Machine Learning, 24, 65–85 (1996)

On Learning Visual Concepts and DNF Formulae*

EYAL KUSHILEVITZ** eyalk@das.harvard.edu
DAN ROTH danr@das.harvard.edu
Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138.

Editor: Leonard Pitt

Abstract. We consider the problem of learning DNF formulae in the mistake-bound and the PAC models. We develop a new approach, which is called *polynomial explainability*, that is shown to be useful for learning some new subclasses of DNF (and CNF) formulae that were not known to be learnable before. Unlike previous learnability results for DNF (and CNF) formulae, these subclasses are not limited in the number of terms or in the number of variables per term; yet, they contain the subclasses of k-DNF and k-term-DNF (and the corresponding classes of CNF) as special cases. We apply our DNF results to the problem of learning *visual concepts* and obtain learning algorithms for several natural subclasses of visual concepts that appear to have no natural boolean counterpart. On the other hand, we show that learning some other natural subclasses of visual concepts is as hard as learning the class of all DNF formulae. We also consider the robustness of these results under various types of noise.

Keywords: Computational Learning, DNF, Visual Concepts

1. Introduction

A central question in computational learning theory is deciding which subclasses of boolean formulae are learnable under the standard learning models. One of the main open problems, which has remained open since proposed by Valiant (Valiant, 1984, Valiant, 1985) is the question of the PAC-learnability of disjunction-normal-form (DNF) formulae.

Despite the efforts devoted to resolving this problem, success was obtained only for relatively simple subclasses, such as k-DNF (DNF formulae in which each term consists of at most k literals) and k-term-DNF (DNF formulae with k terms). The widest subclass of boolean formulae known to be PAC-learnable is that of *k-decision lists (k-DL)* (Rivest, 1987) which contains the above two subclasses as special cases. Instead, work has been focused recently on other models where, with the added ability of the learner to make various kinds of queries, a wider collection of subclasses are known to be learnable; these include, for example, monotone DNF (Valiant, 1984), read-twice DNF (Hancock, 1991, Aizenstein & Pitt, 1991, Pillapakkamnatt & Raghavan, 1993), Horn formulae (Angluin, Frazier, & Pitt, 1992), $\log n$-term DNF (Blum & Rudich, 1992), Read-k-Satisfy-j DNF (Aizenstein & Pitt, 1992, Blum et al., 1994), and Decision Trees (Bshouty, 1993). Most recently, Jackson (Jackson, 1994) shows (using the Fourier algorithm of (Kushilevitz & Mansour, 1993)) how to learn the class of DNF formulae with respect to the *uniform distribution* using membership queries. However, for the DNF problem in the

* An earlier version of this paper appeared in the Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory, COLT93.

** Current address: Department of Computer Science, Technion, Israel. e-mail:eyalk@cs.technion.ac.il

PAC model, simple reductions (Kearns et al., 1987) show that the restrictions suggested by researchers to tackle the problem when queries are available, e.g., limiting the number of occurrences of a variable, considering monotone formulae etc., are not useful in the query-less models. Thus the PAC-learnability of DNF formulae (beyond the above mentioned subclasses) is still a mystery.

One reason for the attraction of the class of DNF, aside from its being a theoretical puzzle, is that people appear to like it for representing knowledge (Valiant, 1985). Indeed, in (Shvaytser, 1990) the observation is made, that learnability results on learning DNF and CNF formulae are useful for the task of learning to recognize *visual concepts* in digital pictures. By mapping boolean variables to pixels in an $n \times n$ digital picture, the equivalence to learning DNF is shown[1]. Then, known DNF and CNF learning algorithms are used to learn the corresponding visual concepts.

We present a new approach to the problem of learning DNF formulae. As a result, we are able to learn any subclass of DNF which is *polynomially explainable*. Informally speaking, we show that, when learning DNF it is not necessary for the set of candidate monomials to be of polynomial size (as is the case, for example, with k-DNF formulae); we can handle DNF formulae defined over a super-polynomial set of monomials, as long as it is possible, given a (positive) example, to output a list of monomials (the *explanations*) such that one of them is satisfied by the example. Negative examples are then used to eliminate "false" explanations[2]. (Similar ideas were developed in different contexts, e.g. in (Blum, 1992, Blum & Rudich, 1992, Aizenstein & Pitt, 1992).) The same approach is used for learning subclasses of *CNF* formulae. DNF formulae in the subclasses shown to be learnable are not limited in the number of terms or in the number of variables per terms, and they contain the subclasses of k-DNF and k-term-DNF (and the corresponding subclasses of CNF) as special cases.

We then apply our approach (and results) to the PAC-learnability of visual concepts (in fact, most of our results hold in the mistake-bound model as well). By a visual concept we mean a collection of *patterns* (e.g., rectangles, chairs), that are defined by certain characteristics (e.g., shape, texture). In the learning scenario, examples of pictures ('scenes') are given to a learning algorithm, labeled as positive or negative according to whether they contain at least one member of an unknown collection of patterns or not. The goal is to acquire the skill of labeling future pictures accordingly.

We consider two versions of the problem: (1) A *static* version, in which each pattern in the visual concept has a *fixed* location in the picture and only pictures containing the pattern in that location are labeled positive; (2) A *dynamic* version, in which a pattern might appear anywhere in the picture, i.e., it might be translated or even rotated and scaled in certain ways. The dynamic version seems more suitable for possible applications. However, it is more convenient to consider the static version for demonstrating the correspondence to the DNF problem. Fortunately, all the algorithms we present for the static version of the problem can be easily modified to work in the dynamic case[3]. We hereafter discuss only the static version of the problem.

We show that structural information on the concept classes, (e.g., all concepts are polygons with k edges) can be used to devise efficient learning algorithms. This gives some learning algorithms for several "natural" subclasses of visual concepts. We also exhibit some of the limitations of this method: we use reductions to prove that learning various other visual concept subclasses is as hard as learning general DNF formulae, although these concept classes have seemingly helpful structure.

It is a natural question whether the ideas presented here (along with some "engineering") can be used to solve some "real-world" problems. We believe that they do; indeed, some experiments that use these idea to solve problems concerning the recognition of human motion (Bender & Roth, 1994) and character recognition (Basri, 1994) seem to be encouraging. A major problem from the engineering side of the problem is how to generate the "explanations" in an efficient and compact way.

A more intrinsic difficulty is that our model is very "clean" while "real-world" data (both the examples and their classifications) is often noisy. Therefore, we discuss the problem of learning polynomially explainable classes in the presence of various kinds of noise. We consider classification noise, malicious noise, and a certain type of attribute noise that occurs in visual concepts (when the target object may be obstructed by other objects). We show how these kinds of noise can be tolerated in learning polynomially explainable classes.

In the next section we formally define the learning models considered. In Section 3 we prove the main theorem in terms of DNF (and CNF) formulae. In Section 4 we present the visual learning problem and some of the results in terms of visual concepts. In section 5 we discuss learning in the presence of noise, and in Section 6 we discuss the results and briefly describe some other applications of our technique.

2. Preliminaries

We begin by formally defining the learning models discussed in this work – the standard PAC model (Valiant, 1984) and the mistake-bound model (Littlestone, 1988).

