Projective DNF Formulae and Their Revision

Robert H. Sloan[1]⋆, Balázs Szörényi[2], and György Turán[3]⋆⋆

[1] U. of Illinois at Chicago, sloan@uic.edu
[2] U. of Szeged, Hungary, szorenyi@inf.u-szeged.hu
[3] U. of Illinois at Chicago and RGAI Hungarian Academy of Sciences, gyt@uic.edu

1 Introduction

Valiant argued that biology imposes various constraints on learnability, and, motivated by these constraints, introduced his model of *projection learning* [14]. Projection learning aims to learn a target concept over some large domain, in this paper $\{0,1\}^n$, by learning some of its projections to a class of smaller domains, and combining these projections. Valiant proved a general mistake bound for the resulting algorithm under certain conditions. The basic assumption underlying projection learning is that there is a family of simple projections that cover all positive instances of the target, where simple means belonging to some efficiently learnable class. The projections describing the target in this way can also be thought of as a set of experts, each specialized to classify a subset of the instances, such that whenever two experts overlap they always agree in their classification.

Perhaps the most natural special case of this framework, also discussed by Valiant, is when the projection domains are subcubes of a fixed dimension, and the restrictions of the target to these domains are conjunctions. In this case, the algorithm learns a class of disjunctive normal forms (DNF) called *projective* DNF, which appears to be a new class of DNF expressions. As the learnability of DNF is a major open problem, it is of interest to identify new learnable subclasses and to understand their scope.

We give some basic properties of projective DNF by comparing them to standard classes such as DNF with terms of bounded size or with a bounded number of terms, and we present lower and upper bounds for the *exclusion dimension* of projective DNF. The exclusion dimension, or certificate size [1, 6, 7], of a concept class is a combinatorial parameter that is closely related to its learning complexity in the model of proper learning with equivalence and membership queries. Thus we obtain bounds for the complexity of learning projective DNF in this model. For the subclass of 1-projective DNF we prove a characterization theorem that gives an explicit description of all such expressions.

One of the main goals of Valiant's work is to find *attribute-efficient* learning algorithms. The notion of an attribute-efficient learning algorithm goes back to the seminal work of Littlestone [10]. His Winnow algorithm learns a (monotone) disjunction of k variables out of a total of n variables with $O(k \log n)$ mistakes.

⋆ Partially supported by NSF grant CCR-0100336.
⋆⋆ Partially supported by NSF grant CCR-0100336.

Thus Winnow is very efficient for learning problems where there are many attributes but most of them are irrelevant. It is argued that learning in the brain, and also many real-life applications have exactly that property [14, 15]. In general, a learning algorithm is called attribute efficient if it has a mistake bound that is polynomial in the number of relevant variables, and only polylogarithmic in the total number of variables. One of the most attractive features of Valiant's projection learning algorithm is its attribute efficiency.

The apparently unrelated area of *theory revision* in machine learning is concerned with the revision, or correction, of an initial theory that is assumed to be a good approximation of a correct theory. A typical application of theory revision is the refinement of an initial version of a theory provided by a domain expert. In that context it is argued that a large and complex theory cannot be learned from scratch, but it is necessary to start with a reasonably close initial theory. The usual assumption of theory revision is that the correct theory can be obtained from the initial one by a small number of syntactic modifications, such as the deletion or the addition of a literal. An efficient revision algorithm is required to be polynomial in the number of literals that need to be modified and polylogarithmic in the total number of literals. Wrobel surveys the complete theory revision literature [16, 17]; efficient revision in the framework of learning with queries is discussed in detail in [4, 5].

Thus, the situation in theory revision is similar to the case of attribute-efficient learning, but instead of assuming that only a few literal occurrences are relevant, one assumes that only a few literal occurrences need to be modified. Roughly speaking, attribute efficient learning requires efficient revision of an empty initial formula. The argument for the biological relevance of attribute-efficient learning can be extended to apply to revision, for example, in the case of information hard-wired at birth.

In view of this relationship, it is an interesting general question whether attribute-efficient learnability results can be extended to results on efficient revision. Previously we gave a positive answer in the case of parity function, where the relationship between the two tasks is straightforward [4]. As a next step, we show here that Winnow is in fact an efficient algorithm for revising disjunctions as well. This in turn raises the question whether Valiant's more general results on the attribute-efficient learnability of projective DNF can also be extended to efficient revision. We show that projective DNF can be revised efficiently by using, just as Valiant does, the original Winnow algorithm on two levels. In our setting, the initial weights are adapted to the task of revision. The correctness proof uses a potential function argument.

We note that the problem of revising is somewhat related to the problem of tracking changing target concepts, which is discussed in several previous papers (see, e.g., [2, 12]). Compared to the model of tracking a changing target concept, our model is less general, as it assumes only a single change from an initial concept to the target. On the other hand, the tracking model starts from scratch, and thus, in order to simulate our setting, the initial formula (which may be large) has to be built up by adding all its relevant variables, and all these additions may enter into the mistake bound of the algorithm. Thus it appears that there is no direct implication between our results and those of Auer and Warmuth on tracking disjunctions [2].

After some preliminaries given in Section 2, Section 3 contains the definition of projective DNF and a discussion of some of their properties, and Section 4 contains the bounds for the exclusion dimension. The characterization of 1-PDNF is presented in Section 5. Section 6 contains the revision algorithms.

2 Preliminaries

The weight of $\mathbf{x} \in \{0,1\}^n$ is the number of its ones, denoted $|\mathbf{x}|$. The vector $\mathbf{e}_I^n$ is the characteristic vector of $I \subseteq [n]$, and the vector $\mathbf{g}_I^n$ is its componentwise complement ($[n] = \{1, \ldots, n\}$). The all 1's vector, $\mathbf{e}_{[n]}^n$, is written $\mathbf{1}$; the all 0's vector is written $\mathbf{0}$. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we write $T(f) = \{\mathbf{x} \in \{0,1\}^n : f(\mathbf{x}) = 1\}$. For $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$ we write $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $1 \leq i \leq n$, and we write $\mathbf{x} \wedge \mathbf{y}$ (resp. $\mathbf{x} \vee \mathbf{y}$) for $(x_1 \wedge y_1, \ldots, x_n \wedge y_n)$ (resp. $(x_1 \vee y_1, \ldots, x_n \vee y_n)$).

