F}'$. Repeatedly, the Learner offers to the Teacher the hypothesis (suitably encoded as a circuit) h: $x \in c$ if and only $\gamma(x) \in h'$, where h' is the current hypothesis of S. If the Teacher responds that h is correct then, of course, we are done. Otherwise, the Teacher will produce a counterexample y. The Learner then passes $\gamma(y)$ to the subroutine.

It is straightforward to verify from the definition of reduction that: (1) as seen by S, the interaction is consistent with being taught the concept $c' = c(\alpha(r))$ for a representation $r \in \mathcal{F}$ for which $c = c(r)$, and (2) if S computes a correct hypothesis h' concerning c', then the hypothesis h offered by the Learner to the Teacher will be correct concerning c.

(1) and (2) insure that after no more rounds of interaction than required by A', the Learner will produce a hypothesis concerning c which will be correct (even if the hypothesis of S on which it is based is not correct about c'). □

Parameterized Computational Complexity

A theory of parameterized computational complexity is introduced in [1], [8], [9], [10], [11], [12], [13], to which the reader should refer for further details. The theory is motivated by the observation that many natural problems have two or more inputs, for example, a graph G and a positive integer k. It is sometimes the case that only a small range of parameter values have practical significance; for a number of examples arising in VLSI, computational biology, programming language design, cryptography, and natural language processing, see [13]. Some problems of this form can be solved in time $f(k)|G|^\alpha$ where α is independent of k (for example, Vertex Cover, Graph Genus, and Min Cut Linear Arrangement [14]). For others (e.g., Independent Set, Dominating Set and Bandwidth) we have only brute-force algorithms requiring time $O(|G|^{f(k)})$. This qualitative complexity issue is formalized as follows.

Definition 2. A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$ where Σ is a fixed alphabet. Let $L_y = \{(x, y) : (x, y) \in L\}$. We call L_y the y-th slice of L.

Definition 3. A parameterized problem L is *fixed-parameter tractable* (FPT) if there exists a constant α and an algorithm to determine if (x, y) is in L in time $f(|y|) \cdot |x|^\alpha$, where $f : N \to N$ is an arbitrary function.

Definition 4. A *uniform parameterized reduction* of a parameterized problem L to a parameterized problem L' is an oracle algorithm A that on input (x, y) determines whether $x \in L_y$ and satisfies
(1) There is an arbitrary function $f : N \to N$ and a polynomial q such that the running time of A is bounded by $f(|y|)q(|x|)$.
(2) For each $y \in \Sigma^*$ there is a finite subset $J_y \subset \Sigma^*$ such that A consults oracles only for fixed-parameter decision problems L'_w where $w \in J_y$.

In [8] and [9] complexity classes of parameterized problems are defined based on a finely resolved circuit model of solution checking. These classes form a hierarchy called the W hierarchy:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$$

A large number of well-known combinatorial decision problems are identified as hard or complete for various levels of this hierarchy. For example, Clique and Independent Set are complete for $W[1]$, and Dominating Set is complete for $W[2]$. If $P = NP$ then the hierarchy collapses, and conversely, if the hierarchy collapses, then a *quantitative* version of the $P \neq NP$ conjecture fails [1].

The Slices of a Learning Problem

We consider a parameterization of learning problems that is both natural in many cases and technically useful in exhibiting learning problem reductions.

Definition 5. The *weight* $w(x)$ of a 0-1 vector x is the number of 1's in x (i.e., the *Hamming weight* of x). Let $\mathcal{F}$ be a learning problem and let c be a concept represented by $r \in \mathcal{F}$. The k^{th} *slice* of c is the concept $c_k = \{x \in c : w(x) = k\}$ which also we view to be represented by r in defining the k^{th} *slice* $\mathcal{F}_k$ of $\mathcal{F}$. That is, in $\mathcal{F}_k$ we simply reinterpret the concept represented by $r \in \mathcal{F}$ to be $c(r)_k$.

The following Lemma provides the structure for some of our arguments. Note that in the hypothesis, the polynomial $q(n)$ provides a completely explicit bound on the number of queries (i.e., we do not mean $O(q(n))$.