The *instance space* X is $\{0,1\}^n$, the set of all possible assignments to n boolean variables. A *concept* f is a boolean function on X. *Positive* (respectively, *negative*) examples of f are examples (instances) on which f is 1 (respectively, 0). A *concept class* is a collection of concepts.

In the learning scenario, we are given a concept class $\mathcal{C}$ and there is some unknown *target concept* $f_T \in \mathcal{C}$ that we are trying to learn. In the *mistake-bound* model, at each learning stage, an example $x \in X$ is presented; the learning algorithm is asked to predict $f_T(x)$ and is then told whether the prediction was correct. Each time the learning algorithm makes an incorrect prediction, we charge it one *mistake*. We say that $\mathcal{C}$ is *mistake-bound learnable* if there exists a polynomial-time prediction algorithm $\mathcal{A}$ (possibly randomized) that for all $f_T \in \mathcal{C}$ and any sequence of examples is guaranteed to make at most polynomially many (in n) mistakes. We say that $\mathcal{C}$ is *expected mistake-bound learnable* if there exists $\mathcal{A}$, as above, such that the expected number of mistakes it makes for all $f_T \in \mathcal{C}$ and *any* sequence of examples is at most polynomially many (in n). Note that the expectation is taken over the random choices made by $\mathcal{A}$; there is no probability distribution associated with the sequences!.

In learning an unknown target function $f_T \in \mathcal{C}$ in the PAC model, we assume that there is a fixed but arbitrary and unknown *distribution* D over the instance space X. The learning algorithm sees examples drawn independently according to D together with their labeling (positive/negative). Then it is required to predict the value of f_T on another example drawn according to D. Denote by $h(x)$ the prediction of the algorithm on the example $x \in X$. The error of the algorithm with respect to f_T and D is measured by $error(h) = Pr_{x \in D}\{f_T(x) \neq h(x)\}$.

We say that $\mathcal{C}$ is *PAC-learnable* if there exists a polynomial-time learning algorithm $\mathcal{A}$ and a polynomial $p(\, , \,)$ such that for all $n \geq 1$, all target concepts $f_T \in \mathcal{C}$, all distribution D over X, and all $\epsilon > 0$ and $0 < \delta \leq 1$, such that if the algorithm $\mathcal{A}$ is given $p(n, 1/\epsilon, 1/\delta)$ examples, then with probability at least $1 - \delta$, $\mathcal{A}$'s hypothesis, h, is such that $error(h) \leq \epsilon$.

It can be shown that if a concept class $\mathcal{C}$ is learnable in the expected mistake-bound model (and thus in the mistake bound model) then it is PAC-learnable (Haussler et al., 1991).

3. The DNF Problem

In this section we present a *mistake-bound* algorithm for subclasses of DNF formulae satisfying certain properties. This, in particular, implies the PAC-learnability of these subclasses (Littlestone, 1989). The main idea is the following: Consider, for example, Valiant‘s algorithm for learning k-DNF (Valiant, 1984) (many other algorithms share the same structure). Before seeing any example, the algorithm enumerates the set of all (polynomially many) monomials of size at most k. Then, it uses the examples to “eliminate” those monomials which are not consistent with the examples. This method clearly cannot work for subclasses whose underlying set of monomials is super-polynomial. In our approach, we do not enumerate all possible monomials ahead of time, but rather enumerate monomials that “explain” the classification of the particular examples that we see (and as before, examples are also used to eliminate “wrong explanations”). For certain classes in which each example has only polynomially many explanations this gives a learning algorithm with the desired behavior. More formally:

Definition. Let $x_1, x_2, \ldots, x_n$ be a set of n boolean variables, $\mathcal{M}$ be any collection of monomials on the literals $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ and $p(n)$, $q(n)$ and $g(n)$ be polynomials. Let $\mathcal{C}_{\mathcal{M}}$ be the class of all functions which are disjunctions of at most $p(n)$ monomials in $\mathcal{M}$. We call $\mathcal{C}_{\mathcal{M}}$ *polynomially explainable* if there exists an efficient (polynomial-time) algorithm $\mathcal{A}$ such that for every function $f \in \mathcal{C}_{\mathcal{M}}$, and every positive example of f as input, $\mathcal{A}$ outputs at most $q(n)$ monomials (not necessarily all of them are in $\mathcal{M}$) such that with probability at least $1/g(n)$ at least one of them appears in f (where the probability is taken over the coin-flips of the algorithm $\mathcal{A}$, in case that $\mathcal{A}$ is a probabilistic algorithm).

We emphasize that f itself is *not* given to the algorithm $\mathcal{A}$. Also note that a function f in the class $\mathcal{C}_{\mathcal{M}}$ may have few logically-equivalent representations as a disjunction of monomials in $\mathcal{M}$. The definition requires the output of the algorithm $\mathcal{A}$ to satisfy the above property, independently of which of these representations of f is considered. The importance of this will become clear when we analyze the learning algorithm below.

THEOREM 1 *If $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable then $\mathcal{C}_{\mathcal{M}}$ is expected mistake-bound learnable. Furthermore, if $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable by an algorithm $\mathcal{A}$ that always outputs at least one term of f (i.e., $g(n) \equiv 1$) then $\mathcal{C}_{\mathcal{M}}$ is mistake-bound learnable.*

Proof: We present an expected mistake-bound algorithm that learns the class $\mathcal{C}_{\mathcal{M}}$ (with expected number of mistakes which is $O(p(n) \cdot q(n) \cdot g(n))$). The algorithm is similar to an algorithm presented in (Blum, 1992). The algorithm maintains an hypothesis h which is a disjunction of monomials. Initially h contains no monomials (i.e., $h \equiv FALSE$). Upon receiving an example e, the algorithm predicts $h(e)$; if the prediction is correct, h is not updated. Otherwise, upon a mistaken prediction, it proceeds as follows:

- If e is positive: execute $\mathcal{A}$ (the algorithm guaranteed by the assumption that $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable) on the example e and add the monomials it outputs to h.
- If e is negative: remove from the hypothesis h all the monomials that are satisfied by e (there must be at least one).

To analyze the algorithm we first fix a representation for f as a disjunction of monomials in $\mathcal{M}$ (in case f has more than one possible representation, choose one arbitrarily; The above definition guarantees that we can work with *any* representation of f that uses only monomials in $\mathcal{M}$). Now, note that a ‘true’ monomial, i.e., a monomial that appears in this representation of the target function f, is never removed from h. Therefore, since on a positive example e, the algorithm $\mathcal{A}$ is guaranteed to output at least one monomial that appears in f, with probability at least $1/g(n)$, then the expected number of mistakes made on positive examples is at most $p(n) \cdot g(n)$. This also implies that the expected total number of monomials included in h during the execution of the algorithm is not more than $p(n) \cdot q(n) \cdot g(n)$.[4] Each mistake on a negative example results in removing at least one of those monomials that were included in h but do not appear in f. The expected number of these monomials is therefore at most $p(n) \cdot q(n) \cdot g(n)$. We get that the expected total number of mistakes made by the algorithm is $O(p(n) \cdot q(n) \cdot g(n))$.