A Boolean function f is monotone if $\mathbf{x} \leq \mathbf{y}$ implies $f(\mathbf{x}) \leq f(\mathbf{y})$, it is $\mathbf{a}$-unate if it holds that $g(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{a})$ is monotone, where $\mathbf{a} \in \{0,1\}^n$ and $\oplus$ is componentwise exclusive-or, and it is unate if it is $\mathbf{a}$-unate for some $\mathbf{a} \in \{0,1\}^n$.

We use the standard model of *mistake bounded* learning [10], which proceeds in a sequence of *rounds* (or trials). In each round the learner receives an instance $\mathbf{x}$, and produces a prediction $\hat{y}$. Then the correct classification y is revealed. If $\hat{y} \neq y$ then the learner made a *mistake.* The mistake bound of the learning algorithm is the maximal number of mistakes, taken over all possible runs, i.e., sequences of instances.

In addition to the standard mistake-bounded model, as a technical tool for the learning result, we also consider the more general model of learning monotone disjunctions with *attribute errors* (Auer and Warmuth [2], also used by Valiant [14] with a different name). In this model y may *not* be the correct classification of $\mathbf{x}$. It is assumed that the error comes from some components of $\mathbf{x}$ being incorrect, and the number of attribute errors committed in a round is the minimal number of components that need to be changed in order to get the correct classification. More precisely, if in round r the y_r is not the correct classification of $\mathbf{x}_r$, and $y_r = 1$ then $AttrErr(r) = 1$, and if $y_r = 0$ then $AttrErr(r)$ is the number of variables, which are included in the target disjunction and which are set to 1 in $\mathbf{x}_r$. The total number of attribute errors for a given run, denoted $\#AttributeErrors$, is the sum of the attribute errors of the rounds.

For a conjunction (or term) t, we write $Lit(t)$ for the set of literals in t. A term is monotone if it consists of unnegated variables. Given $\mathbf{a} \in \{0,1\}^n$, a term is $\mathbf{a}$-unate if every literal in it is unnegated iff the corresponding component of $\mathbf{a}$ is 0. A *subcube* (or simply *cube*) is any set of vectors that is of the form $T(t)$ for some conjunction (i.e., term) t. For terms t_1, t_2 it holds that $t_1 \to t_2$ iff $T(t_1) \subseteq T(t_2)$ iff $Lit(t_1) \supseteq Lit(t_2)$.

Proposition 1 *A set $A \subseteq \{0,1\}^n$ is a cube iff for every $\mathbf{x}, \mathbf{y} \in A$ and every $\mathbf{z} \in \{0,1\}^n$ such that $\mathbf{x} \wedge \mathbf{y} \leq \mathbf{z} \leq \mathbf{x} \vee \mathbf{y}$, it also holds that $\mathbf{z} \in A$.*

It follows, in particular, that if a cube contains two vectors with weights $w_1 < w_2$, then it also contains vectors of weight w for every $w_1 < w < w_2$.

3 Projective DNF

In this section we introduce projective disjunctive normal forms and we briefly discuss some of their properties.

Definition 2 *A DNF formula φ is a k-*projective DNF*, or k-*PDNF *if it is of the form*

$$\varphi = \rho_1 c_1 \vee \cdots \vee \rho_\ell c_\ell,$$

where every ρ_i, referred to as a k-conjunction, is a conjunction of exactly k literals, c_i is a conjunction and for every i it holds that

$$\rho_i \varphi \equiv \rho_i c_i. \tag{1}$$

Note that in order to specify a k-PDNF, it is *not* sufficient to specify its terms, but for each term one has to specify its ρ-part and its c-part; that is, the projection and the corresponding conjunction have to be distinguished. If necessary, we indicate this distinction by placing a dot between the two parts. For example,

$$(x \cdot y) \vee (z \cdot y) \qquad \text{and} \qquad (x \cdot y) \vee (\bar{x} \cdot yz) \tag{2}$$

are two different 1-PDNF for the same function. The dots are omitted whenever this does not lead to confusion. The conjunctions ρ_i and c_i may have common literals in general. The requirement (1) is equivalent to requiring that $\rho_j \rho_i c_i \equiv \rho_i \rho_j c_j$ for every i and j. This makes it easy to verify that a given expression, such as those in (2), is indeed a k-PDNF. It also shows that the disjunction of any set of terms of a k-PDNF is again a k-PDNF.

A function $f : \{0,1\}^n \to \{0,1\}$ is *k-projective* if it can be written as a k-PDNF formula. If a function is k-projective, then it is k'-projective for every k' with $k \leq k' \leq n$. The class of n-variable k-projective functions is denoted by k-PDNF$_n$.

We briefly discuss the relationship between projective DNF and some other standard classes. Let k-DNF$_n$, resp. K-term-DNF$_n$, denote the class of n-variable Boolean functions expressible as a DNF with all terms having size at most k, resp. with at most K terms. Let k-DL$_n$ denote the class of k-decision lists on n variables [13].

Proposition 3 *Assume $k < n$, and let $K = 2^k \binom{n}{k}$. Then*

$$k\text{-}DNF_n \subset k\text{-}PDNF_n \subset K\text{-}term\text{-}DNF_n.$$

Proposition 4 *k-PDNF$_n \subset (k+1)$-DL$_n$.*

We also give some bounds for the number of n-variable k-projective functions. In view of Proposition 3, an upper bound is provided by the straightforward upper bound for the number of K-term DNF's. The exponent of the lower bound matches the exponent of the upper bound for every fixed k.

