Projective DNF Formulae and Their Revision ⋆

Robert H. Sloan [a,1] Balázs Szörényi [b,2] György Turán [c,b,1,3]

[a] *Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053, USA*

[b] *Research Group on Artificial Intelligence, Hungarian Academy of Sciences and University of Szeged, Szeged, Hungary-6720*

[c] *Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7045, USA*

Abstract

Combinatorial and learnability results are proven for projective disjunctive normal forms, a class of DNF expressions introduced by Valiant.

Dedicated to Prof. Peter Hammer on the occasion of his 70th birthday.

1 Introduction

The model of *projection learning* was introduced by Valiant [17], motivated by constraints imposed on learnability by biology. Projection learning aims to learn a target concept over some large domain, in this paper $\{0,1\}^n$, by learning some of its projections (or restrictions) to a class of smaller domains, and combining these projections. Valiant proved a general mistake bound for

⋆ This paper represents an extended and revised version of the conference paper [16].

Email addresses: sloan@uic.edu (Robert H. Sloan), szorenyi@inf.u-szeged.hu (Balázs Szörényi), gyt@uic.edu (György Turán).

URL: www.cs.uic.edu/~sloan (Robert H. Sloan).

[1] Research supported in part by by the National Science Foundation under grants CCR-0100336 and CCF-0431059.

[2] Part of this work was done while Szörényi was visiting at University of Illinois at Chicago.

[3] Research supported in part by the Hungarian National Science Foundation under grant OTKA T-25721.

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. The class of projective DNF expressions does not appear to have been studied at all before Valiant's work. As the learnability of DNF is a major open problem in computational learning theory [4], it is of interest to those who study computational learning theory to identify new learnable subclasses and to understand their scope.

In this paper we discuss various combinatorial and learnability properties of projective DNF. 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 decision lists. We also present lower and upper bounds for the *exclusion dimension* of projective DNF. The exclusion dimension, or certificate size [2, 7, 8], 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 [11]. His Winnow algorithm learns a (monotone) disjunction of k variables out of a total of n variables with $O(k \log n)$ mistakes. Thus Winnow is very efficient for learning problems where there are many attributes but most of them are irrelevant. Valiant argues that learning in the brain, and also many real-life applications, has exactly that property [17, 18]. 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 both Winnow and Valiant's projection learning algorithm

[4] While this article was under review, Alekhnovich et al. showed that DNF is not properly PAC learnable in polynomial time unless NP = RP [1], providing further motivation to find positive learnability results.

is their attribute efficiency.

The seemingly 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 deletions or additions of literals. 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 gives a general survey of the theory revision literature [19, 20]; efficient revision in the framework of learning with queries is discussed in detail in [5, 6].

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 simple [5]. 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; this approach differs from that of Valiant.

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., [3, 13]). 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 [3].

After some preliminaries given in Section 2, Section 3 contains the definition of projective DNF and a discussion of some of their properties. Section 4 contains the bounds for the exclusion dimension. The characterization of 1-PDNF is presented in Section 5. Section 6 presents the formal model of revision and the revision algorithms. Finally, we mention some open problems and make further remarks in Section 7.

2 Preliminaries

We use various standard terms from propositional logic, such as variable, conjunction, and disjunction. We call elements of $\{0,1\}^n$ *vectors*, and denote specific vectors as strings over $\{0,1\}$, sometimes using exponential notation. Thus for the vector $(0,0,0,1,0) \in \{0,1\}^5$ we will write either 00010 or $0^3 10$. We also use notation such as $\mathbf{x}^{x_i=1}$ to denote vector $\mathbf{x} \in \{0,1\}^n$ with its i'th component fixed to 1. We always use n for the number of propositional variables; all vectors will be from $\{0,1\}^n$ unless we specifically state otherwise.

The weight of vector $\mathbf{x}$ 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 ($[n] = \{1,\ldots,n\}$). With a slight abuse of notation we also use $\mathbf{e}_i^n$ (resp. $\mathbf{g}_i^n$) to denote $\mathbf{e}_{\{i\}}^n$ (resp. $\mathbf{g}_{\{i\}}^n$). The all 1's vector 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 the set of its true vectors. 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. $\mathbf{x} \vee \mathbf{y}$ and $\mathbf{x} \oplus \mathbf{y}$) for $(x_1 \wedge y_1,\ldots,x_n \wedge y_n)$ (resp. $(x_1 \vee y_1,\ldots,x_n \vee y_n)$ and $(x_1 \oplus y_1,\ldots,x_n \oplus y_n)$). We call $\mathbf{x} \wedge \mathbf{y}$ the *meet*, and $\mathbf{x} \vee \mathbf{y}$ the *join* of $\mathbf{x}$ and $\mathbf{y}$.

A literal is an unnegated or negated variable. Negation of a variable x is denoted by $\bar{x}$. Unnegated variables are called positive literals; negated variables negative literals. A conjunction of literals is also called a term. The empty conjunction (denoted by $\top$) is always true. For a term t, we write $Lit(t)$ for the set of literals in t. A k-conjunction is a conjunction of k literals. A disjunctive normal form (DNF) is a disjunction of terms. A k-DNF is a DNF such that each of its terms contains at most k literals. A K-term DNF is a DNF with at most K terms.

A decision list [14] is an ordered list of pairs $L = (c_1,b_1),\ldots,(c_m,b_m)$, where $c_1,\ldots,c_{m-1}$ are terms, $c_m = \top$, and $b_1,\ldots,b_m \in \{0,1\}$. Decision list L evaluates to b_ℓ on vector $\mathbf{x} \in \{0,1\}^n$ if $c_1,\ldots,c_{\ell-1}$ are false and c_ℓ is true on $\mathbf{x}$. A

decision list is a k-decision list if all its terms have size at most k.