Finally, note that in the case $g(n) \equiv 1$ we get a truly mistake-bound algorithm, whose number of mistakes is bounded by $p(n) \cdot q(n)$. ■

A special case of the above theorem is learning k-DNF formulae. It is obtained by taking $\mathcal{M}$ to be the set of all monomials of size k. In this case $\mathcal{M}$ itself is of polynomial size. The following corollary gives another important special case of the theorem, in which $\mathcal{M}$ might be of exponential size but $\mathcal{C}_{\mathcal{M}}$ is still learnable.

COROLLARY 1 *Let $S_1, ..., S_t$ be subsets of $\{x_1, \ldots, x_n\}$, where t is polynomial in n. Let $\mathcal{B}$ be an efficient algorithm that on input n enumerates these sets (and possibly some more). Let $\mathcal{M}$ be any collection of monomials with the property that for every $m \in \mathcal{M}$ the set of variables in m is S_i for some $1 \leq i \leq t$ (i.e., any set S_i may correspond to at most $2^{|S_i|}$ monomials in $\mathcal{M}$, by choosing for each $x_j \in S_i$ whether x_j or $\bar{x}_j$ appears in the monomial). Then, $\mathcal{C}_{\mathcal{M}}$ is mistake-bound learnable.*

Proof: We use the algorithm $\mathcal{B}$ to construct an algorithm $\mathcal{A}$, as required in the conditions of Theorem 1. The algorithm $\mathcal{A}$ works as follows: given a positive example e, it first uses $\mathcal{B}$ to produce the sets $S_1, \ldots, S_t$ (and possibly others). Then, for each such set S_i there is exactly one monomial m_i^e which contains all the variables of S_i and is satisfied by e (for each $x_j \in S_i$ the monomial contains x_j if the j-th bit of e is 1 and $\bar{x}_j$ if the j-th bit is

0). $\mathcal{A}$ outputs $m_1^e, \ldots, m_t^e$. Notice that for each positive example e, at least one of these monomial must be in $\mathcal{M}$ (but some may not). ■

The following examples are all special cases of the this Corollary. Other examples can be obtained by "translating" the examples of visual concept subclasses, that are given in Section 4 below, to the terminology of DNF formulae; however, the resulted subclasses of DNF seem to be somewhat artificial.

Example: Consider the class of all DNF formulae in which each term contains *at least* $n-k$ literals (for some constant k). This class was previously considered in (Li & Vitanyi, 1989). In spite of its similarity to the class of k-DNF formulae (where each term contains *at most* k literals) its learnability seems to be more difficult. This is because in this case there are exponentially many possible monomials to choose from (as opposed to the case of k-DNF where there are only polynomially many possible monomials). Nevertheless, Corollary 1 implies that this class is learnable[5]. □

Example: Consider the class of all DNF formulae in which the variables in each monomial have consecutive indices; e.g., $x_1\bar{x}_2x_3x_4 \vee \bar{x}_4\bar{x}_5x_6 \vee x_8x_9$. The above corollary shows that this class of functions can be efficiently learned since the $\binom{n}{2} < n^2$ sets $S_{i,j}$ $(1 \leq i \leq j \leq n)$ defined by $S_{i,j} = \{x_i, x_{i+1}, \ldots, x_j\}$ satisfy the above condition. □

3.1. The CNF Problem

The polynomial explainability approach can be easily extended to deal with CNF formulae. To show that, we prove the following theorem, which is an analog of Corollary 1. This result includes as special case the learnability of k-CNF formulae (and therefore also the learnability of k-term-DNF formulae).

THEOREM 2 *Let $p(n)$ be a polynomial. Let $S_1, \ldots, S_t$ be subsets of $\{x_1, \ldots, x_n\}$, where t is polynomial in n. Let $\mathcal{B}$ be an efficient algorithm that on input n enumerates these sets (and possibly some more). Let $\mathcal{D}$ be any collection of disjunctions with the property that for every $d \in \mathcal{D}$ the set of variables in d is S_i for some $1 \leq i \leq t$. Then, the concept class $\mathcal{C}_{\mathcal{D}}$ of all functions which are conjunctions of at most $p(n)$ disjunctions in $\mathcal{D}$, is mistake-bound learnable.*

Proof: We present a learning algorithm which is dual to the algorithm in the proof of Theorem 1. The algorithm maintains an hypothesis h which is a conjunction of disjunctions (i.e., a CNF formula). Initially h contains no disjunctions (i.e., $h \equiv$ TRUE). Upon receiving an example e, the algorithm predicts $h(e)$; if the prediction is correct, h is not updated. Otherwise, upon a mistaken prediction, it proceeds as follows:

- If e is negative: this means that the hypothesis is still missing at least one disjunction that is not satisfied by the example e. For each S_i there is exactly one candidate disjunction like that. We add all these disjunctions to h.

- If e is positive: this means that h contains at least one disjunction that does not appear in the target function. We remove from h all the disjunctions that are not satisfied by e.

With an argument similar to the one used in the proof of Theorem 1 it can be shown that the number of mistakes made is at most $p(n) \cdot t$. ■

Example: Consider the class of all CNF formulae in which the variables in each disjunction have consecutive indices; e.g., $(x_3 \vee \bar{x}_4 \vee x_5 \vee x_6) \wedge (\bar{x}_5 \vee \bar{x}_6 \vee x_7 \vee x_8 \vee x_9)$. The above theorem shows that this class of functions is efficiently learnable. □

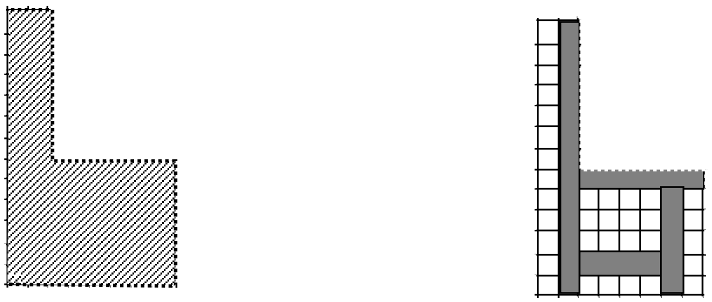

Figure 1. Shape Pattern

4. Learning Visual Concepts

In this section we discuss the learnability of classes of visual concepts. We give positive and negative results in terms of some properties of these classes. We start with some definitions.