Lemma 2. *Let $\mathcal{F}$ be a family of concepts, and suppose there is a polynomial $q(n)$ and a polynomial-time uniform family of polynomial-time exact learning algorithms A_k for $1 \leq k \leq n$, such that each algorithm A_k learns $\mathcal{F}_k$ by making at most $q(n)$ extended equivalence queries. Then there is an exact polynomial-time learning algorithm A for $\mathcal{F}$ that makes at most $n \cdot q(n)$ extended equivalence queries.*

Proof Sketch. Since the Learner makes only extended equivalence queries, the algorithm consists of some number of rounds of (1) the Learner presenting a hypothesis, and (2) the Teacher presenting a counterexample. The Learner essentially runs the algorithms $A_1, \ldots, A_n$ in parallel, presenting at each occasion (1) the *collective current hypothesis*: $x \in c$ if and only if $w(x) = k$ and x is accepted by the current hypothesis of A_k. When the Teacher responds with a counterexample y, this is "refered" to the algorithm A_j, where $j = w(y)$. Each A_i independently learns the i^{th} slice $\mathcal{F}_i$ of $\mathcal{F}$. After at most $n \cdot q(n)$ rounds each slice has been learned correctly. The uniformity hypothesis insures that the slice

algorithms A_k can be generated in polynomial time by the Learner. □

Lemma 2 shows how we can learn $\mathcal{F}$ by learning the slices $\mathcal{F}_k$ of $\mathcal{F}$. The following easy lemma describes a passage in the other direction.

Lemma 3. *If $\mathcal{F}$ is EE learnable then for all k, $\mathcal{F}_k$ is EE learnable.*

Proof Sketch. Let A denote a learning algorithm for $\mathcal{F}$. The Learner simply executes A with the modification that the hypothesis offered to the teacher is: $x \in c_k$ if and only if $w(x) = k$ and x is accepted by the current hypothesis offered by A regarding c. From the point of view of (the subroutine) A, it is just as if one were interacting with a Teacher of c who (strangely) only offered counterexamples of a certain size. In no more rounds than required by A the Learner will produce a correct hypothesis regarding c_k. □

We remark that Lemma 3 seems to fail for EE+M learning. The proof of Lemma 3 actually gives us a little more, which we capture with the following definition.

Definition 6. A learning problem $\mathcal{F}$ is *slicewise uniformly learnable* if there is a polynomial q and a (parameterized) learning algorithm L such that given k and n by the Teacher, $1 \leq k \leq n$, L correctly learns $\mathcal{F}_k$ (by interacting with the Teacher of $\mathcal{F}_k$) in time bounded by $q(n)$. That is, the single learning algorithm L can be "set" by the parameter k to solve any of the learning problems $\mathcal{F}_k$ in time bounded by the polynomial q. We say that $\mathcal{F}$ is *fixed-parameter learnable* if there is a parameterized learning algorithm L for $\mathcal{F}$ which learns $\mathcal{F}_k$ in time $f(k) \cdot n^{\alpha}$ where α is independent of k.

Lemma 3 shows that if $\mathcal{F}$ is EE learnable, then $\mathcal{F}$ is slicewise uniformly EE learnable.

3. To Learn DNF Learn Dominating Sets

For convenience, we shift the venue from DNF to CNF. The following is straightforward (and well-known).

Lemma 4. *(1) DNF is EE (EE+M) learnable if and only if CNF is EE (EE+M) learnable. (2) Monotone DNF is EE (EE+M) learnable if and only if monotone CNF is EE (EE+M) learnable.*

Proof Sketch. The proof for EE learning is easy to see using Lemma 1 and considering compositions of reductions based on the reference maps α that map an expression e either to its negation or to its bitwise complement. This argument can be strengthened slightly to establish the case for EE+M learning. □

We next describe a parameterized complexity reduction between the problems Weighted Satisfiability and Dominating Set. We will use this reduction to simultaneously prove both Theorems 1 and 2. We will subsequently point out an easier reduction that can be used for Theorem 1 (and later Theorem 2) but not for Theorem 3. This reduction is used in [9] to show the completeness of Dominating Set for the parameterized complexity class $W[2]$.

A *dominating set* of vertices in a graph $G = (V, E)$ is a set of vertices $V' \subseteq V$ such that for every vertex $u \in V$, either $u \in V'$ or $uv \in E$ for some vertex $v \in V'$. It is easy to see that the dominating sets in a graph G are in a natural 1:1 correspondence with the truth assignments to a monotone CNF formula which has one clause for each neighborhood in G. By the *weight* of a truth assignment to a set of boolean variables, we mean the number of variables assigned the value *true*.