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 $g(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{a})$ is monotone, where $\mathbf{a} \in \{0,1\}^n$, and it is unate if it is $\mathbf{a}$-unate for some $\mathbf{a} \in \{0,1\}^n$. A term is monotone if it consists of unnegated variables. Given $\mathbf{a} \in \{0,1\}^n$, a term is $\mathbf{a}$-unate if the sign of every literal in it agrees with $\mathbf{a}$—that is, a literal is positive iff the corresponding component of $\mathbf{a}$ is 0. For example, if $n = 3$ and $\mathbf{a} = 101$ then $\bar{x}_1 x_2$ is $\mathbf{a}$-unate.

We use the standard model of *mistake bounded* learning [11], 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, that is, sequences of instances.

In addition to the standard mistake-bounded model, as a technical tool for the learning result, we also consider a model of learning in the presence of noise. In the model of learning monotone disjunctions with *attribute errors* (Auer and Warmuth [3], also used by Valiant [17] with a different name) it may happen that y is *not* the correct classification of $\mathbf{x}$. It is assumed that the error comes from some components (or attributes) 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 classification y_r is not the correct classification of $\mathbf{x}_r$, then, if $y_r = 1$ then $\textsc{AttrErr}(r) = 1$ (as it is enough to switch one bit on to satisfy a disjunction), and if $y_r = 0$ then $\textsc{AttrErr}(r)$ is the number of variables that 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 $\#\textsc{AttributeErrors}$, is the sum of the attribute errors of the rounds. This notion is used only for technical purposes: it plays an important role inside some proof, but does not appear in any results.

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, where $t_1 \not\equiv 0$, it holds that $t_1 \to t_2$ iff $T(t_1) \subseteq T(t_2)$ iff t_1 is subsumed by t_2 (i.e., $Lit(t_1) \supseteq Lit(t_2)$).

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 $\mathbf{z} \in A$.*

PROOF. The $\Rightarrow$ direction is easy to see.

The $\Leftarrow$ direction follows by noting that the condition implies that the $\wedge$ and the $\vee$ of all the vectors in A is in A, and every vector between these two vectors

is also in A. The conjunction of those literals to which value 1 is assigned by both of the above vectors is a term that is satisfied by exactly the vectors in A. $\Box$

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*

$$\varphi = \rho_1 c_1 \vee \cdots \vee \rho_\ell c_\ell \ , \tag{1}$$

where every ρ_i is a k-conjunction, every c_i is a conjunction and for every i it holds that

$$\rho_i \varphi \equiv \rho_i c_i. \tag{2}$$

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

The k-conjunctions ρ_i are also called *k-projections*, or, when k is clear from context, simply *projections*. Conditions (1) and (2) mean that when restricted to the subcube $T(\rho_i)$, the formula φ is equivalent to the conjunction c_i, and every true point of φ arises this way for some restriction. This corresponds to the intuition, described in the Introduction, that the restrictions to a prespecified set of simple domains are simple, and the whole function can be patched together from these restrictions.

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,

$$(x \cdot y) \vee (z \cdot y) \quad \text{and} \quad (x \cdot y) \vee (\bar{x} \cdot yz) \tag{3}$$

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. The requirement (2) is equivalent to requiring that

$$\rho_j \rho_i c_i \equiv \rho_i \rho_j c_j \tag{4}$$

for every i and j. This makes it easy to verify that a given expression, such as those in (3), 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.

If a function is k-projective, then it is k'-projective for every k' with $k \leq k' \leq n$. Note that the complete DNF (consisting of n-conjunctions corresponding to the true points of f) shows that every n-variable function is n-projective.

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 k-DNF, resp. K-term DNF. Let k-DL$_n$ denote the class of k-decision lists on n variables.

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

$$k\text{-DNF}_n \subset k\text{-PDNF}_n \subset K\text{-term-DNF}_n \ .$$

PROOF. Every k-DNF can be expanded into a DNF with every term containing exactly k literals, and such a DNF can be viewed as a k-PDNF: the terms are the projections themselves. The number of k-conjunctions over a given set of n variables is $2^k \binom{n}{k}$; thus every k-PDNF over n variables is a DNF with at most K terms. The properness of the first inclusion follows by noting that $x_1 \wedge \cdots \wedge x_n$ is a k-PDNF for every $k \geq 1$. The second proper inclusion follows from Proposition 8 below. □

Proposition 4 *For every $k < n$ it holds that k-PDNF$_n \subseteq (k+1)$-DL$_n$. There is a constant $0 < \alpha < 1$ such that the inclusion is proper if n is sufficiently large and $k \leq \alpha n$.*

PROOF. Let $\varphi = \rho_1 c_1 \vee \cdots \vee \rho_\ell c_\ell$ be a k-PDNF$_n$ formula. We construct a $(k+1)$-decision list for φ as follows. For each i in order, we put $|c_i| + 1$ entries into the list. The first $|c_i|$ entries consist of the conjunction of ρ_i and the negation of one literal of c_i, with truth value 0. The last is just the conjunction ρ_i with truth value 1. This corresponds to the fact that when ρ_i is satisfied, φ evaluates to 0 if and only if c_i is not satisfied. Finally, after all ℓ terms are handled, we have the default truth value 0, because φ can only be satisfied by satisfying some term $\rho_i c_i$.

For the proper inclusion consider $\bigwedge_{i=1}^{\lfloor \frac{n}{2} \rfloor}(x_i \vee y_i)$. This is a 2-CNF expression, and thus it is a $(k+1)$-DL for every k: $(\bar{x}_1\bar{y}_1, 0), \ldots, (\bar{x}_{\lfloor \frac{n}{2} \rfloor}\bar{y}_{\lfloor \frac{n}{2} \rfloor}, 0), (\top, 1)$. On the other hand, it is an easily seen and often used fact that every equivalent DNF has at least $2^{\lfloor \frac{n}{2} \rfloor}$ terms. Thus there cannot be an equivalent k-PDNF expression for any k satisfying $2^k\binom{n}{k} < 2^{\lfloor \frac{n}{2} \rfloor}$. The proper inclusion then follows by standard calculation. $\Box$

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 DNFs. The exponent of the lower bound matches the exponent of the upper bound in order of magnitude for every fixed k.