Proposition 5

$$3^{\lfloor \frac{n}{k+1} \rfloor \binom{\lceil \frac{k}{k+1} n \rceil}{k}} \leq |k\text{-}PDNF_n| \leq 3^{n\, 2^k \binom{n}{k}}.$$

4 Exclusion dimension

We present the definitions of specifying sets and the exclusion dimension for the case of k-projective functions, following the terminology of Angluin [1] ([6, 7] use unique specification dimension and certificate size for the same notion).

Let f be an n-variable Boolean function. A set $A \subseteq \{0,1\}^n$ is a *specifying set* of f (with respect to the class of k-projective functions) if there is at most one k-projective function that agrees with f on A. (So clearly $\{0,1\}^n$ is always a specifying set.) The *specifying set size* of f is

$$spec(f) = \min\{|A| : A \text{ is a specifying set for } f\},$$

and the *exclusion dimension* of the class of n-variable k-projective functions is

$$XD(k\text{-PDNF}_n) = \max\{spec(f) : f \text{ is an } n\text{-variable non-}k\text{-projective function}\}.$$

A function f is *minimally* non-k-projective if it is not k-projective, but any f' with $T(f') \subset T(f)$ is k-projective.

Proposition 6 *If f is minimally non-k-projective, then $spec(f) \geq |T(f)| - 1$.*

We now present a lower and an upper bound for the exclusion dimension of k-PDNF$_n$, which show that for fixed k the exclusion dimension is $\Theta(n^k)$. We begin with a lemma that characterizes k-PDNF, give some examples, and then continue to the main theorem of this section that gives the bound.

Lemma 7 a) *A function f is k-projective iff for every $\mathbf{x} \in T(f)$ there is a k-conjunction ρ such that $\mathbf{x} \in T(\rho)$ and $T(f) \cap T(\rho)$ is a cube.*

b) *If for every $\mathbf{x} \in T(f)$ there is a k-conjunction ρ such that $T(f) \cap T(\rho) = \{\mathbf{x}\}$, then f is k-projective.*

(Proof of lemma omitted due to space constraints.)

We illustrate Lemma 7 with the following example. We claim that the function $f(x_1, x_2, x_3, x_4) = x_1x_2 \vee x_3x_4$ is not 1-projective. Call a vector that violates condition (a) of the lemma *deviant.* It suffices to show that $\mathbf{1}$ is deviant. For symmetry reasons, we only need to show that $T(f) \cap T(x_1)$ is not a cube. Indeed, it contains 1100 and 1011, but it does not contain $1100 \wedge 1011 = 1001$.

Proposition 8 *For every k and $n \geq k+2$ there is a non-k-projective function with $|T(f)| = k+3$.*

Theorem 9 **a)** *$XD(k\text{-}PDNF_n) \leq 3\binom{n}{k} + 1$.*
b) *If $n \geq 4k(k+1)$, then $XD(k\text{-}PDNF_n) \geq \binom{\lfloor n/4 \rfloor}{k} - 1$.*

Proof. The upper bound follows directly from Lemma 7 and Proposition 1.

For the lower bound, in view of Proposition 6, it is sufficient to construct a minimally non-k-projective n-variable function $f_{n,k}$ that takes the value 1 at many points. First we describe the construction in the case when n is even and $k = 1$. Let $n = 2s$, and let $T(f_{n,k}) = \{\mathbf{a}_i = (\mathbf{g}_i^s, \mathbf{e}_i^s) : i = 1, \ldots, s\} \cup \{\mathbf{0}\}$. We claim that $f_{n,k}$ is minimally non-1-projective. The non-1-projectivity of $f_{n,k}$ follows from the fact that $\mathbf{0}$ is deviant: any 1-projection ρ containing $\mathbf{0}$ is of the form $\bar{x}_j$, and thus it contains some vector(s) $\mathbf{a}_i$, but it does not contain any vector of positive weight less than s. Thus, by the remark following Proposition 1, $T(f_{n,k}) \cap T(\rho)$ is not a cube. On the other hand, the $\mathbf{a}_i$'s are *not* deviant for $f_{n,k}$. This holds as they satisfy the condition of part *b)* of Lemma 7: the 1-conjunction x_{s+i} contains only $\mathbf{a}_i$ from $T(f_{n,k})$. Now we show that every f' with $T(f') \subset T(f_{n,k})$ is 1-projective. Indeed, if $f'(\mathbf{0}) = 0$ then this follows from part *b)* of Lemma 7 directly. Otherwise the only thing to note is that if $f'(\mathbf{a}_i) = 0$, then the 1-conjunction $\bar{x}_i$ contains only $\mathbf{0}$ from $T(f')$.

For the construction in the general case we use the following lemma. In the lemma we consider $\{0,1\}^p$ to be the p-dimensional vector space over $GF(2)$.

Lemma 10 *Let A be a $p \times p$ 0-1 matrix such that both A and $A \oplus I$ are non-singular. Assume that $k(k+1) < 2^p$ and define the mapping*

$$h(b_1, \ldots, b_k) = (b_1 \oplus Ab, \ldots, b_k \oplus Ab),$$

where $b_1, \ldots, b_k \in \{0,1\}^p$ and $b = b_1 \oplus \ldots \oplus b_k$. Then it holds that

a) *h is a bijection,*

b) *for every $b_1, \ldots, b_{k-1}$ and $c_1, \ldots, c_k$ there is a b_k different from $b_1, \ldots, b_{k-1}$, such that the components of $h(b_1, \ldots, b_k)$ are all different from the c_i's.*

(Proof of lemma omitted due to space constraints.)

It is easily verified that, for example, the matrix A with all 0's except $a_{1,1} = a_{1,p} = a_{i,i+1} = 1$ (where $i = 2, \ldots, p-1$) satisfies the conditions of the lemma. It is clear from the definition of h that if the b_i's are all different, then the components of $h(b_1, \ldots, b_s)$ are also all different, and if we permute the b_i's then the components of the image are permuted in the same way. Thus if $I = \{b_1, \ldots, b_k\} \subseteq \{0,1\}^p$, then with an abuse of notation we can write $h(I)$ for the k-element subset of $\{0,1\}^p$ formed by the components of $h(b_1, \ldots, b_k)$.