Dominating Set
Instance: A graph $G = (V, E)$.
Parameter: A positive integer k.
Question: Does G have a k-element dominating set?

Weighted Satisfiability
Instance: A boolean expression X in conjuctive normal form.
Parameter: A positive integer k.
Question: Is there a truth assignment of weight k that satisfies X?

Lemma 5. [11] *There is a uniform parameterized reduction from Weighted Satisfiability to Dominating Set.*

Proof. Let X be a Boolean expression in conjuctive normal form consisting of m clauses $C_1, ..., C_m$ over the set of n variables $x_0, ..., x_{n-1}$. We show how to produce in polynomial-time by local replacement, a graph $G = (V, E)$ that has a dominating set of size $2k$ if and only if X is satisfied by a truth assignment of weight k.

The vertex set V of G is the union of the following sets of vertices:
$V_1 = \{a[r, s] : 0 \leq r \leq k-1, 0 \leq s \leq n-1\}$
$V_2 = \{b[r, s, t] : 0 \leq r \leq k-1, 0 \leq s \leq n-1, 1 \leq t \leq n-k+1\}$
$V_3 = \{c[j] : 1 \leq j \leq m\}$
$V_4 = \{a'[r, u] : 0 \leq r \leq k-1, 1 \leq u \leq 2k+1\}$
$V_5 = \{b'[r, u] : 0 \leq r \leq k-1, 1 \leq u \leq 2k+1\}$
$V_6 = \{d[r, s] : 0 \leq r \leq k-1, 0 \leq s \leq n-1\}$

For convenience, we introduce the following notation for important subsets of some of the vertex sets above. Let
$A(r) = \{a[r, s] : 0 \leq s \leq n-1\}$
$B(r) = \{b[r, s, t] : 0 \leq s \leq n-1, 1 \leq t \leq n-k+1\}$
$B(r, s) = \{b[r, s, t] : 1 \leq t \leq n-k+1\}$

The edge set E of G is the union of the following sets of edges. In these descriptions we implicitly quantify over all possible indices.
$E_1 = \{c[j]a[r, s] : x_s \in C_j\}$
$E_2 = \{a[r, s]a[r, s'] : s \neq s'\}$
$E_3 = \{b[r, s, t]b[r, s, t'] : t \neq t'\}$
$E_4 = \{a[r, s]b[r, s', t] : s \neq s'\}$
$E_5 = \{b[r, s, t]d[r, s'] : s' \neq s + t \pmod{n}\}$
$E_6 = \{a[r, s]a'[r, u]\}$
$E_7 = \{b[r, s, t]b'[r, u]\}$
$E_8 = \{c[j]b[r, s, t] : \exists i\ \overline{x_i} \in C_j, s < i < s + t\}$
$E_9 = \{d[r, s]a[r', s] : r' = r + 1 \pmod{k}\}$

Suppose X has a satisfying truth assigment τ of weight k, with variables $x_{i_0}, x_{i_1}, ..., x_{i_{k-1}}$ assigned the value *true*. Suppose $i_0 < i_2 < ... < i_{k-1}$. Let $d_r = i_{r+1 (\mathrm{mod} k)} - i_r \pmod{n}$ for $r = 0, ..., k-1$. It is straightforward to verify that the set of $2k$ vertices

$$D = \{a[r, i_r] : 0 \leq r \leq k-1\} \cup \{b[r, i_r, d_r] : 0 \leq r \leq k-1\}$$

is a dominating set in G.

Conversely, suppose D is a dominating set of $2k$ vertices in G. The closed neighborhoods of the $2k$ vertices $a'[0, 1], ..., a'[k-1, 1], b'[0, 1], ..., b'[k-1, 1]$ are disjoint, so D must consist of exactly $2k$ vertices, one in each of these closed neighborhoods. Also, none of the vertices of $V_4 \cup V_5$ are in D, since if $a'[r, u] \in D$ then necessarily $a'[r, u'] \in D$ for $1 < u' < 2k+1$ (otherwise D fails to be dominating), which contradicts that D contains exactly

$2k$ vertices. It follows that D contains exactly one vertex from each of the sets $A(r)$ and $B(r)$ for $0 \leq r \leq k-1$.