Proposition 5 *The following bounds hold for* $|k\text{-PDNF}_n|$*:*

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

PROOF. For the lower bound, let $\ell < n$ be fixed. For any k-element subset $I \subseteq [\ell]$ consider the k-conjunction $\rho_I = \bigwedge_{i\in I} x_i$ and the term $t_I^* = \bigwedge_{i\in[\ell]\setminus I} \bar{x}_i$. Form the expression

$$\bigvee_{I\subseteq[\ell],|I|=k} \rho_I\, t_I^*\, t_I \ ,$$

where the t_I's are arbitrary conjunctions of some literals from $\{x_{\ell+1}, \ldots, x_n\}$. Different choices of the t_I's represent different k-projective functions. Thus the number of k-projective functions is at least $(3^{n-\ell})^{\binom{\ell}{k}}$: in each of the $\binom{\ell}{k}$ terms variables $x_{\ell+1}, \ldots, x_n$ can be negated, unnegated or missing. The bound follows by choosing $\ell = \left\lceil \frac{k}{k+1}n \right\rceil$. $\Box$

4 Exclusion dimension

We present the definitions of specifying sets and the exclusion dimension, following the terminology of Angluin [2]. (With minor variations, exclusion dimension is called unique specification dimension by Hegedüs [7] and certificate size by Hellerstein et al. [8].)

Let f be an n-variable Boolean function. A set $A \subseteq \{0,1\}^n$ is a *specifying set* of f with respect to a class $\mathcal{C}$ of Boolean functions if there is at most one

function in $\mathcal{C}$ that agrees with f on A. (So clearly $\{0,1\}^n$ is always a specifying set.) The *specifying set size* of f with respect to $\mathcal{C}$ is

$$spec_{\mathcal{C}}(f) = \min\{|A| : A \text{ is a specifying set for } f \text{ with respect to } \mathcal{C}\},$$

and the *exclusion dimension* of the class $\mathcal{C}$ is

$$XD(\mathcal{C}) = \max\{spec_{\mathcal{C}}(f) : f \notin \mathcal{C}\}.$$

A specifying set A for $f \notin \mathcal{C}$ such that *no* function in $\mathcal{C}$ agrees with f on A is also called a *certificate of exclusion* (or simply *certificate*) for f with respect to $\mathcal{C}$. In our constructions below, we will usually give certificates of exclusion, which clearly give upper bound for the specifying set size.

For the rest of this article specifying sets are always with respect to k-PDNF, so we write $spec(f)$, omitting the subscript $\mathcal{C}$.

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 $spec(f) \geq |T(f)|-1$.*

PROOF. Suppose $|A| \leq |T(f)|-2$ for some $A \subseteq \{0,1\}^n$. Let $x, y \in T(f) \setminus A$ be two different vectors. As f is minimally non-k-projective, there is $g_x \in k\text{-PDNF}_n$ (resp. $g_y \in k\text{-PDNF}_n$) such that $T(g_x) = (A \cap T(f)) \cup \{x\}$ (resp. $T(g_y) = (A \cap T(f)) \cup \{y\}$). Now g_x and g_y are different elements of $k\text{-PDNF}_n$ that agree with f on A, thus A is not a specifying set for f. $\square$

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. We show only *(a)*, as *(b)* follows directly from *(a)*. If f is k-projective then it can be written as $\varphi = \rho_1 c_1 \vee \cdots \vee \rho_\ell c_\ell$. Consider an $\mathbf{x} \in T(f)$. Then $\rho_i c_i(\mathbf{x}) = 1$ for some i, thus $\mathbf{x} \in T(\rho_i)$. The definition of PDNF implies that $T(f) \cap T(\rho_i) = T(\rho_i c_i)$, which is a cube.

For the other direction, let us assume that for every $\mathbf{x} \in T(f)$ there is a k-projection $\rho_{\mathbf{x}}$ such that $\mathbf{x} \in T(\rho_{\mathbf{x}})$ and $T(f) \cap T(\rho_{\mathbf{x}}) = Q_{\mathbf{x}}$ is a cube. Then $Q_{\mathbf{x}}$ can be written as $T(\rho_{\mathbf{x}} c_{\mathbf{x}})$ for some conjunction $c_{\mathbf{x}}$, and f can be written as the k-PDNF expression $\bigvee_{\mathbf{x} \in T(f)} \rho_{\mathbf{x}} c_{\mathbf{x}}$. $\square$

We illustrate Lemma 7 with the following example. We claim that the function $f(x_1, x_2, x_3, x_4) = x_1 x_2 \vee x_3 x_4$ is not 1-projective. Call a vector that violates condition *(a)* in the lemma *k-deviant*, or simply *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 1101 and 1011, but it does not contain $1101 \wedge 1011 = 1001$. Another application is given by the following proposition, which was already referred to in Proposition 3.

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

PROOF. Let $T(f) = \{\mathbf{e}_i^n : 1 \leq i \leq k+2\} \cup \{\mathbf{0}\}$. Then $\mathbf{0}$ is k-deviant, as every k-conjunction ρ satisfied by $\mathbf{0}$ contains at least two $\mathbf{e}_i^n$'s, but $T(f) \cap T(\rho)$ does not contain the join of these two unit vectors, and thus it cannot be a cube according to Proposition 1. $\square$

The proposition gives a $(k+3)$-term DNF function which is not k-projective. As $k + 3 < 2^k \binom{n}{k}$, this gives the second separation in Proposition 3.