Now let $p = \lfloor \log \frac{n}{2} \rfloor$, and put $s = 2^p$. If I is a k-element subset of $[s]$, define $\mathbf{a}_I = (\mathbf{g}_I^s,\ \mathbf{e}_{h(I)}^s,\ 0^{n-2s})$, and let $T(f_{n,k}) = \{\mathbf{a}_I : I \subseteq [s], |I| = k\} \cup \{\mathbf{0}\}$.

We claim that $f_{n,k}$ is minimally non-k-projective. This follows very similarly to the special case above. □

Using the results on the relation between the exclusion dimension and the complexity of learning with membership and proper equivalence queries [1, 6, 7] we get the following.

Proposition 11 *The class $k-PDNF_n$ can be learned with $O\left(n\, 2^k \binom{n}{k}^2\right)$ membership and proper equivalence queries.*

The number of queries is polynomial in n for every fixed k. On the other hand, the running time of the learning algorithm is not necessarily polynomial.

Blum [3], using ideas from Littlestone and Helmbold et al. [8, 11], shows that 1-DL is efficiently learnable in the mistake-bounded model. It follows from a straightforward generalization of this result and Proposition 4 that for every fixed k, the class k-PDNF is learnable with polynomially many *improper* equivalence queries and with polynomial running time.

5 A characterization of 1-PDNF

In this section we give a description of 1-projective functions. First let us note that if φ is a 1-PDNF with two complementary projections, i.e., of the form $xt_1 \vee \bar{x}t_2 \vee \cdots$ for some variable x, then by deleting everything else besides these two terms, we get an equivalent formula. Indeed, as every $\mathbf{x}$ satisfying φ satisfies either x or $\bar{x}$, it follows from the definition of projective DNF that $\mathbf{x}$ also satisfies the corresponding term.

We formulate a notion of irredundancy for 1-PDNF, which we call p-irredundancy to distinguish it from the usual notion of irredundancy for DNF. Unlike the standard notion, p-irredundancy of a 1-PDNF is easy to decide.

Definition 12 *A 1-PDNF formula $\varphi = \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell$ is* p-irredundant *if the following conditions hold:*

a) $Lit(\rho_i t_i) \not\subseteq Lit(\rho_j t_j)$ for every $1 \le i, j \le \ell$,
b) $\rho_i \notin Lit(t_i)$ for every $1 \le i \le \ell$,
c) if $\ell \ge 3$ then $\rho_i \ne \bar{\rho}_j$ for every $1 \le i, j \le \ell$.

The first condition says that no term implies another, the second that in each term the projection and conjunction parts are disjoint, and the third that if there are at least three terms, then no two projections are complementary.

Proposition 13 *There is a polynomial algorithm which, given a 1-projective DNF, transforms it into an equivalent p-irredundant 1-projective DNF.*

First we give a description of those p-irredundant 1-projective DNF that represent either a monotone or an $\mathbf{a}$-unate function, and then we give the general description. We assume throughout this section that the 1-PDNF formulas in question are nonempty; thus they represent a function that is not identically 0.

Theorem 14 *A formula φ is a p-irredundant 1-PDNF formula representing a monotone (resp.* **a***-unate) function if and only if it is either of the form*

$$\varphi = \rho_1 t \vee \cdots \vee \rho_\ell t,$$

where $\rho_1, \ldots, \rho_\ell$ are different unnegated variables (resp. literals whose signs agree with **a***) not contained in t, and t is a monotone (resp.* **a***-unate) term, or t is of the form*

$$\varphi = \rho t \vee \bar{\rho} t t',$$

where ρ is an unnegated variable (resp. agrees with **a***) and t, t' are monotone (resp.* **a***-unate) terms not containing ρ or $\bar{\rho}$.*

Proof. We prove only the monotone case, as the **a**-unate case follows by considering the monotone function obtained by replacing $\mathbf{x}$ with $\mathbf{x} \oplus \mathbf{a}$. One direction of the theorem follows immediately from the definition of p-irredundancy.

Separating the negated and unnegated projections, let us write φ as

$$\varphi = \bigvee_{i \in I} x_i t_i \vee \bigvee_{j \in J} \bar{x}_j t'_j.$$

As $\varphi(\mathbf{1}) = 1$, it must be the case that I is nonempty. Also, as $\mathbf{1}$ is contained in every unnegated projection, projectivity implies that it satisfies every term $x_i t_i$ with $i \in I$, thus it must be the case that every t_i is monotone. We claim that

$$T(\varphi) \subseteq T(t_i) \tag{3}$$

for every $i \in I$. Indeed, if $\varphi(\mathbf{x}) = 1$, then by monotonicity $\varphi(\mathbf{x}^{x_i=1}) = 1$, by projectivity $t_i(\mathbf{x}^{x_i=1}) = 1$, and by *b)* of p-irredundancy $t_i(\mathbf{x}) = 1$. (Here $\mathbf{x}^{x_i=1}$ is $\mathbf{x}$ with its first component fixed to 1.)

Now consider any two terms $x_i t_i$ and $x_j t_j$ from φ. From (3) we get $T(x_i t_i) \subseteq T(\varphi) \subseteq T(t_j)$ and $T(x_j t_j) \subseteq T(\varphi) \subseteq T(t_i)$. Thus

$$Lit(t_j) \subseteq Lit(x_i t_i) \quad \text{and} \quad Lit(t_i) \subseteq Lit(x_j t_j). \tag{4}$$