Given an $n \times n$ array, a *shape* of *size* k is a collection of k points of the array,

$$S = \{(i_\ell, j_\ell); 1 \leq i_\ell, j_\ell \leq n, 1 \leq \ell \leq k\}.$$

A *binary pattern*, associated with a shape S, is defined by giving a boolean value to every point in S. Formally,

$$P = \{\{(i_\ell, j_\ell), b_\ell\}; (i_\ell, j_\ell) \in S, b_\ell \in \{0, 1\}\}.$$

In this way, any shape of size k can be associated with up to 2^k different patterns. An $n \times n$ *picture* is a pattern whose shape is of size n^2, that is, every point of the $n \times n$ array is given

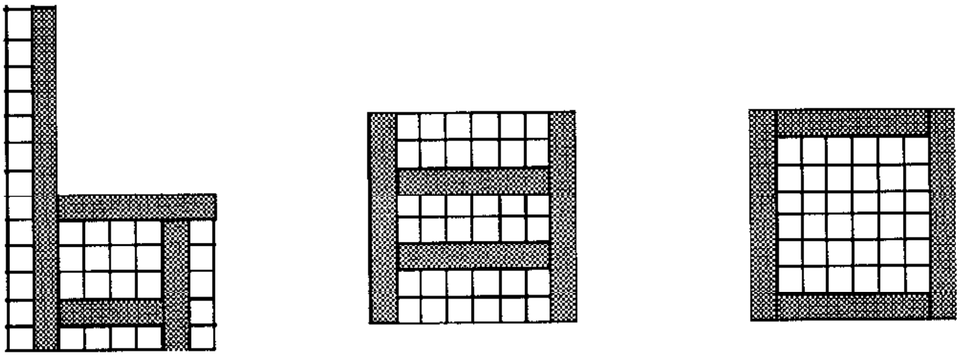

Figure 2. A Visual concept

a boolean value. Throughout the paper we consider shapes and patterns embedded in $n \times n$ pictures, without mentioning the dependence on n explicitly.

Figure 1 gives examples of a shape and a pattern associated with it. (We associate the boolean value 1 with, say, black squares and 0 with white squares.)

Given a collection $\mathcal{P}$ of patterns and a polynomial $p(n)$, the class of all *visual concepts* over $\mathcal{P}$ is defined by

$$C_{\mathcal{P}} = \{T \subseteq \mathcal{P} : \mathcal{P} \text{ is a set of patterns and } |T| \leq p(n)\}.$$

A *visual concept* T is thus a collection of at most $p(n)$ many patterns from $\mathcal{P}$.

We consider the following problem: given $n \times n$ pictures, labeled as positive or negative according to whether they contain a pattern from an unknown collection $T \in C_{\mathcal{P}}$ or not, we try to "learn" T. In Figure 3 positive and negative examples are labeled with respect to the concept defined in Figure 2.

A set $T \subseteq \mathcal{P}$ as above defines a boolean concept in a natural way; moreover, it has a simple translation to the terminology of DNF formulae: the n^2 pixels can be viewed as a set of n^2 boolean variables, each shape is just a subset of these variables, each pattern in T corresponds to a monomial over the set of variables defined by its shape, and the boolean visual concept is the disjunction of those monomials (i.e., it is a DNF formulae). The concept class $C_{\mathcal{P}}$ is the collection of all those concepts.

When no restrictions are made on the patterns collection $\mathcal{P}$ (i.e., $\mathcal{P}$ is the set of all possible patterns), the problem of learning the concept class $C_{\mathcal{P}}$ is clearly equivalent to that of learning the class of all DNF formulae on n^2 variables. The following theorem shows that some non-trivial pattern collections can still be learned. (We consider learning in the mistake-bound model, which implies, as mentioned, PAC-learnability.)

We say that a collection $\mathcal{S}_n$ of shapes on the $n \times n$ array is *polynomially enumerable* if there exists an algorithm that enumerates all the shapes in $\mathcal{S}_n$ in time polynomial in n. In particular, this implies that the set of shapes is of polynomial size.

THEOREM 3 *Let $\mathcal{P}$ be a collection of patterns on the $n \times n$ array, whose corresponding set of shapes is polynomially enumerable. (Notice that $\mathcal{P}$ may be of super-polynomial size). Then, $C_{\mathcal{P}}$ is mistake-bound learnable.*

Proof: For the proof, it is convenient to consider the subclass of DNF formulae corresponding to $\mathcal{C}_{\mathcal{P}}$ (by the correspondence described above). In this terminology, restricting the set of shapes to be polynomially enumerable means that we can enumerate the sets of variables over which the monomials (in the corresponding subclass of DNF formulae) are defined. Hence, we can apply Corollary 1 to get the result. ■

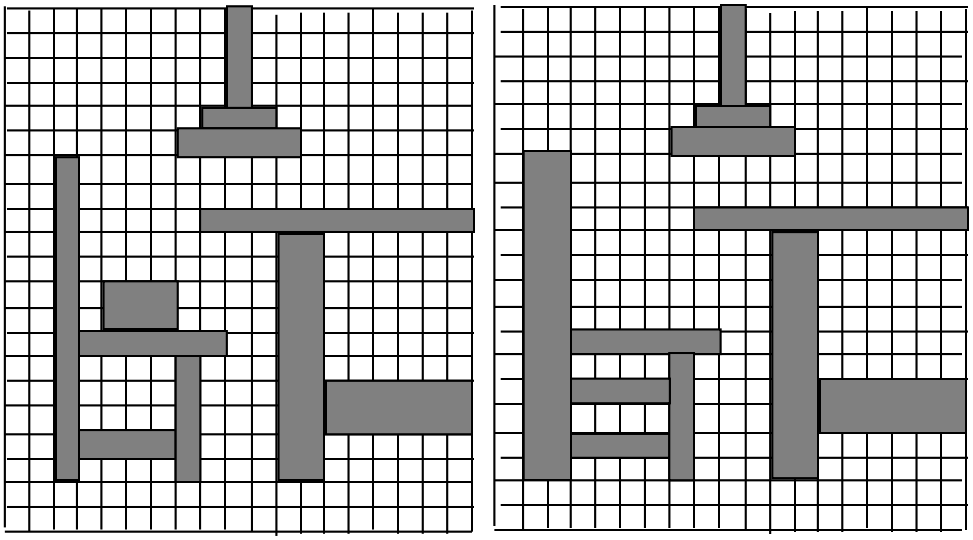

Figure 3. Positive Example Negative Example

To present concrete examples of interesting classes of patterns that are learnable, it will be convenient to think of the picture as embedded in the real plane, where the point (i, j) of the picture corresponds to the square defined by the real points $(i - 1, j - 1), (i - 1, j), (i, j - 1), (i, j)$. Thus, a shape corresponds to a set of polygons with integer vertices and edges which are axis-parallel (see Figure 1). An *edge* of a shape is any of the edges of the associated polygons. The *perimeter* of a shape is the total perimeter of the associated polygons.

COROLLARY 2 *$\mathcal{C}_{\mathcal{P}}$ is learnable for the following collections of patterns:*

- *$\mathcal{P}$ is a collection of patterns whose shapes have a constant number of edges (e.g., rectangles).*
- *$\mathcal{P}$ is a collection of patterns whose shapes consist of a constant number of polygons, each with perimeter $O(\log n)$.*

Proof: Using Theorem 3, it is enough to show that the set of shapes corresponding to each of these collections is polynomially enumerable (actually, we only argue that these sets are of polynomial size; the computation involved with enumerating them is very easy):

- The number of shapes with constant number (c) of edges is polynomial since each of them can be specified by a list of c integer points, and there are $(n+1)^2$ such points (so there are at most $(n+1)^{2c}$ such shapes[6]). Hence, this set of shapes is polynomially enumerable.
- The number of shapes with constant number of polygons with perimeter $O(\log n)$ is polynomial since each of these polygons can be specified by specifying an integer starting point and then $O(\log n)$ times specifying one of the directions: *up, down, left, right* (so there are at most $4^{O(\log n)} = poly(n)$ such shapes). Hence, this set of shapes is polynomially enumerable.