Theorem 9 *(a) For all n and k,*

$$XD(k\text{-PDNF}_n) \leq 3 \binom{n}{k} + 1 \ ,$$

and

(b) if $n \geq 4k(k+1)$, then

$$XD(k\text{-PDNF}_n) \geq \binom{\lfloor n/4 \rfloor}{k} - 1 \ .$$

PROOF. For the upper bound, we will calculate an upper bound on the size of a certificate of exclusion for any $f \notin k\text{-PDNF}_n$ with respect to $k\text{-PDNF}_n$.

To show that a a function f is not k-projective, it suffices to present a deviant vector $\mathbf{x}$ (i.e., $\mathbf{x}$ violates Condition (a) of Lemma 7) together with a certificate of $\mathbf{x}$'s deviance. For the certificate of $\mathbf{x}$'s deviance it suffices to specify, according to Proposition 1, for every k-conjunction ρ with $\rho(\mathbf{x}) = 1$, three vectors

$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ such that $\rho(\mathbf{x}_1) = \rho(\mathbf{x}_2) = \rho(\mathbf{x}_3) = 1$, $\mathbf{x}_1 \wedge \mathbf{x}_2 \leq \mathbf{x}_3 \leq \mathbf{x}_1 \vee \mathbf{x}_2$ and $f(\mathbf{x}_1) = f(\mathbf{x}_2) = 1$, $f(\mathbf{x}_3) = 0$. The number of k-conjunctions with $\rho(\mathbf{x}) = 1$ is $\binom{n}{k}$. Thus the upper bound follows: 1 for $\mathbf{x}$ itself, and then 3 vectors each for at worst all of the k-conjunctions.

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 $\mathbf{0}$ must be a negative literal, 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 nonsingular. Assume that $k(k+1) < 2^p$ and define the mapping*

$$h(\mathbf{b}_1, \ldots, \mathbf{b}_k) = (\mathbf{b}_1 \oplus A\mathbf{b}, \ldots, \mathbf{b}_k \oplus A\mathbf{b}),$$

where $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \{0,1\}^p$ and $\mathbf{b} = \mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_k$. Then it holds that

(a) *h is a bijection, and*
(b) *for every $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$ and $\mathbf{d}_1, \ldots, \mathbf{d}_k$ there is a $\mathbf{b}_k$ different from $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$, such that the components of $h(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ are all different from the $\mathbf{d}_i$'s.*

PROOF. If $h(\mathbf{b}_1, \ldots, \mathbf{b}_k) = (\mathbf{d}_1, \ldots, \mathbf{d}_k)$, then $\mathbf{d}_1 \oplus \cdots \oplus \mathbf{d}_k = \mathbf{b} \oplus (k \bmod 2)A\mathbf{b}$, which is equal to $\mathbf{b}$ (resp., $(A \oplus I)\mathbf{b}$), if k is even (resp., odd). Thus, knowing $\mathbf{d}_1, \ldots, \mathbf{d}_k$ we can first determine $\mathbf{b}$, and then we can determine every $\mathbf{b}_i$ by $\mathbf{b}_i = \mathbf{d}_i \oplus A\mathbf{b}$. Hence h is injective, and thus it is also bijective.

For *(b)*, note that a value for $\mathbf{b}_k$ can fail to satisfy the requirement only if it is either equal to one of the $\mathbf{b}_i$'s, or it makes $\mathbf{b}_i \oplus A\mathbf{b} = \mathbf{d}_j$ for some $1 \leq i, j \leq k$.

In each case we can solve for $\mathbf{b}_k$, thus there are altogether at most $k + k^2$ bad choices. Choosing any of the other $2^p - (k+k^2)$ vectors meets our requirements for $\mathbf{b}_k$. $\square$

Now we continue the proof of Theorem 9 with the general case $k > 1$. First, we need a matrix that fulfills the conditions of Lemma 10. It is easily verified that, for example, the matrix A with all 0's except $a_{1,1} = a_{p,1} = a_{i,i+1} = 1$ (where $i = 1, \ldots, p-1$) is such a matrix. It is clear from the definition of h that if the $\mathbf{b}_i$'s are all different, then the components of $h(\mathbf{b}_1, \ldots, \mathbf{b}_s)$ are also all different, and if we permute the $\mathbf{b}_i$'s then the components of the image are permuted in the same way. Thus if $I = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subseteq \{0,1\}^p$, 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 $h(\mathbf{b}_1, \ldots, \mathbf{b}_k)$.

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

We claim that $f_{n,k}$ is minimally non-k-projective. The argument for this is very similar to the argument in the special case above. The projection $\rho_I = \bigwedge_{i \in h(I)} x_{s+i}$ contains only $\mathbf{a}_I$ from $T(f_{n,k})$ by part *(a)* of Lemma 10, and if $\mathbf{a}_I$ is not contained in $T(f')$, then the projection $\rho_0 = \bigwedge_{i \in I} \bar{x}_i$ contains only $\mathbf{0}$ from $T(f')$. It only needs to be shown that $\mathbf{0}$ is deviant for $f_{n,k}$. Let ρ be any k-conjunction containing $\mathbf{0}$. We can assume that every literal $\bar{x}_i$ in ρ has $i \leq 2s$, as the other literals do not exclude any $\mathbf{a}_I$. We show that besides $\mathbf{0}$ there is an $\mathbf{a}_I$ in $T(\rho)$, which implies the claim by the remark following Proposition 1. If all the literals come from the first s variables then $\mathbf{a}_I$ corresponding to these literals clearly satisfies the requirements. Otherwise, let us assume that the literals in ρ are of the form $\bar{x}_i$, for $i \in I_1 \cup I_2$, $I_1 \subseteq [s], I_2 \subseteq [s+1, 2s]$, $|I_2| > 0$ and $|I_1| + |I_2| = k$. By part *(b)* of Lemma 10 there is an $I \subseteq [s], |I| = k, I_1 \subset I$ such that $h(I) \cap I_2 = \emptyset$, and by definition, $\mathbf{a}_I \in T(\rho)$. $\square$