However, $x_j \notin Lit(x_i t_i)$ and $x_i \notin Lit(x_j t_j)$, as otherwise using (4) it follows that φ is p-redundant. But then $Lit(t_j) = Lit(t_i)$. Thus so far we know that φ is of the form

$$\varphi = \bigvee_{i \in I} x_i t \vee \bigvee_{j \in J} \bar{x}_j t'_j,$$

where $I \neq \emptyset$. If $J = \emptyset$, we are done. Otherwise let us consider a term $\bar{x}_j t'_j$ with $j \in J$. By projectivity, $T(\bar{x}_j t'_j) = T(\bar{x}_j \varphi)$, and monotonicity implies that t'_j is monotone. It follows from (3) that $T(\bar{x}_j t'_j) \subseteq T(\varphi) \subseteq T(t)$, thus $Lit(t) \subseteq Lit(\bar{x}_j t'_j)$, and so $Lit(t) \subseteq Lit(t'_j)$. Thus φ can further be written as

$$\varphi = \bigvee_{i \in I} x_i t \vee \bigvee_{j \in J} \bar{x}_j t t''_j,$$

where now $I, J \neq \emptyset$ and t, t''_j are monotone terms. If $|J| = 1$ and $I = J = \{i\}$ for some i, then we are done. Otherwise, there are terms $x_i t$ and $\bar{x}_j t t''_j$ in φ such that $i \neq j$. Now $T(\bar{x}_j x_i t) = T(x_i \bar{x}_j t t''_j) \neq \emptyset$, and so either $t''_j = x_i$ or t''_j is the empty term. But $t''_j = x_i$ would imply that φ is p-redundant, thus t''_j is the empty term. Hence $T(\bar{x}_j t) \subseteq T(\varphi)$, and by monotonicity $T(t) \subseteq T(\varphi)$. With (3) this implies $T(t) = T(\varphi)$. But then for every other term $\bar{x}_k t t''_k$ of φ it holds that $T(\bar{x}_k t t''_k) = T(\bar{x}_k \varphi) = T(\bar{x}_k t)$, and so t''_k is the empty term. Therefore

$$t \equiv \varphi = \bigvee_{i \in I} x_i t \vee \bigvee_{j \in J} \bar{x}_j t \equiv \left(\bigvee_{i \in I} x_i \vee \bigvee_{j \in J} \bar{x}_j \right) t.$$

This can only hold if some variable occurs both in I and J, contradicting condition *c)* of the p-irredundancy of φ. □

The example of (2) shows that the representation as a p-irredundant 1-PDNF is not always unique. Also, it is an interesting consequence of the theorem that there are 1-projective monotone functions, which cannot be expressed with monotone 1-projective DNF. Consider, for example the 1-PDNF

$$(x \cdot 1) \vee (\bar{x} \cdot yz),$$

representing the monotone function $x \vee yz$. If there is an equivalent monotone 1-PDNF, then it can be transformed into a monotone p-irredundant 1-PDNF, which must look like the first case in the theorem. But then the minimal true vectors must have Hamming distance at most 2, which is not the case for this function.

Theorem 15 *A formula φ is a p-irredundant 1-PDNF formula if and only if it is either of the form*

$$\varphi = \bigvee_{i=1}^{s} (\rho^i_1 t_i \vee \cdots \vee \rho^i_{\ell_i} t_i),$$

where $\rho^i_u \notin Lit(t_i)$ and $\bar{\rho}^j_u \in Lit(t_i)$ for every $i \neq j$ and $1 \leq u \leq \ell_j$, or it is of the form

$$\varphi = xt \vee \bar{x}t' \ ,$$

where $x \notin Lit(t_i)$ and $\bar{x} \notin Lit(t')$.

Proof. Again one direction of the theorem is immediate from the definition of p-irredundant. For the other direction, if there are two complementary projections in φ, then by condition *c)* of p-irredundancy, it must be of the form $xt \vee \bar{x}t'$. Otherwise, let us assume that φ is of the form $\varphi = \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell$. Consider any two terms $\rho_i t_i$ and $\rho_j t_j$. If $T(\rho_i t_i) \cap T(\rho_j t_j) \neq \emptyset$, then $\rho_i t_i \vee \rho_j t_j$ is unate, and by Theorem 14 it must be the case that $t_i = t_j$. On the other hand, if $T(\rho_i t_i) \cap T(\rho_j t_j) = \emptyset$, then by projectivity, it holds that $T(\rho_i \rho_j t_j) = \emptyset$, thus $\bar{\rho}_i \in Lit(t_j)$. Thus for every term $\rho_i t_i$, those terms $\rho_j t_j$ for which $T(\rho_i t_i) \cap T(\rho_j t_j) \neq \emptyset$ have the same conjunction part, and all the other terms contain $\bar{\rho}_i$ in their conjunction part. □

6 Revising disjunctions and *k*-PDNF

In this section we consider two different revision algorithms. First we define the *revision distance* between two k-PDNF's, which is used in our complexity measure.

The distance of two terms t and t^* is $|t \oplus t^*|$, the number of literals occurring in exactly one of the two terms. The revision distance between an *initial* formula

$$\varphi_0 = \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell \vee \rho_{\ell+1} t_{\ell+1} \vee \cdots \vee \rho_{\ell+d} t_{\ell+d}$$

and a *target* formula $\varphi = \rho_1 t_1^* \vee \cdots \vee \rho_\ell t_\ell^* \vee \rho'_1 t'_1 \vee \cdots \vee \rho'_a t'_a$ is

$$dist(\varphi_0, \varphi) = d + \sum_{i=1}^{\ell} |t_i \oplus t_i^*| + \sum_{i=1}^{a} \max(|t'_i|, 1).$$

The distance is *not* symmetric, and this reflects the fact that we are interested in the number of edit operations required to transform φ_0 to φ. These edit operations are the deletion of a literal or a term, and the addition of a literal. For example, the d term in the definition of *dist* corresponds to the deletion of the d terms $\rho_{\ell+1} t_{\ell+1}, \cdots, \rho_{\ell+d} t_{\ell+d}$.

Given an initial formula φ_0 and a target formula φ, we want our mistake bound to be polynomial in the revision distance $e = dist(\varphi_0, \varphi)$, and logarithmic (or polylogarithmic) in all other parameters. In this case, that means logarithmic in n and, for k-PDNF, in the total number of projections of size k.