The possibilities for D are further constrained by the edges of E_4, E_5 and E_9. The vertices of D in V_1 represent the variables set to *true* in a satisfying truth assignment for X, and the vertices of D in V_2 represent intervals of variables set to *false*. Since there are k variables to be set to *true* there are, considering the indices of the variables mod n, also k intervals of variables to be set to *false*.

The edges of E_4, E_5 and E_9 enforce that the $2k$ vertices in D must represent such a choice consistently. To see how this enforcement works, suppose $a[3,4] \in D$. This represents that the third of k distinct choices of variables to be given the value *true* is the variable x_4. The edges of E_4 force the unique vertex of D in the set $B(3)$ to belong to the subset $B(3,4)$. The index of the vertex of D in the subset $B(3,4)$ represents the difference (mod n) between the indices of the third and fourth choices of a variable to receive the value *true*, and thus the vertex represents a range of variables to receive the value *false*. The edges of E_5 and E_9 enforce that the index t of the vertex of D in the subset $B(3,4)$ represents the "distance" to the next variable to be set *true*, as it is represented by the unique vertex of D in the set $A(4)$.

It remains only to check that the fact that D is a dominating set insures that the truth assignment represented by D satisfies X. This follows by the definition of the edge sets E_1 and E_8. □

There are two important aspects of the proof which require our notice in this context. (1) In the proof of Lemma 5, the reduction described is accomplished in time polynomial in n and k. (2) For each slice, the reduction satisfies the definition of an EE learning reduction.

Proof Sketch for Theorem 1. (Theorem 3 works by an essentially similar argument). By Lemma 4, it is enough to show that the problem of learning the truth assignments to a polynomial-sized CNF expression reduces to learning the dominating sets in a graph, a special case of monotone CNF. By Lemma 3, if the dominating sets in a graph can be learned in polynomial time by extended equivalence queries, then also the k-element dominating sets (for $1 \leq k \leq n$) can be slicewise uniformly learned in polynomial time in this model. By Lemma 5, the above observations, and a diagonalization trick to handle the problem of supplying the subroutines the correct concept sizes, the slices of CNF reduce to the slices of Dominating Sets, in total time polynomial in n, and so by Lemma 2 we are done. □

Alternative Proof Sketch for Theorem 1. For those interested only in Theorem 1, we remark that Theorem 1 can also be obtained with essentially the same argument but from a simpler fairly standard reduction from CNF to Dominating Sets:
Let X be as in the proof of Lemma 5. This time the vertex set of G is the union of the sets $V_1, ..., V_4$ below:
$V_1 = \{x[j] : j = 0, ..., n-1\}$
$V_2 = \{y[j] : j = 0, ..., n-1\}$
$V_3 = \{z[j] : j = 0, ..., n-1\}$, and
$V_4 = \{c[j] : j = 1, ..., m\}$.
The edge sets of G is the union of the sets $E_1, ...E_5$ below:
$E_1 = \{x[j]y[j] : j = 0, ..., n-1\}$
$E_2 = \{y[j]z[j] : j = 0, ..., n-1\}$
$E_3 = \{x[j]z[j] : j = 0, ..., n-1\}$,
$E_4 = \{c[i]y[j] : x_j \in C_i\}$, and
$E_5 = \{c[i]z[j] : \overline{x_j} \in C_i\}$.
We say a dominating set has *correct form* if it does not involve $x[j]$ or $c[i]$ for any i or j. The point is that X is satisfiable if and only if G has a dominating set of size n if and only if G has a dominating set of size n in correct form. Since we can recognise when a dominating set is in correct form, we can recognise which correspond to valid truth assignments and hence since we are only using EE queries the argument goes through. □

Proof Sketch for Theorem 2. We handle "if" by improving on the argument for Theorem 1, noting that if the subroutine L_k of the Learner which is devoted to the k^{th} slice wants to make a membership query about the vector x, then in consideration of the structure of the graph in the proof of Lemma 5, $x \notin c_k$ if x fails to meet certain conditions. In particular, we must have $w(x) = k$ and the vertex set indicated by x must contain exactly one vertex in each of the sets $A(r)$ for $0 \leq r \leq k-1$, or the Learner can supply (consistently) the answer "no" without consulting the Teacher. If these conditions are met then the Learner can compute a truth assignment of weight k corresponding to x and make a membership query to the Teacher in order to obtain the correct answer to pass to the subroutine L_k.