Using the results on the relation between the exclusion dimension and the complexity of learning with membership and proper equivalence queries [2, 7, 8] 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.*

PROOF. The query complexity of a class $\mathcal{C}$ is at most $O(XD(\mathcal{C}) \cdot \log |\mathcal{C}|)$ (see, e.g., [2]). The result follows by using the upper bound on the size of k-PDNF$_n$ from Proposition 5, and the upper bound on its exclusion dimension from Theorem 9. $\square$

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 [4], using ideas from Littlestone and Helmbold et al. [9, 12], 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 that includes two complementary projections, that is, 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 all hold:*

(a) $Lit(\rho_i t_i) \not\subseteq Lit(\rho_j t_j)$ *for every* $1 \leq i \neq j \leq \ell$,
(b) $\rho_i \notin Lit(t_i)$ *for every* $1 \leq i \leq \ell$,
(c) *if* $\ell \geq 3$ *then* $\rho_i \neq \bar{\rho}_j$ *for every* $1 \leq i \neq j \leq \ell$.
(d) $\rho_i \notin Lit(t_i)$, *for* $1 \leq i \leq \ell$

Otherwise, φ is called p-redundant.

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.

Given a 1-PDNF expression, one can easily transform it into a p-irredundant form as follows. First delete any term violating *(d)*. Next check if there are two complementary projections, and if there are, then delete all the other terms, thereby guaranteeing *(c)*. Otherwise, delete every term subsumed by another term, ensuring *(a)*. Finally, if in a remaining term the t-part contains the projection literal, then delete the projection literal from that term. The final expression is a p-irredundant 1-PDNF, which is equivalent to the original one.

The above algorithm runs in polynomial time, thus we have:

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

Next we give a description of those p-irredundant 1-projective DNF that represent either a monotone or an **a**-unate function, and then we give the general description. We assume *w.l.o.g.* throughout this section that each 1-PDNF in question determines a non-constant function and has terms that do not contain any complementary literals.

Lemma 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*

$$\varphi = \rho_1 t \vee \cdots \vee \rho_\ell t, \tag{5}$$

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 it is of the form*

$$\varphi = \rho t \vee \bar{\rho} t t', \tag{6}$$

where ρ is an unnegated variable (resp. its sign 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 $\mathbf{x} \oplus \mathbf{a}$. (Note that $f(a)$ is k-PDNF *iff* $f(\mathbf{x} \oplus \mathbf{a})$ is.) It follows directly from the definitions that every expression of the form of Equation (5) or (6) is indeed a p-irredundant 1-PDNF expression.

Let φ be an arbitrary monotone p-irredundant 1-PDNF formula. Separating the negated and unnegated projections, let us write φ as

$$\varphi = \bigvee_{i \in I} (x_i \cdot t_i) \vee \bigvee_{j \in J} (\bar{x}_j \cdot t_j).$$

(This representation of φ is convenient for the following series of claims.)

Claim 15 *The index set I is nonempty, and t_ℓ is monotone for all $\ell \in I \cup J$.*

PROOF. The first part of the Claim holds because φ determines a non-constant monotone function, thus $\varphi(\mathbf{1}) = 1$.

To prove monotonicity for t_i, $i \in I$, note that $\mathbf{1}$ is contained in every unnegated projection, thus by projectivity $x_i t_i(\mathbf{1}) = \varphi(\mathbf{1})$—and this can only hold if t_i is monotone.

Finally, let us consider a term $\bar{x}_j t_j$ with $j \in J$. Asssume for contradiction that term t_j contains negative literal $\bar{x}_s$ for some $1 \leq s \leq n$. (Note that $s \neq j$ by p-irredundancy.) Let $\mathbf{x}$ be any vector satisfying the term $\bar{x}_j \cdot t_j$ and thus φ. By monotonicity $\mathbf{x}^{x_s=1}$ must satisfy φ. However, then, by projectivity, $\mathbf{x}^{x_s=1}$ must satisfy t_j, a contradiction. □

Claim 16 *For all $i \in I$, we have $T(\varphi) \subseteq T(t_i)$.*

PROOF. Let $\mathbf{x} \in T(\varphi)$, so $\varphi(\mathbf{x}) = 1$. 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 $t_i(\mathbf{x}) = 1$, which proves the claim. □

Claim 16 can be used to show that the first half of φ (consisting of the terms with positive literals as projections) is in the right form. Claim 17 does this.

Claim 17 *There must be a single term t such that we can write*

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

PROOF. Consider any two terms $x_i t_i$ and $x_j t_j$ from φ. From Claim 16 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

$$Lit(t_j) \subseteq Lit(x_i t_i) \quad \text{and} \quad Lit(t_i) \subseteq Lit(x_j t_j). \tag{7}$$

It must be the case that $x_j \notin Lit(x_i t_i)$ and $x_i \notin Lit(x_j t_j)$, as otherwise using (7) it follows that φ is p-redundant. But then $Lit(t_j) = Lit(t_i)$. □

Putting together Claims 15 and 17, it follows that we are done if $J = \emptyset$. The remaining case (i.e., when $J \neq \emptyset$) is handled by the following Claim.

Claim 18 *Let π be a monotone p-irredundant 1-PDNF formula of the form*

$$\pi = \bigvee_{i \in I} (x_i \cdot t) \vee \bigvee_{j \in J} (\bar{x}_j \cdot t_j),$$

where I and J are nonempty sets, furthermore t_j, $j \in J$ and t are monotone terms. Then $\pi = x_i t \vee \bar{x}_i t t'$ for some variable x_i and some monotone term t'.