■

In fact, the result above can be extended in many ways. One can consider, for example, polygons with integer vertices and constant number of edges which are not necessarily axis-parallel, and define the shape corresponding to this polygon to be, say, the set of all unit squares (i.e, $(i-1,j-1),(i-1,j),(i,j-1),(i,j)$) contained in the polygon[7]. The number of such shapes is polynomial (using the same proof as we used above for the case of axis-parallel edges) and only the computation of which pixels are in the shape is a bit different. Other families can be defined, for example, by the set of all circles, ellipses etc. (where the corresponding shapes are defined in a similar way). In this way, for example, scaling does not affect the actual number of edges of a polygonal shapes, and thus their learnability.

As remarked before, the problem of learning $\mathcal{C}_{\mathcal{P}}$, where $\mathcal{P}$ is not restricted, is equivalent to learning general DNF formulae. Next we show that the problem remains difficult even if we are restricted to a family of patterns with a fairly simple shape; in particular, we prove it for the case where the shape is a single (simple) polygon.

THEOREM 4 *Let $\mathcal{P}$ be the family of patterns whose shape is a simple polygon. Then, learning $\mathcal{C}_{\mathcal{P}}$ is equivalent to learning DNF.*

Proof: We show, using a reduction, that given an algorithm for learning $\mathcal{C}_{\mathcal{P}}$ we can learn the class of DNF formulae. The result holds in both the PAC and the mistake-bound models. For simplicity we present it in the mistake-bound model.

Let $\phi(x_1,\ldots,x_n) = m_1 \vee m_2 \vee \ldots \vee m_t$ be the target DNF function, where each of the m_i's is a monomial. With each m_i we associate a pattern p_i as follows: the pattern p_i includes all the points $(1,j)$ with the value 1. In addition, the point $(2,j)$ is included in the pattern if and only if x_j or $\bar{x}_j$ appear in m_i: it is included with the value '1' if x_j appears in m_i and with the value '0' if $\bar{x}_j$ appears in m_i. Note that the construction immediately implies $p_i \in \mathcal{P}$ (including the points $(1,j)$, for all j, in the pattern guarantees that the corresponding shape is a simple polygon). Let $T = \{p_1,\ldots,p_t\}$ and let f_T be the corresponding boolean function.

Using the above definitions, we now show how to construct a mistake-bound algorithm $\mathcal{A}$ for learning DNF, given a mistake-bound algorithm $\mathcal{B}$, for learning the concept class $\mathcal{C}_{\mathcal{P}}$ defined above. Given an example $x = (x_1,\ldots,x_n)$, the algorithm $\mathcal{A}$ constructs a picture $\hat{x}$

by letting all its points contain '1' except for each of the points $(2, j)$ whose value is the value of the corresponding variable x_j. The main observation is that for all x, $\phi(x) = f_T(\hat{x})$. Algorithm $\mathcal{A}$ feeds $\hat{x}$ as an input to $\mathcal{B}$, and uses the prediction of $\mathcal{B}$ on $\hat{x}$ as its prediction on x. When it receives the label of x it feeds it to $\mathcal{B}$ as the label of $\hat{x}$. Clearly, $\mathcal{A}$ has the same running-time as $\mathcal{B}$. Also, as $\mathcal{B}$ is guaranteed to make at most polynomially many mistake on *any* $f_T \in \mathcal{C}_{\mathcal{P}}$ and *any* sequence of examples, then in particular this is true for f_T and the examples as defined above. By the observation, $\mathcal{A}$ makes at most polynomially many mistakes as well. ■

Our positive result shows, in particular, that patterns whose shapes are bounded by k edges are learnable (even though these patterns might correspond to monomials of $O(n)$ variables). On the other hand we have:

COROLLARY 3 *Let $\mathcal{P}$ be a collection of patterns whose shape is bounded by $k(n)$ edges. Learning $\mathcal{P}$ is as hard as learning $4(k(n) - 1)$-DNF.*

Proof: We use the same reduction as in the proof of Theorem 4. The result follows since a monomial m_i of length $k(n)$ is mapped to a pattern p_i whose shape has at most $4(k(n) + 1)$ edges. ■

5. Noisy Data

In this section we consider the robustness of learning *polynomially explainable* classes. First we consider a type of noise that is relevant for the class of visual concepts. We define *obstruction noise*, a type of noise that occurs in pictures, and give a learning algorithm that can tolerate this type of noise. Then we consider more general types of noise that have been considered previously in learning, namely, classification noise (Angluin & Laird, 1988) and malicious noise (Valiant, 1985, Kearns & Li, 1993). We show that any polynomially explainable class can be learned using statistical queries (Kearns, 1993), and therefore can be learned in the presence of classification noise with error rate of up to $1/2$, and in the presence of a certain amount of malicious error.

5.1. Obstruction Noise

In this section we consider the problem of learning a pattern in the presence of "noise" in the pictures (but not on their labeling). The type of noise we allow is the one that usually occurs when other objects appear in the picture, behind or in front of the target object (see Figure 4). We also restrict ourselves to the case where the concept consists of a *single* pattern. We first define formally the type of noise considered:

Let p_1, p_2 be two patterns. We say that p_1 $k(n)$*-dominates* p_2 if p_1 can be obtained from p_2 by changing at most $k(n)$ of the '0's of p_2 to '1's. Formally, (a) both patterns have the same shape, (b) all the '0's of p_1 are also '0' in p_2, and (c) all but at most $k(n)$ of the '1's of p_1 are '1's in p_2.

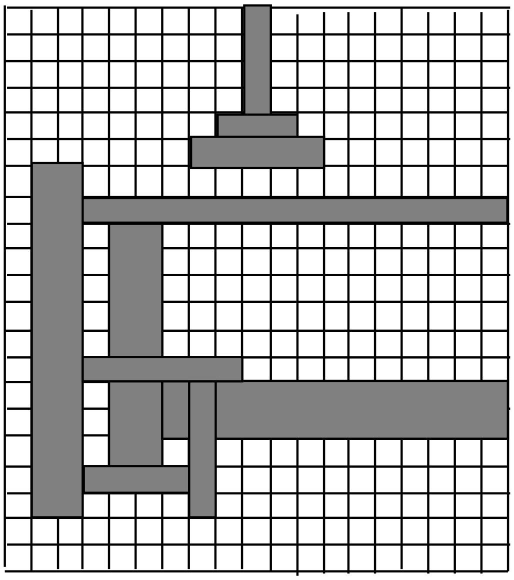

Figure 4. Noise

Let $\mathcal{P}$ be a collection of patterns, whose corresponding set of shapes is polynomially enumerable. We associate with $\mathcal{P}$ a concept class $C_{\mathcal{P},k(n)} = \{f_{p,k(n)} | p \in \mathcal{P}\}$ where $f_{p,k(n)}$ is defined to be positive if and only if the input picture contains a pattern p' (not necessarily in $\mathcal{P}$) that $k(n)$-dominates p.