Conversely, suppose that the Learner is informed of the parameter k and the number of vertices n in the graph. Let A denote a polynomial-time algorithm for EE+M learning of CNF. The Learner creates a subroutine S which executes A, initially passing to S the concept size kn. The Learner interacts with S according to a "mental model" based on the following reduction of Dominating Set to Satisfiability (complementary, in some sense, to Lemma 5). Let $e(G,k)$ denote the CNF expression in the variables $a[i,j]$ for $1 \leq i \leq k$ and $1 \leq j \leq n$ described as follows. (We may assume that the vertex set of G is $\{1, ..., n\}$.)

$$e(G,k) = e_1(G,k) \cdot e_2(G,k) \cdot e_3(G,k)$$

where

$$e_1(G,k) = \prod_{i=1}^{k} \prod_{1 \leq r < s \leq n} (\neg a[i,r] + \neg a[i,s]) \cdot q_k,$$

$$\text{where } q_k = \prod_{j=1}^{n} \prod_{1 \leq r < s \leq k} (\neg a[r,j] + \neg a[s,j])$$

and

$$e_2(G,k) = \prod_{1 \leq u \leq n} \left(\sum_{v \in N[u]} \sum_{i=1}^{k} a[i,v] \right)$$

and

$$e_3(G,k) = \prod_{i=1}^{k} \sum_{j=1}^{n} a[i,j]$$

It is easy to observe (1) that any truth assignment satisfying $e(G,k)$ has weight exactly k and (2) any truth assignment τ satisfying $e_1(G,k)$ corresponds naturally with a k-element set of vertices in G that is a dominating set if and only if τ also satisfies $e_2(G,k)$.

From the point of view of S, the Learner behaves as if teaching $e(G,k)$. Even though $e(G,k)$ is only partly known to the Learner, this can be accomplished in the following way. If S makes a membership query to the Learner, then the Learner responds "no" immediately if the query truth assignment does not have weight k or if it fails to satisfy $e_1(G,k)$ (which the Learner can determine). If the query truth assignment has weight k and satisfies $e_1(G,k)$ then it corresponds to a set of k vertices in G about which the Learner queries the Teacher in order to determine the correct answer to pass to the subroutine S. Handling equivalence queries is similar. In no more rounds of interaction than required by A, the subroutine S will produce a correct hypothesis about the truth assignments satisfying $e(G,k)$. This yields by the obvious translation a correct hypothesis concerning the k-element dominating sets in G. □

Theorem 4 is proved by a similar argument. As with Theorems 1 and 3, there is also a simpler proof of Theorem 2 that does not, however, lead to a proof of Theorem 4. For this, we augment the construction employed in the alternate proof of Theorem 1 with $2n$ additional vertices

$$\{p[j,t] : t = 1, ..., 2n\},$$

and with $4n$ additional edges

$$\{p[j,t]y[j] : t = 1, ..., 2n\} \cup \{q[j,t]z[j] : t = 1, ..., 2n\}$$

These additional edges and vertices force any n element dominating set to be of the correct form.

By Theorem 2, if polynomial-sized DNF (CNF) formulas are EE+M learnable then the k-element dominating sets in a graph can be learned in time polynomial in k and the size n of the graph. Such an outcome might be considered surprising, since the best known algorithm at present makes $O(n^k)$ queries. Are the k-element dominating sets at least fixed-parameter learnable?

4. Graph-Theoretic Learning Problems

Theorems 2 and 4 invite us to further investigate the natural problems of learning graph structures. This area seems very poorly developed. Together with Lane Clark and Walter Wallis, Evans and Fellows have proven the following concerning the learnablity of vertex sets. Details and further results will appear elsewhere [7]. We remark that some of the algorithmic strategies (e.g., for Proposition 3), are taken from the theory of fixed parameter tractability.