PROOF. It follows from Claim 16 that $T(\bar{x}_j t_j) \subseteq T(\pi) \subseteq T(t)$, thus $Lit(t) \subseteq Lit(\bar{x}_j t_j)$, and so $Lit(t) \subseteq Lit(t_j)$ (note that $\bar{x}_j \notin Lit(t_j)$, as t_j is positive). Thus π can further be written as

$$\pi = \bigvee_{i \in I} (x_i \cdot t) \vee \bigvee_{j \in J} (\bar{x}_j \cdot t t'_j),$$

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 $(x_i \cdot t)$ and $(\bar{x}_j \cdot t t'_j)$ in π such that $i \neq j$. By projectivity $T(\bar{x}_j x_i t) = T(x_i \bar{x}_j t t'_j) \neq \emptyset$, and so either $t'_j = x_i$ or t'_j is the empty term. But $t'_j = x_i$ would imply that π is p-redundant; thus t'_j is the empty term.

If π contains only two terms, it must be of the form $\pi = x_i \cdot t \vee \bar{x_j} \cdot t$. If $x_j \notin t$, then it is not monotone (in variable x_j). If $x_j \in t$, then it is not p-irredundant (violates condition (d) of the definition).

From now on we have that π has at least three terms. Since t'_j is the empty term, we have $T(\bar{x}_j t) \subseteq T(\pi)$, and by monotonicity $T(t) \subseteq T(\pi)$. With Claim 16. this implies $T(t) = T(\pi)$. But then for every other term $\bar{x}_k t t'_k$ of π it holds that $T(\bar{x}_k \pi) = T(\bar{x}_k t)$, meanwhile by projectivity $T(\bar{x}_k t t'_k) = T(\bar{x}_k \pi)$, so t'_k is the empty term. Therefore

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

This can only hold if some variable occurs both in I and J, contradicting condition *(c)* of the definition of p-irredundancy for π. $\square$

This completes the proof of the lemma. $\square$

The example of (3) 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 monotone 1-PDNF functions, which cannot be written as a monotone 1-PDNF. Consider, for example, the 1-PDNF

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

representing the monotone function $x \vee yz$. If there were an equivalent monotone 1-PDNF, then it could 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 19 *A formula φ is a p-irredundant 1-PDNF formula if and only if it is either of the form*

$$\varphi = \bigvee_{i=1}^{s} (\rho_1^i t_i \vee \cdots \vee \rho_{\ell_i}^i t_i),$$

where $\rho_u^i \notin Lit(t_i)$ and $\bar{\rho}_u^j \in Lit(t_i)$ 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 $x \notin Lit(t)$ and $\bar{x} \notin Lit(t')$.

PROOF. Again, one direction of the theorem is immediate from the definition of p-irredundancy. 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 Lemma 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. $\square$

Informally, the first case of the theorem says the following. Let us consider a term in a p-irredundant 1-PDNF to consist of a "stem" t and a "petal" ρ. Then the petal of each term is not included in its stem (that much is clear from the definition of p-irredundancy) and if two terms have different stems then each stem contains the negation of the other one's petal. In other words, each stem consists of the negations of all the petals corresponding to terms with different stems, plus, possibly, some other literals.

6 Revising disjunctions and k-PDNF

In this section we consider two different revision algorithms. First we define the *revision distance* between two k-PDNFs, 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. Similarly, the distance between two disjunctions is also the number of literals occurring in exactly one of the two disjunctions.

The revision distance between an *initial* k-PDNF formula φ_0 and a *target* k-PDNF formula φ of the form

$$\begin{aligned} \varphi_0 &= \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell \vee \rho_{\ell+1} t_{\ell+1} \vee \cdots \vee \rho_{\ell+d} t_{\ell+d} \ , \\ \varphi &= \rho_1 t_1^* \vee \cdots \vee \rho_\ell t_\ell^* \vee \rho'_1 t'_1 \vee \cdots \vee \rho'_a t'_a \end{aligned}$$

is

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

where $\{\rho_{\ell+1}, \ldots, \rho_{\ell+d}\} \cap \{\rho'_1, \ldots, \rho'_a\} = \emptyset$. 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, the addition of a new empty term of the form $\rho \cdot \top$, 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, which is $2^k \binom{n}{k}$.

We begin by demonstrating that the original Winnow [11], with appropriately modified initial weights (see Figure 6.1), is an efficient revision algorithm in the mistake bound model—even in the presence of attribute errors, if we are willing to tolerate a number of 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.

Initialization: For $i = 1, \ldots, n$ 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 $i = 1, \ldots, n$ to

$$w_{r,i} = w_{r-1,i} \cdot 2^{\mathbf{x}_{r,i}(y_r - \hat{y}_r)}$$

Fig. 1. Algorithm RevWinn(φ_0)

6.1 Revising disjunctions

Consider a version of Winnow, which we call RevWinn, presented in Figure 6.1. Note that this is Littlestone's Winnow2 [11] 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 in the interval $(0, 1]$. This can be seen using an induction argument as follows. Initially the statement is true. Now assume that the weights after round $r - 1$ are all between 0 and 1. If $y_r = \hat{y}_r$, then the weights are not changed. If $y_r = 0$ and $\hat{y}_r = 1$, then some weights are halved, and some unchanged—thus the statement will be true after round r. If $y_r = 1$ and $\hat{y}_r = 0$, then $\mathbf{w}_{r-1} \cdot \mathbf{x}_r < 1/2$, so the sum of the weights of components having 1 in vector $\mathbf{x}_r$ is less then $1/2$. As RevWinn doubles the weights of exactly these components, the statement will remain true after round r.