THEOREM 5 *Let $\mathcal{P}$ be a collection of patterns, whose corresponding set of shapes is polynomially enumerable. Then the concept class $C_{\mathcal{P},k(n)}$ is PAC-learnable.*

Proof: Let $\{S_1, \ldots, S_t\}$ be the polynomial-size set of shapes guaranteed by the assumption that $\mathcal{P}$ is polynomially enumerable. We give an "Occam" algorithm that learns $C_{\mathcal{P},k(n)}$ using $C_{\mathcal{P}',k'(n)}$ as the hypothesis space, where $\mathcal{P}'$ is the collection of *all* patterns whose shapes are in $\{S_1, \ldots, S_t\}$, and $k'(n) \leq n^2$. Given a sample of m labeled examples, we show how to construct in polynomial time a consistent hypothesis of size that is independent of m. Blumer et al. (Blumer et al., 1987) show that this is sufficient for PAC-learnability.

Given a sample of size m, the learner starts by enumerating all the shapes. For each shape S_i, assuming it is the true shape, the learner identifies a target pattern p_i and a threshold k_i, and construct a hypothesis $f_i = f_{p_i,k_i} \in C_{\mathcal{P}',k'(n)}$. The algorithm finally returns the lowest indexed f_i that is consistent with all the examples in the sample.

For a particular shape S_i ($i = 1, \ldots, t$), a pair (p_i, k_i) is identified as follows:

- *Finding the pattern:* The learner considers only the positive examples in the sample. The set $T_i \subseteq S_i$ is defined to be the set of all the points in S_i with value '1' in all the positive examples. The pattern p_i is defined to be 1 on points of T_i and 0 on points of $S_i \setminus T_i$.

- *Finding a threshold:* The learner considers the negative examples in the sample that have 1's in all the points of T_i. For each such example, it computes the number of points in $S_i \setminus T_i$ which are '1' in the example. It takes k_i to be minimum of these values minus 1. If no such example exists it chooses k_i arbitrarily.

The correctness of the algorithms relies on the observation that if $S = S_r$ is the correct shape then the hypothesis $f_r = f_{p_r,k_r}$ found in the r-th round of the algorithm is consistent with all the examples in the sample. Notice that a hypothesis determined based on a wrong shape is not necessarily consistent with all the examples.

Denote by T_S the set of 1's of the true pattern in the true shape S, and by k_S the true threshold. To see that f_r is indeed consistent with the examples, notice that by the construction of T_r, $T_S \subseteq T_r$, but $|T_r| \leq |T_S| + k_S$. Let e be a picture in the sample, that is labeled positive by the hypothesis f_r: either the set of 1's in S is the set T_r, in which case e is indeed a positive example by the above inequality, or that the number of 1's in $S \setminus T_r$ is less than k_r (by the definition of k_r and since e is labeled positive by f_r), and again, it must be a positive example by the way k_r was determined. Similarly, given a picture e that is labeled negative by the hypothesis f_r, it either has a 0 inside T_r, which implies, by the way T_r was constructed, that e is indeed a negative example, or, there are more than k_r 1's in $S \setminus T_r$, which again implies, by the way k_r was determined, that it must be a negative example.

Thus, the algorithm above produces, in polynomial time, a hypothesis in $C_{\mathcal{P}',k'(n)}$ that is consistent with the sample. To represent this hypothesis we need $O(n^2)$ bits (n^2 bits to specify the shape, n^2 bits to specify values for the bits of the shape, and $2 \log n$ to specify the threshold) which is independent of the sample size, m. As shown in (Blumer et al., 1987), using a sample size that is proportional to $\log |C_{\mathcal{P}',k'(n)}|$ is sufficient for PAC-learnability. ■

Note that we do not use any "structure" of the patterns in $\mathcal{P}$, except the fact that the corresponding shapes are polynomially enumerable.

A dual type of noise, in which the target concept is "over-exposed to the light" rather than obstructed, can also be defined. Formally, if p_1, p_2 be two patterns, we say that p_1 is $k(n)$-*dominated* by p_2 if p_1 can be obtained from p_2 by changing at most $k(n)$ of the '1's of p_2 to '0's (that is, p_2 $k(n)$-*dominates* p_1). Similarly, given a collection $\mathcal{P}$ of patterns, whose corresponding set of shapes is polynomially enumerable, we can associate with it a concept class $C'_{\mathcal{P},k(n)}$; in this case an input picture is defined to be positive if and only if it contains a pattern p' (not necessarily in $\mathcal{P}$) that is $k(n)$-dominated by $p \in \mathcal{P}$. It is not hard to see that the same algorithm used in the proof of Theorem 5, when exchanging the roles of 0 and 1 there, leads to the analogous result with respect to this type of noise.

It is worth noticing why the results above are restricted to the case where the concept consists of a *single* pattern. The problem is a "credit assignment" type problem: the technique we use in order to find the minimal "1"-part of the pattern does not work even if there are two possible patterns that can make a picture positive.

5.2. Classification Noise

In this section we discuss PAC learning with *classification noise* (Angluin & Laird, 1988). In this case, whenever we get an example, there is some probability $\eta < 1/2$ (usually referred to as the *error rate*) that the label of this example is flipped (from 0 to 1 or vice versa). Before proving that polynomially explainable concept classes can be learned in the presence of classification noise, we show a robustness property of these classes that will be useful for the proof.

LEMMA 1 *If $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable, then there exists an efficient (polynomial time) algorithm $\mathcal{B}$ such that for every function $f \in \mathcal{C}_{\mathcal{M}}$, given any positive example z of f as input, $\mathcal{B}$ outputs a list of at most $q(n)$ monomials such that with probability at least* $1/2$ all *the terms in f that satisfy z appear in the list. (As before, this is independent of the representation of f.)*

Proof: Since $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable, then there exists an efficient algorithm $\mathcal{A}$ such that for every function $f \in \mathcal{C}_{\mathcal{M}}$, given any positive example z of f as input, $\mathcal{A}$ outputs a list of at most $q'(n)$ monomials such that with probability at least $1/g(n)$ there exists a term of f that satisfies z and appears in the list. We define $\mathcal{B}$ to be as follows: on a positive example z it executes $\mathcal{A}$ for $n \cdot g(n)$ times and outputs all terms produced by $\mathcal{A}$ in those executions (i.e, at most $q(n) = n \cdot g(n) \cdot q'(n)$ terms).

Our first claim is that for every function $f \in \mathcal{C}_{\mathcal{M}}$ and every positive example z of f,

$$Prob[\mathcal{B} \text{ fails to output a term of } f \text{ that satisfies } z] \leq \frac{1}{e^n}. \tag{1}$$

This is because the probability that $\mathcal{A}$ fails to output such term in a single execution is at most $1 - 1/g(n)$. Therefore the probability that $\mathcal{A}$ fails in all $n \cdot g(n)$ executions is at most $(1 - 1/g(n))^{n \cdot g(n)} \leq e^{-n}$.