We begin by demonstrating that the original Winnow, with appropriately modified initial weights, is an efficient revision algorithm—even in the presence of attribute errors, if we are willing to tolerate mistakes polynomial in the number of attribute errors as well as the usual parameters. (Previous work on theory revision has not given much consideration to noise.) We will use this result to show how to use an algorithm similar to Valiant's PDNF learning algorithm to revise PDNF. There, two levels of Winnow are run, and even with noise-free data, mistakes made by the lower-level Winnows will represent attribute errors in the input to the top-level Winnow.

Throughout this section, we will sometimes need to discuss both the components of vectors and which round of a mistake-bounded algorithm a vector is used in. When we need to discuss both, we will write $\mathbf{v}_{r,i}$ to denote component i of the value that vector $\mathbf{v}$ takes on in round r.

6.1 Revising disjunctions

Consider a version of Winnow, which we call RevWinn, presented as Algorithm 1. Note that this is Littlestone's Winnow2 [10] using different initial weights, with his parameters set to $\alpha = 2$, and $\theta = n/2$ (except that we have divided all the weights by n, because we feel it makes the analysis below a little easier to follow).

Note that throughout, all of the weights are always strictly between 0 and 1.

Algorithm 1 Algorithm RevWinn(φ_0)

```
Initialization: For $i = 1, \dots, n$ initialize the weights to
    $w_{0,i} = \begin{cases} 1 & \text{if variable } x_i \text{ appears in } \varphi_0 \\ \frac{1}{2n} & \text{otherwise} \end{cases}$
The hypothesis function in round $r$ is
    $h_r(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{w}_{r-1} \cdot \mathbf{x} < 1/2 \\ 1 & \text{otherwise} \end{cases}$
In round $r$ predict $\hat{y}_r = h_r(\mathbf{x}_r)$
If $\hat{y}_r \neq y_r$, then update the weights for $i = 1, \dots, n$ to
    $w_{r,i} = w_{r-1,i} \cdot 2^{\mathbf{x}_r(y_r - \hat{y}_r)}$
```

Theorem 16 *The number of mistakes made by Algorithm RevWinn with initial (monotone) disjunction $\varphi_{\mathbf{0}}$ and target (monotone) disjunction φ is*

$$O(\#\mathit{AttributeErrors} + e \log n) \enspace ,$$

where $e = dist(\phi_0, \phi)$.

Proof. Consider any run of the algorithm of length R. Let I be the set of indices i of variables that appear in both the initial and target disjunctions, such that for at least one round r variable $\mathbf{x}_{r,i} = 1$ but $y_r = 0$. Let J be the set of indices of variables that appear in the target disjunction but not in the initial disjunction. Let us also introduce the notation $\overline{I \cup J} = \{1, \dots, n\} \setminus (I \cup J)$. When no confusion arises, we will sometimes refer to a variable x_i belonging to one of these sets when we really should say that the variable's index belongs to the set.

We will use later the fact that any variable in both φ_0 and φ that is *not* in I never has its weight changed from 1.

For the proof we use a potential function $\Phi(\mathbf{w})$ that is somewhat different from those used in some other cases for analyzing Winnow (e.g., in [2, 9]). Put $\Phi(\mathbf{w}) = \sum_{i=1}^{n} \Phi_i(\mathbf{w})$, where

$$\Phi_i(\mathbf{w}) = \begin{cases} w_i - 1 + \ln \frac{1}{w_i} & \text{if } i \in I \cup J \\ w_i & \text{otherwise.} \end{cases}$$

It can be verified that $\Phi_i(\mathbf{w}) \geq 0$ for any $\mathbf{w} \in (0, 1]^n$.

Let $\Delta_r = \Phi(\mathbf{w}_{r-1}) - \Phi(\mathbf{w}_r)$ denote the change of the potential function during round r. We will derive both upper and lower bounds on $\sum_{r=1}^{R} \Delta_r$ that will allow us to relate the number of mistakes made by RevWinn to e, n, and $\#\mathit{AttributeErrors}$.

First we derive an upper bound:

$$\sum_{r=1}^{R} \Delta_r = \Phi(\mathbf{w}_0) - \Phi(\mathbf{w}_R) \leq \Phi(\mathbf{w}_0) - \sum_{i \in \overline{I \cup J}} w_{R,i}$$

$$= \sum_{i \in I} \mathit{\Phi}_i(\mathbf{w}_0) + \sum_{j \in J} \mathit{\Phi}_j(\mathbf{w}_0) + \sum_{i \in \overline{I \cup J}} (w_{0,i} - w_{R,i}) \ . \tag{5}$$

For $i \in I$ we initialized $\mathbf{w}_{0,i} = 1$ so $\mathit{\Phi}_i(\mathbf{w}_{0,i}) = 0$. Also, $|J| \le e$, and $\mathit{\Phi}_j(\mathbf{w}_{0,j}) = \ln(2n) - (2n-1)/2n < \ln(2n)$ for $j \in J$, so the sum of the first two terms is at most $e \ln(2n)$. Now we need to bound the third term. The variables that appear neither in φ nor in φ_0 have initial weights $1/2n$, and so altogether can contribute at most $1/2$ to the sum. There are at most e variables in $\varphi_0 \setminus \varphi$, so those variables can contribute at most e to the sum. Finally, as noted earlier, the weights never change for those variables in both φ_0 and φ but not in I. Thus we get

$$\sum_{r=1}^{R} \mathit{\Delta}_r \le e \ln 2n + e + 1/2 \ . \tag{6}$$

To get a lower bound on the sum, we begin by deriving a lower bound on the change in potential in one round. Now

$$\begin{aligned}\mathit{\Delta}_r &= \sum_{i \in I \cup J} \left(w_{r-1,i} - w_{r,i} + \ln \frac{w_{r,i}}{w_{r-1,i}} \right) + \sum_{i \in \overline{I \cup J}} (w_{r-1,i} - w_{r,i}) \\ &= \sum_{i=1}^{n} (w_{r-1,i} - w_{r,i}) + \sum_{i \in I \cup J} \ln \frac{w_{r,i}}{w_{r-1,i}} \ .\end{aligned} \tag{7}$$

There are three cases for a round r: no change in weights, a demotion, and a promotion. Obviously, when no update is done in round r (i.e., $\hat{y}_r = y_r$), then $\mathit{\Delta}_r = 0$.