Theorem 20 *The number of mistakes made by Algorithm* RevWinn *with initial (monotone) disjunction* $\varphi_\mathbf{0}$ *and target (monotone) disjunction* φ *is*

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

where $e = dist(\varphi_0, \varphi)$*, and* n *is the number of variables in the universe.*

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 $\overline{I \cup J} = \{1, \ldots, n\} \setminus (I \cup J)$. 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 [3, 10]). 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 #AttributeErrors.

First we derive an upper bound:

$$\begin{aligned} \sum_{r=1}^{R} \Delta_r &= \Phi(\mathbf{w}_0) - \Phi(\mathbf{w}_R) \\ &\leq \Phi(\mathbf{w}_0) - \sum_{i \in \overline{I \cup J}} w_{R,i} \\ &= \sum_{i \in I} \Phi_i(\mathbf{w}_0) + \sum_{j \in J} \Phi_j(\mathbf{w}_0) + \sum_{i \in \overline{I \cup J}} (w_{0,i} - w_{R,i}) \ . \end{aligned} \tag{8}$$

For $i \in I$ we initialized $\mathbf{w}_{0,i} = 1$ so $\Phi_i(\mathbf{w}_0) = 0$. Also, $|J| \leq e$, and $\Phi_j(\mathbf{w}_0) = \ln(2n) - (2n-1)/2n < \ln(2n)$ 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 φ_0 that are not present in φ, 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

$$\sum_{r=1}^{R} \Delta_r \leq e \ln 2n + e + 1/2 \ . \tag{9}$$

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

$$\begin{aligned}\Delta_r &= \sum_{i\in I\cup J}\left(w_{r-1,i} - w_{r,i} + \ln\frac{w_{r,i}}{w_{r-1,i}}\right) + \sum_{i\in\overline{I\cup J}}(w_{r-1,i} - w_{r,i}) \\ &= \sum_{i=1}^{n}(w_{r-1,i} - w_{r,i}) + \sum_{i\in I\cup J}\ln\frac{w_{r,i}}{w_{r-1,i}} \ . \end{aligned} \tag{10}$$

Examining the RevWinn code, we see that there are three cases for updating weights at the end of a round r: no change in any weights, some or all weights are decreased, which we will call a "demotion" round, and some or all weights are increased, which we will call a "promotion" round. Obviously, when no update is done in round r (i.e., $\hat{y}_r = y_r$), then $\Delta_r = 0$.

In a demotion round, $\hat{y}_r = 1$ and $y_r = 0$. By the definition of I and J, in this case $\textsc{AttrErr}(r) = |(I \cup J) \cap \mathbf{x}_r|$.[5] Also, the total weight of components being on in $\mathbf{x}_r$ is at least $1/2$ (recall how $\hat{y}_r$ is evaluated), and the weight of each of those components is halved. So, using (10),

$$\Delta_r \geq 1/4 + |(I \cup J) \cap \mathbf{x}_r| \ln\left(\frac{1}{2}\right) = 1/4 - (\ln 2)\textsc{AttrErr}(r) \ . \tag{11}$$

In a promotion round, $\hat{y}_r = 0$ and $y_r = 1$. We know that the components of $\mathbf{x}_r$ that are on have total weight less than $1/2$ (again, by the evaluation rule of $\hat{y}_r$), and that each of these components is multiplied by 2. So the first term in (10) is at least $-1/2$. Thus $\Delta_r \geq -1/2 + |(I \cup J) \cap \mathbf{x}_r| \ln 2$. Now if $y_r = \varphi(\mathbf{x}_r)$, then $|(I \cup J) \cap \mathbf{x}_r| \geq 1$, because we know that $\hat{y}_r = 0$ and we know that all the weights of variables in both φ_0 and φ but not in I are 1. If $y_r \neq \varphi(\mathbf{x}_r)$, then $\textsc{AttrErr}(r) = 1$. Thus, in a promotion round, it always holds that

$$\Delta_r \geq -1/2 + (\ln 2)(1 - \textsc{AttrErr}(r)) \ . \tag{12}$$

Finally, let M^- denote the total number of demotions and M^+ the total number of promotions. Then (11) and (12) give us

$$\begin{aligned}\sum_{r=1}^{R}\Delta_r \geq &\sum_{\{r:\hat{y}_r=1,y_r=0\}}(1/4 - (\ln 2)\textsc{AttrErr}(r)) \\ &+ \sum_{\{r:\hat{y}_r=0,y_r=1\}}(\ln 2 - 1/2) - (\ln 2)\textsc{AttrErr}(r)\end{aligned}$$

[5] 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 $\mathbf{x}_r$.

$$= M^-/4 + (\ln 2 - 1/2)M^+ - (\ln 2)\#\textsc{AttributeErrors} \ .$$

Combining this with (9) gives the desired mistake bound. □

Notice that, unlike other uses of potential functions in mistake-bound proofs, we do not make any claims about the relation between the value of the potential function used here and the distance between the actual weight vector $\mathbf{w}_r$ and a weight vector for the target. Indeed, we do not see any obvious relation between the value of this potential function and any measure of distance between $\mathbf{w}_r$ and a weight vector for the target.