We now show that $\mathcal{B}$ has the property required in the lemma. Otherwise, there exists a function $f \in \mathcal{C}_{\mathcal{M}}$ and a positive example z, such that the probability that $\mathcal{B}$ outputs all the terms in f that satisfy z is less than $1/2$. Let $t_1, \ldots, t_d$ be all the terms in f that satisfy z. Since with probability greater than $1/2$ the algorithm $\mathcal{B}$ fails to output one of the terms $t_1, \ldots, t_d$, then there exists a term t_i ($1 \leq i \leq d$) such that the probability that $\mathcal{B}$ "misses" it is greater than $1/2d$. Consider now the function $f' \equiv t_i$. Clearly, $f' \in \mathcal{C}_{\mathcal{M}}$ and z is a positive example for it as well. However,

$$Prob[\mathcal{B} \text{ fails to output a term of } f' \text{ that satisfies } z] \geq \frac{1}{2d}, \tag{2}$$

where d is bounded by a polynomial in n. This contradicts Eq. (1) above, and proves the lemma. ■

For the main theorem of this section we use a the "statistical queries" (SQ) model recently introduced by Kearns (Kearns, 1993). The SQ learning model can be viewed as a tool for demonstrating that a PAC learning algorithm is noise-tolerant. We first introduce the SQ learning model and state Kearns result, and in the next theorem give a statistical queries algorithm for the class $\mathcal{C}_{\mathcal{M}}$. In the SQ model, the example oracle $EX(f, D)$ of the standard PAC model, which provides examples of the function f drawn randomly according to the distribution D, is replaced by a statistics oracle $STAT(f, D)$. An SQ algorithm queries the $STAT$ oracle for the values of various statistics on the distribution of labeled examples (e.g., ”What is the probability that a randomly chosen labeled example (e, l) has variable $x_i = 1$ and $l = 0$?”), and the $STAT$ oracle returns the requested statistics to within some specified additive error. Formally, a statistical query is of the form $[\chi, \tau]$. Here χ is a mapping from labeled examples to $\{0, 1\}$ corresponding to an indicator function for those labeled examples about which the statistics are to be gathered, while τ is an additive error parameter, the *tolerance* of the query. A call $[\chi, \tau]$ to $STAT(f, D)$ returns an estimate $\hat{P_\chi}$ of $P_\chi = Pr_D[\chi(x, f(x))]$ which satisfies $|\hat{P_\chi} - P_\chi| \leq \tau$. An SQ algorithm is said to be efficient if $1/\tau$, the time required to evaluate χ and the running time of the algorithm are all polynomial.

In addition to the oracle $STAT(f, D)$ we will provide the learner access to a source of *unlabeled* examples drawn randomly according to the distribution D. To summarize, we formally define learnability in the SQ model:

We say that a class $\mathcal{C}$ of concepts over X is *efficiently learnable from statistical queries* if there exists a learning algorithm $\mathcal{A}$ and polynomials $p(.,.,.), q(.,.)$ and $r(.,.,.)$ such that for any $f \in \mathcal{C}$ over input of length n, for any distribution D over X, and for any $0 < \varepsilon \leq 1$ and $0 < \delta \leq 1$ the following holds: if $\mathcal{A}$ is given inputs ε, δ, n and $size(f)$, and $\mathcal{A}$ is given access to $STAT(f, D)$ and a source $EX(f, D)$ of unlabeled examples, then (1) for every query $[\chi, \tau]$ made by $\mathcal{A}$, χ can be evaluated in time $q(n, size(f))$ and $1/\tau$ is bounded by $r(1/\varepsilon, n, size(f))$, and (2) $\mathcal{A}$ halts in time bounded by $p(1/\varepsilon, 1/\delta, n, size(f))$ and with probability at least $1 - \delta$ outputs a hypothesis h that satisfies $error(h) \leq \varepsilon$.

Notice that this variation of the SQ model, that allows the learner to obtain unlabeled examples, does not give the learner any advantage towards tolerating classification noise over the “pure” SQ model. This is clear from the fact that when learning with noise, the learner also has access to this oracle (simply by ignoring the (possibly noisy) labels). The access to the source of random examples, though, reintroduces the confidence parameter δ into this model, since the learning algorithm is allowed a small probability of error, due to an unrepresentative sample from $EX(f, D)$. We now state the result we use later in this section:

THEOREM 6 ((KEARNS, 1993)) *Let $\mathcal{C}$ be a class of concepts over X and suppose that $\mathcal{C}$ is efficiently learnable from statistical queries using an algorithm $\mathcal{A}$. Then, for any noise rate $0 \leq \eta < 1/2$, $\mathcal{C}$ is learnable with classification noise. The running time of the noise tolerant algorithm is proportional to the running time of the SQ algorithm and to $1/\tau$ and $1/(1 - 2\eta)^2$. (Here τ is a lower bound on the tolerance of the statistical queries used in $\mathcal{A}$.)*

THEOREM 7 *If $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable then $\mathcal{C}_{\mathcal{M}}$ is PAC learnable with classification noise of rate $0 \leq \eta < 1/2$.*

Proof: Using Kearns results stated in Theorem 6 it is sufficient to prove that such a concept class can be learned using a “statistical queries” algorithm.

Let $\mathcal{B}$ be the algorithm guaranteed by Lemma 1 and $q(n)$ an upper bound on the number of monomials $\mathcal{B}$ outputs given an example. Let $p(n)$ be an upper bound on the size of functions in $\mathcal{C}_{\mathcal{M}}$. We now describe a "statistical queries" algorithm for the class $\mathcal{C}_{\mathcal{M}}$.

- *Collecting terms:* The learner draws $N = \frac{4p^2(n)}{\varepsilon} \cdot \ln \frac{p(n)}{\delta}$ examples. For each of them it executes the algorithm $\mathcal{B}$ to get at most $q(n)$ terms. (We assume, without loss of generality, that $\mathcal{B}$ halts in polynomial time even if the example is negative). Let $N' \leq N \cdot q(n)$ be the number of terms generated in this step. Denote these terms by $t_1, \ldots, t_{N'}$.
- *Eliminating terms:* For each term t_i ($1 \leq i \leq N'$) the learner asks for an estimation, within tolerance $\tau = \varepsilon/4N'$, of $p_i \overset{\triangle}{=} Pr_{x\in D}[t_i(x) = 1 \wedge f(x) = 0]$. If the answer is greater than $\varepsilon/4N'$ then it eliminates the term t_i.
- The hypothesis h is the disjunction of all remaining terms.

Clearly, the estimation of the probabilities p_i's falls into the statistical queries model by setting $\chi_i(x,l) = 1$ if and only if $t_i(x) = 1$ and $l = 0$. The evaluation of χ_i is polynomial, and therefore all the algorithm runs in polynomial time.