In a demotion step, $\hat{y}_r = 1$ and $y_r = 0$. By the definition of I and J, in this case $\mathit{AttrErr}(r) = |(I \cup J) \cap \mathbf{x}_r|$.[4] Also, the total weight of components being on in $\mathbf{x}_r$ is at least $1/2$, and the weight of each of those components is halved. So, using (7),

$$\mathit{\Delta}_r \ge 1/4 + |(I \cup J) \cap \mathbf{x}_r| \ln \left(\frac{1}{2} \right) = 1/4 - \ln 2 \mathit{AttrErr}(r) \ . \tag{8}$$

In a promotion step, $\hat{y}_r = 0$ and $y_r = 1$. We know that the components of $\mathbf{x}_r$ that are on have total weight at most $1/2$, and that each of these components is multiplied by 2. So the first term in (7) is at least $-1/2$. Thus $\mathit{\Delta}_r \ge -1/2 + |(I \cup J) \cap \mathbf{x}_r| \ln 2$. Now if $y_r = \varphi(\mathbf{x}_r)$, then $|(I \cup J) \cap \mathbf{x}_r| \ge 1$, because we know that $\hat{y}_r = 0$ and we know that all the weights of variables in φ but not in I are 1. If $y_r \ne \varphi(\mathbf{x}_r)$, then $\mathit{AttrErr}(r) = 1$. Thus, in a promotion step, it always holds that

$$\mathit{\Delta}_r \ge -1/2 + \ln 2 (1 - \mathit{AttrErr}(r)) \ . \tag{9}$$

[4] With mild abuse of notation, we write $S \cap \mathbf{x}_r$ to denote the set of indices that are both in the set S and set to 1 in the vector $\mathbf{x}_r$.

Finally, let M^- denote the total number of demotions and M^+ the total number of promotions. Then (8) and (9) give us

$$\begin{aligned}\sum_{r=1}^{R} \Delta_r &\geq \sum_{\hat{y}_r=1, y_r=0} (1/4 - \ln 2 \mathit{AttrErr}(r)) \\ &\quad + \sum_{\hat{y}_r=0, y_r=1} (\ln 2 - 1/2) - \ln 2\, \mathit{AttrErr}(r) \\ &= M^-/4 + (\ln 2 - 1/2)M^+ - \ln 2 \# \mathit{AttributeErrors} \ .\end{aligned}$$

Combining this with (6) gives the desired mistake bound. □

Note that the potential function used depends on the particular run of the algorithm, and that it does not appear to be any obvious measure of distance between the actual weight vector $\mathbf{w}_r$ and a weight vector for the target.

Remark 17 *Using the De Morgan rules one can easily modify the code of Algorithm RevWinn to make it revise conjunctions instead of disjunctions, and have the same mistake bound. Call the resulting algorithm RevWinnC.*

6.2 Revising *k*-PDNF

Now we give a revision algorithm for PDNFs. We use Valiant's two-level algorithm [14] for learning PDNFs, except that we use the different initial weights in the individual copies of Winnow that were discussed in the previous subsection. We present this as Algorithm Rev-k-PDNF (see Algorithm 2). Rev-k-PDNF consists of a top-level RevWinn algorithm that handles the selection of the appropriate projections. On the lower level, instances of RevWinnC are run, one for each projection, to find the appropriate term for that particular projection. Each instance of RevWinnC maintains its own separate hypothesis h^ρ for one of the $2^k\binom{n}{k}$ projections ρ. We will write this as h_r^ρ when we need to indicate the current hypothesis in a particular round r.

For each projection ρ, introduce a new Boolean variable v_ρ. We denote by $\mathbf{v}$ the vector formed by all these variables. The top level RevWinn learns a disjunction over these variables; its hypothesis in round r is denoted by h_r. In round r, we define variable

$$v_{r,\rho} = \rho(\mathbf{x}_r) h_r^\rho(\mathbf{x}_r) \ .$$

Algorithm Rev-k-PDNF predicts $h_r(\mathbf{v}_r)$ in round r.

Theorem 18 *Suppose that the initial and target formulas are, respectively, the k-PDNF$_n$ formulas*

$$\begin{aligned}\varphi_0 &= \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell \vee \rho_{\ell+1} t_{\ell+1} \vee \cdots \vee \rho_{\ell+d} t_{\ell+d} \ , \\ \varphi &= \rho_1 t_1^* \vee \cdots \vee \rho_\ell t_\ell^* \vee \rho_1' t_1' \vee \cdots \vee \rho_a' t_a' \ ,\end{aligned}$$

and $e = dist(\varphi_0, \varphi)$. Then algorithm Rev-$k$-PDNF makes $O(ek \log n)$ mistakes.

Algorithm 2 The procedure Rev-k-PDNF(φ_0)

```
1: {$\varphi_0 = \rho_1 t_1 \vee \cdots \vee \rho_{\ell+d} t_{\ell+d}$ is the $k$-PDNF to be revised to another $k$-PDNF}
2: Initialize a RevWinn instance for the top level algorithm with initial disjunction
   $v_{\rho_1} \vee \cdots \vee v_{\rho_{\ell+d}}$.
3: Initialize a RevWinnC instance for each $k$-projection $\rho_i$ with parameter $t_i$ for
   $\rho_1, \ldots, \rho_{\ell+d}$ respectively and with parameter the empty conjunction for the rest.
4: for each round $r$ with instance $\mathbf{x}_r$ do
5:    Set each $v_{r,\rho} = \rho(\mathbf{x}_r) h_r^{\rho}(\mathbf{x}_r)$
6:    Predict $\hat{y}_r = h_r(\mathbf{v}_r)$
7:    if $\hat{y}_r \neq y_r$ then
8:       Update top-level RevWinn for a $\hat{y}_r \neq y_r$ mistake on $\mathbf{v}_r$.
9:       for each $\rho$ with $\rho(\mathbf{x}_r) = 1$ and $v_{r,\rho} \neq y_r$ do
10:         Update the low-level RevWinnC instance $\rho$ for a $v_{r,\rho} \neq y_r$ mistake on $\mathbf{x}_r$.
```

Proof. The top-level RevWinn is revising a disjunction over the v_ρ's. There will be two sources of mistakes. First, the initial disjunction is not correct; it needs revising. Second, the values of the v_ρ's will sometimes be erroneous, because the low-level RevWinnC's are imperfect. (The actual $\mathbf{x}$ input and y classification are assumed to be noiseless.)