Remark 21 *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* RevWinn*C.*

6.2 Revising k-PDNF

Now we give a revision algorithm for k-PDNFs. We use Valiant's two-level algorithm [17] 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 Figure 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^ρ_r 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, and its current value in round r will be denoted by $\mathbf{v}_r$. 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

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

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

Theorem 22 *Suppose that the initial and target formulas are, respectively, the k-PDNF$_n$ formulas*

$$\varphi_0 = \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell \vee \rho_{\ell+1} t_{\ell+1} \vee \cdots \vee \rho_{\ell+d} t_{\ell+d} \ ,$$
$$\varphi = \rho_1 t^*_1 \vee \cdots \vee \rho_\ell t^*_\ell \vee \rho'_1 t'_1 \vee \cdots \vee \rho'_a t'_a \ ,$$

1: Initialize a RevWinn instance for the top-level algorithm with initial disjunction $v_{\rho_1} \vee \cdots \vee v_{\rho_{\ell+d}}$.
2: Initialize a RevWinnC instance for each k-projection ρ_i with parameter t_i for $\rho_1, \ldots, \rho_{\ell+d}$ respectively and with parameter $\top$ for the rest.
3: **for** each round r with instance $\mathbf{x}_r$ **do**
4: Set each $v_{r,\rho} = \rho(\mathbf{x}_r) h_r^\rho(\mathbf{x}_r)$
5: Predict $\hat{y}_r = h_r(\mathbf{v}_r)$
6: **if** $\hat{y}_r \neq y_r$
7: Update the weights of the v_ρ variables in the top-level RevWinn for the mistake $h_r(\mathbf{v}_r) \neq y_r$
8: **for** each ρ with $\rho(\mathbf{x}_r) = 1$ and $v_{r,\rho} \neq y_r$
9: Update the low-level RevWinnC instance ρ for a $v_{r,\rho} \neq y_r$ mistake on $\mathbf{x}_r$

Fig. 2. The procedure Rev-k-PDNF(φ_0). The k-PDNF to be revised to another k-PDNF is $\varphi_0 = \rho_1 t_1 \vee \cdots \vee \rho_{\ell+d} t_{\ell+d}$.

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

PROOF. The top-level RevWinn revises 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—that is, $v_{r,\rho} \neq \rho t(\mathbf{x}_r)$ might occur in some round r for some term $(\rho \cdot t)$ of φ. (The actual input $\mathbf{x}$ and classification y are assumed to be noiseless.)

Theorem 20 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 21 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 20, $O(F + (d+a)\log\left(2^k\binom{n}{k}\right)) = O(ek \log n)$, since $F = O(e \log n)$. □

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 [17]: $O(kms \log n)$, where s is the maximum number of variables in any term in the target.

7 Concluding Remarks

In this paper we examined the class of k-PDNF expressions, introduced by Valiant [17] in the context of learning theory. We related this class to other commonly studied classes of Boolean functions, gave combinatorial results concerning its exclusion dimension, and provided a description of the subclass of 1-PDNF functions. It would be interesting to get a description of k-PDNF functions for larger k. Another direction could be to study the computational complexity of algorithmic questions related to PDNF. The exclusion dimension bound leaves open the question whether there is a computationally efficient equivalence and membership query learning algorithm for k-PDNF.

Another contribution of the paper is the extension of Valiant's [17] attribute-efficient learning algorithm for PDNF to an efficient PDNF revision algorithm. The resulting learning algorithm combines the biologically plausible features of localized learning (provided by the projection learning framework) and efficient updating of previously obtained "close but not correct" concept. This may form a step towards the design of biologically realistic learning algorithms for expressive representation formalisms. In a recent paper, the result here on Winnow's revising disjunctions in the presence of noise has been extended to revising PDNFs in the presence of noise [15].

References

[1] Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. Learnability and automatizability. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 621–630. IEEE Computer Society, 2004.

[2] Dana Angluin. Queries revisited. *Theoret. Comput. Sci.*, 313(2):175–194, 2004. Special issue for ALT 2001.

[3] Peter Auer and Manfred K. Warmuth. Tracking the best disjunction. *Machine Learning*, 32(2):127–150, 1998. Earlier version in 36th FOCS, 1995.

[4] Avrim Blum. On-line algorithms in machine learning. Available from `http://www-2.cs.cmu.edu/~avrim/Papers/pubs.html`, 1996.

[5] Judy Goldsmith, Robert H. Sloan, Balázs Szörényi, and György Turán. Theory revision with queries: Horn, read-once, and parity formulas. *Artificial Intelligence*, 156:139–176, 2004.

[6] Judy Goldsmith, Robert H. Sloan, and György Turán. Theory revision with queries: DNF formulas. *Machine Learning*, 47(2/3):257–295, 2002.

[7] Tibor Hegedűs. 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.

[8] Lisa Hellerstein, Krishnan Pillaipakkamnatt, Vijay Raghavan, and Dawn Wilkins. How many queries are needed to learn? *J. ACM*, 43(5):840–862, 1996.

[9] 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).

[10] 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.

[11] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[12] Nick Littlestone. A mistake-bound version of Rivest's decision-list algorithm. Personal communication to Avrim Blum, 1989.

[13] Chris Mesterharm. Tracking linear-threshold concepts with Winnow. *Journal of Machine Learning Research*, 4:819–838, 2003.

[14] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[15] Robert H. Sloan and Balázs Szörényi. Revising projective DNF in the presence of noise. In *Proc. Kalmár Workshop on Logic and Computer Science*, pages 143–152, Szeged, Hungary, October 2003. Dept. of Informatics, University of Szeged. Journal-length version submitted and under review. Both versions available online from URL `http://www.cs.uic.edu/~sloan/papers.html`.

[16] Robert H. Sloan, Balázs Szörényi, and György Turán. Projective DNF formulae and their revision. In *Learning Theory and Kernel Machines, 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 625–639. Springer, 2003.

[17] Leslie G. Valiant. Projection learning. *Machine Learning*, 37(2):115–130, 1999.

[18] Leslie G. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 47(5):854–882, 2000.

[19] Stefan Wrobel. *Concept Formation and Knowledge Revision*. Kluwer, 1994.

[20] Stefan Wrobel. First order theory refinement. In L. De Raedt, editor, *Advances in ILP*, pages 14–33. IOS Press, Amsterdam, 1995.