Proposition 1. *The independent sets (the cliques) in a graph are E learnable with $O(n^2)$ queries.*
Proof Sketch. We give the argument for independent sets. The Learner begins with the hypothesis H of the complete graph. If the graph G being taught is not complete, then the Teacher must respond with a positive counterexample V'. Since every singleton set is independent, V' must contain at least two vertices. The Learner can deduce that there are no edges in G between vertices in V'. The algorithm makes only equivalence queries. At each stage, the Learner presents a hypothesis graph H that contains edges between all pairs of vertices except those pairs for which the Learner has deduced that no edge is present in G. It follows that the Teacher must respond with a positive counterexample, and this must allow for the non-presence of an edge to be deduced for at least one new pair of vertices. The algorithm will terminate in at most $\binom{n}{2}$ rounds. □

Proposition 2. *The vertex covers in a graph are E learnable with $O(n^2)$ queries.*
Proof Sketch. Similar to Proposition 1. □

Proposition 3. *The k-element vertex covers in a graph are E learnable with $2^{2^{2k}+1}$ queries (independently of the size of the graph).* □
Proof Sketch. A complicated argument based on an extremal theorem concerning minimal vertex covers, and the tree-search technique discussed in [13]. □

Proposition 4. *The dominating sets in a graph are EE+M learnable with $O(n^2)$ queries.*
Proof Sketch. This can be done by adapting Valiant's algorithm for learning monotone DNF [18]. The algorithm can be briefly described as follows. The Learner begins with the hypothesis that the graph is complete, i.e., that any non-empty set of vertices is a dominating set. If this is not the case, the Teacher is obliged to produce a negative counterexample: a set S of vertices that is not dominating set. By augmenting S, making at most n membership queries, the Learner can identify a *maximal* non-dominating set of vertices S'. Of necessity, S' is the closed neighborhood of some vertex u. By repeatedly identifying the closed neighborhoods in the graph in this way, the Learner can compute a circuit that correctly identifies the dominating sets, since a dominating set is precisely a set having non-empty intersection with each closed neighborhood. □

The evidence to date suggests that perhaps only relatively "simple" concepts can be learned in polynomial time. The W hierarchy may be a useful reference structure in exploring polynomial-time learnability, because it makes a finely resolved classification of problems according to the complexity of checking solutions. For example, Vertex Cover is in FPT, Independent Set is complete for $W[1]$ and Dominating Set is complete for $W[2]$. The propositions above suggest that we might look for the boundary between what can be learned in polynomial time and what cannot, "between $W[1]$ and $W[2]$." Several natural problems in this range are identified in [11].

5. VC Dimension is Complete for $W[1]$

The following concept has proved to be of some importance in proving lower bounds in computational learning theory.

Definition. The *Vapnik-Chervonenkis dimension* (VC dimension) of a family of subsets F of a base set X is the maximum cardinality of a set $S \subseteq X$ such that for every subset $S' \subseteq S$, $\exists Y \in F$ such that $S \cap Y = S'$. In general, we say that such a subset S' of S is *generated* in S by F. The VC dimension of F is thus the largest cardinality of $S \subseteq X$ such that every subset of S is generated by F.

VC Dimension
Instance: A family of subsets F of a base set X.
Parameter: A positive integer k.
Question: Is the VC dimension of F at least k?

The VC dimension of a family of sets $\mathcal{F}$ over a base set X of cardinality n can be shown to be at most $\log n$. Consequently, the above problem is unlikely to be NP-

complete [17]. In the following theorem, both membership and hardness for the parameterized complexity class $W[1]$ involve reductions that are exponential in the parameter k. Note that this is permitted in the theory of parameterized complexity (see Definition 4).

Theorem 3. *VC Dimension is complete for* $W[1]$.

Proof. A proof of membership in $W[1]$ can be found in [13]. We argue that VC Dimension is hard for $W[1]$ by a reduction from Clique.

Given a graph $G = (V, E)$ and a positive integer k we describe how to compute a family of sets F over a base set X, so that F has V-C dimension k if and only if G has a k-clique. The cardinality of the family F that we will describe is $O(k^2n^2 + 2^k)$, and the size of the base set X is kn, where n is the order of G. For convenience we will assume that $V = \{1, \ldots, n\}$. We write $[m]$ to denote the set $\{1, \ldots, m\}$.