Theorem 16 tells us how to calculate the overall number of mistakes of the top-level RevWinn as a function of three quantities: the revision distance, which is $d + a$, the total number of variables, both relevant and irrelevant for the disjunction, which is $2^k\binom{n}{k}$, and the total number of attribute errors, which we will now calculate.

In fact, we will *not* count all the attribute errors. We will count (actually provide an upper bound on) only those attribute errors that occur when RevWinn is charged with a mistake.

For $i = 1, \ldots, \ell$, the RevWinnC instance corresponding to projection ρ_i predicts v_{ρ_i} according to h^{ρ_i}. That RevWinnC instance updates for a mistake only when the overall algorithm makes a mistake (i.e., $\hat{y}_r \neq y_r$), its prediction was different from y_r, and $\rho_i(\mathbf{x}_r) = 1$. Now $y_r = \varphi(\mathbf{x}_r) = t_i^*(\mathbf{x}_r)$ (the last equation holds because of projectivity and because $\rho_i(\mathbf{x}_r) = 1$). This means that the mistake bound for this RevWinnC tells us how many times this RevWinnC can make errors on rounds when the overall algorithm makes an error; after that number of mistakes, this RevWinnC will then always predict correctly. By Remark 17 the mistake bound on this RevWinnC is $O(|t_i \oplus t_i^*| \ln n)$.

For $j = 1, \ldots, a$ a similar argument shows that there are at most $O(|t'_j| \ln n)$ rounds r where $v_{r,\rho'_j} \neq \rho'_j(\mathbf{x}_r) t'_j(\mathbf{x}_r)$ and the top-level RevWinn makes a mistake. Put $F = (\sum_{i=1}^{\ell} |t_i \oplus t_i^*| + \sum_{j=1}^{a} |t'_j|) \ln n$.

How many times can Rev-k-PDNF err when predicting? We just argued that the total number of attribute errors that occur when the top-level RevWinn makes a mistake is $O(F)$. The total number of variables that the top-level RevWinn is working with is $2^k\binom{n}{k}$. Thus, the overall mistake bound is, by Theorem 16, $O(F + (d+a)\log(2^k\binom{n}{k})) = O(ek \log n)$, since $F = O(e \log n)$. $\square$

Remark: For learning an m-term k-PDNF$_n$ from scratch, that is, for revising the empty k-PDNF$_n$ to a target k-PDNF$_n$, this algorithm has the same asymptotic mistake bound as Valiant's learning algorithm [14]: $O(kms \log n)$, where s is the maximum number of variables in any term in the target.

References

1. Dana Angluin. Queries revisited. In *Algorithmic Learning Theory, 12th International Conference, ALT 2001, Washington, DC, USA, November 25–28, 2001, Proceedings*, volume 2225 of *Lecture Notes in Artificial Intelligence*, pages 12–31. Springer, 2001.
2. Peter Auer and Manfred K. Warmuth. Tracking the best disjunction. *Machine Learning*, 32(2):127–150, 1998. Earlier version in 36th FOCS, 1995.
3. Avrim Blum. On-line algorithms in machine learning. Available from `http://www-2.cs.cmu.edu/~avrim/Papers/pubs.html`, 1996.
4. Judy Goldsmith, Robert H. Sloan, B. Szörényi, and György Turán. Theory revision with queries: Horn, read-once, and parity formulas. Technical Report TR03-039, Electronic Colloquium on Computational Complexity (ECCC), 2003. Available at `http://www.eccc.uni-trier.de/eccc/`. Also submitted for journal publication.
5. Judy Goldsmith, Robert H. Sloan, and György Turán. Theory revision with queries: DNF formulas. *Machine Learning*, 47(2/3):257–295, 2002.
6. Tibor Hegedüs. Generalized teaching dimensions and the query complexity of learning. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 108–117. ACM Press, New York, NY, 1995.
7. Lisa Hellerstein, Krishnan Pillaipakkamnatt, Vijay Raghavan, and Dawn Wilkins. How many queries are needed to learn? *J. ACM*, 43(5):840–862, 1996.
8. David Helmbold, Robert Sloan, and Manfred K. Warmuth. Learning nested differences of intersection closed concept classes. *Machine Learning*, 5(2):165–196, 1990. Special Issue on Computational Learning Theory; first appeared in 2nd COLT conference (1989).
9. Jyrki Kivinen and Manfred K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Proc. 27th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, New York, NY, 1995.
10. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
11. N. Littlestone. A mistake-bound version of Rivest's decision-list algorithm. Personal communication to Avrim Blum, 1989.
12. Chris Mesterharm. Tracking linear-threshold concepts with Winnow. In *15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 2002, Proceedings*, volume 2375 of *Lecture Notes in Artificial Intelligence*, pages 138–152. Springer, 2002.
13. Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.
14. Leslie G. Valiant. Projection learning. *Machine Learning*, 37(2):115–130, 1999.
15. Leslie G. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 47(5):854–882, 2000.
16. S. Wrobel. *Concept Formation and Knowledge Revision.* Kluwer, 1994.
17. S. Wrobel. First order theory refinement. In L. De Raedt, editor, *Advances in ILP*, pages 14–33. IOS Press, Amsterdam, 1995.