The base set X is simply:

$$X = \{(u, i) : u \in V, \quad i \in [k]\}$$

The family F consists of four subfamilies, $F = F_0 \cup F_1 \cup F_2 \cup F_3$ which are described as follows. (These correspond, roughly, to the cardinality of the sets in the subfamilies.)

$$F_0 = \{\emptyset\}$$

$$F_1 = \{\{(u, i)\} : u \in V, \quad i \in [k]\}$$

$$F_2 = \{\{(u, i), (v, j)\} : uv \in E, \quad i, j \in [k]\}$$

$$F_3 = \{\{(u, i) : i \in S\} : S \subseteq [k], \#(S) \geq 3\}$$

To see this that this construction works, let C be the clique in G and let f be any 1:1 map from C to $\{1, \ldots, k\}$. Consider the set $S \subseteq X$ of cardinality k:

$$S = \{(u, f(u)) : u \in C\}$$

If $S' \subseteq S$ has cardinality at least 3, then it is generated by the corresponding set in F_3. It is straightforward to verify that subsets of S of cardinality smaller than 3 are generated by $F_0 \cup F_1 \cup F_2$.

Conversely, suppose S is a k-element subset of X, every subset of which is generated by F. For each subset $S' \subseteq S$ choose a witness $W \in F$ with $W \cap S = S'$. If S' has cardinality at least 3, then its witness must be chosen from F_3. But this implies that every set in F_3 must serve as a witness for some $S' \subseteq S$ of cardinality at least 3.

The witnesses for sets $S' \subseteq S$ of cardinality 2 must therefore belong to F_2. We cannot have both (u, i) and (u, j) in S for $i \neq j$, else there is no witness possible for the 2-element set consisting of these (by the definition of F_2). Consequently S must range over k different vertex indices. The fact that there are witnesses for all of the 2-element subsets implies that there is a corresponding k-clique in G. □

References

[1] K. Abrahamson, R. Downey and M. Fellows. Fixed-parameter intractability II. *Proceedings Tenth Symposium on Theoretical Aspects of Computing* (1993) 374-385.

[2] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of the 23rd Annual ACM Syposium on Theory of Computing* (1991), 444-454. New York, May 1991. ACM Press.

[3] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation* 75 (1987), 87-106.

[4] D. Angluin. Queries and concept learning. *Machine Learning* 2 (1987), 319-342.

[5] D. Angluin. Negative results for equivalence queries. *Machine Learning* 5 (1990), 121-150.

[6] D. Angluin. Computational Learning theory: survey and selected bibliography. *Proceedings 24th ACM Symposium on the Theory of Computing* (1992) 351-369.

[7] L. Clark, P. Evans, M. Fellows and W. Wallis. Algorithms for learning and teaching sets of vertices in graphs. University of Victoria Technical Report DCS-212-IR (1993).

[8] R. Downey and M. Fellows. Fixed parameter tractability and completeness. *Congr. Num.*, 87 (1992) 161-187.

[9] R. Downey and M. Fellows. Fixed parameter tractability and completeness I: basic results. To appear.

[10] R. Downey and M. Fellows. Fixed parameter tractability and completeness II: on completeness for $W[1]$. To appear.

[11] R. Downey and M. Fellows. Fixed parameter intractability (extended abstract). *Proceedings of the Seventh Annual IEEE Conference on Structure in Complexity Theory* (1992), 36-49.

[12] R. Downey and M. Fellows. Fixed parameter tractability and completeness III: some structural aspects of the W-Hierarchy. To appear in: *Complexity Theory (Proceedings of the 1992 Dagstuhl Workshop on Structural Complexity)*, (ed. Ambos-Spies *et. al.*) Cambridge University Press.

[13] R. Downey and M. Fellows. Parameterized computational feasibility. To appear in: *Feasible Mathematics II* (ed. Clote and Remmel), Birkhauser, Boston.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[15] M. Kearns, M. Li, L. Pitt and L. Valiant. On the learnability of boolean formulae. In *Proceedings of the 19th ACM Symposium on Theory of Computing* (1987), 285-295. ACM Press.

[16] L. Pitt and M. Warmuth. Prediction-preserving reducibility. *J. of Computer and Systems Sciences* (1990), 430-467.

[17] C. H. Papadimitriou and M. Yannakakis. On the complexity of computing the V-C dimension. Manuscript, May 1992.

[18] L. G. Valiant. A theory of the learnable," *Communications of the ACM* 27 (1984), 1134-1142.