To evaluate the probability that $h(x) \neq f(x)$ we consider two types of mistakes. Either $h(x) = 1$ and $f(x) = 0$ or $h(x) = 0$ and $f(x) = 1$. For each x of the first type there exists (at least one) *bad* term t_i, for which $t_i(x) = 1$, that we added in the first step and failed to eliminate in the second step. For every bad t_i we have $p_i > 0$. Moreover, if t_i is not eliminated it must be the case that $p_i < \varepsilon/4N' + \tau = \varepsilon/4N' + \varepsilon/4N' = \varepsilon/2N'$ (otherwise the answer that we get to the query about p_i must be greater than $\varepsilon/4N'$ and the term will be eliminated). Therefore, since N' terms are generated in "collecting terms", there can be at most N' bad terms, and

$$Pr_{x\in D}[(h(x) = 1) \wedge (f(x) = 0)] \leq \sum_{\text{bad } t_i} \varepsilon/2N' \leq \varepsilon/2.$$

Each x of the second type is caused by a term t_i which appears in the function but is not found in the "collecting terms" step. We call a term t_i *important* if $Pr_{x\in D}[t_i(x) = 1] \geq \frac{\varepsilon}{2p(n)}$ (note that once a term which appears in the function is found, it cannot be eliminated as the corresponding probability p_i equals 0). First, we claim that if we find all the important terms in the "collecting terms" step (and therefore they all appear in h) then the total error of the second type is bounded by $\varepsilon/2$. This is because in this case

$$\begin{aligned}
&Pr_{x\in D}[(h(x) = 0) \wedge (f(x) = 1)] \\
&\leq \sum_{\text{non-important } t_i} Pr_{x\in D}[(h(x) = 0) \wedge (t_i(x) = 1)] \\
&\leq \sum_{\text{non-important } t_i} Pr_{x\in D}[t_i(x) = 1] \\
&\leq p(n) \cdot \frac{\varepsilon}{2p(n)} = \frac{\varepsilon}{2}.
\end{aligned}$$

Therefore, it is enough to show that the probability that all important terms are found is at least $1 - \delta$. Consider an important term t_i and a single example drawn in the "collecting terms" step. The probability that t_i is found by this example is at least the probability that this example is satisfied by t_i, times the probability that $\mathcal{B}$ will output *all* the terms satisfied by the example, which is therefore at least $\frac{\varepsilon}{2p(n)} \cdot \frac{1}{2} = \frac{\varepsilon}{4p(n)}$. Therefore, the probability that t_i will not be found using $\frac{4p(n)}{\varepsilon} \cdot \ln \frac{p(n)}{\delta}$ examples is at most

$$\left(1 - \frac{\varepsilon}{4p(n)}\right)^{\frac{4p(n)}{\varepsilon} \cdot \ln \frac{p(n)}{\delta}} \le \frac{\delta}{p(n)}.$$

Thus, after drawing $p(n) \cdot \frac{4p(n)}{\varepsilon} \cdot \ln \frac{p(n)}{\delta}$ examples the probability that we fail to find an important term is at most δ. Hence choosing N as in the algorithm suffices.

All together, we get that with probability at least $1 - \delta$ the algorithm finds a hypothesis h such that $Pr[h(x) \neq f(x)] \le \varepsilon$. This completes the proof of the theorem. ■

We note that in the proof we have used "statistical-queries" with tolerance $\tau = \frac{\epsilon^2}{16p^2(n)q(n)\ln\frac{p(n)}{\delta}}$.

5.3. *Malicious Noise*

In the model of PAC learning with malicious error ((Valiant, 1985, Kearns & Li, 1993)), when a learner sees an example, only with probability $1 - \beta$ it is drawn from the probability distribution D, and is labeled correctly according to the target concept. With probability β a malicious adversary may select any example and label it either positive or negative. Let $q(n)$ be an upper bound on the number of monomials produced by the algorithm $\mathcal{B}$ in Lemma 1 and $p(n)$ be an upper bound on the size of functions in $\mathcal{C}_{\mathcal{M}}$. We show:

THEOREM 8 *If $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable then $\mathcal{C}_{\mathcal{M}}$ is PAC learnable in the presence of malicious error of rate less than $\beta = \frac{\epsilon}{\ln\frac{1}{\varepsilon} \cdot 16p^2(n)q(n) \cdot \ln\frac{p(n)}{\delta}}$.*

Proof: By a result of Decatur (Decatur, 1993), the existence of a "statistical-queries" learning algorithm with tolerance τ for $\mathcal{C}$ implies that there exists a PAC learning algorithm for $\mathcal{C}$ that can tolerate malicious error rate $\Omega(\tau)$.

In the proof of Theorem 7 we present a statistical queries algorithm for $\mathcal{C}_{\mathcal{M}}$ with tolerance $\tau = \frac{\epsilon^2}{16p^2(n)q(n)\ln\frac{p(n)}{\delta}}$. It is shown in (Aslam & Decatur, 1993) that by using hypothesis boosting techniques, this tolerance can be made smaller. In particular, the ε^2 factor can be reduced to $\frac{\varepsilon}{ln\frac{1}{\varepsilon}}$. Putting those together yields the desirable error rate.

■

We have considered above classification noise, malicious noise and a type of attribute noise that is relevant to visual concepts. The known technique for handling (unrestricted) attribute noise in learning DNF formulae (Shackelford & Volper, 1988) works in cases where the noise-free algorithm uses at any point only a small number of attributes to update its hypothesis. In this way, with non-negligible probability, the noise-tolerant algorithm will get examples in which this small set of attributes is noise-free, and will learn using the noise-free algorithm. This technique was used to learn k-DNF in the presence of attribute noise (with a fixed error rate) (Shackelford & Volper, 1988). It is not hard to see that the same technique, coupled with the SQ algorithm presented in Section 5.2, can be used to learn a wider class of functions. In particular, we could learn the class of all functions which are disjunctions of polynomially many monomials from $\mathcal{M} \cap \mathcal{L}$, where $\mathcal{C}_{\mathcal{M}}$ is a polynomially explainable class corresponding to $\mathcal{M}$ and $\mathcal{L}$ is the set of all monomials of size at most $\log n$. This restriction guarantees that, when allowing attribute noise of up to $1/2$, by seeing n times more examples in the *collection step*, the algorithm receives, for every term, a noise-free example. Hence, the collection succeeds with high probability in spite of the attribute noise. Then, the *elimination step* uses the algorithm of (Shackelford & Volper, 1988).

6. Discussion

We present a new approach, *polynomial explainability*, to the problem of learning DNF formulae from examples and use it to learn some subclasses of DNF (and CNF) which were not known to be learnable before. As mentioned, polynomial-explainable classes discussed here contain the subclasses of k-DNF and k-term-DNF as special cases. It is natural to ask how these results relate to the learnability of k-DL (Rivest, 1987) (for some constant k), which was the widest subclass of boolean formulae known to be PAC-learnable from examples. We show that the results are incomparable. To see that polynomially explainable subclasses may contain functions which are not expressible by any k-DL (of any size), it is convenient to use the terminology of visual concepts. Let m be a parameter (say, $m = \sqrt{n}$). Consider the function f which is 1 if and only if the picture contains an $m \times m$ square all of its pixels are 1 (this can be expressed as a DNF with n^2 terms each of size m^2). Suppose that there exists a k-DL for this function f and consider its first node. In this node, k literals are examined and if they are all satisfied by the assignment (the picture) the function is evaluated to have some value $\sigma \in \{0, 1\}$. However, since $k \ll m$, no matter what k literals are examined, the picture can always be extended in two ways, one which contains an $m \times m$ square as required and one which does not. Hence, no matter what the value of σ is the decision list must err on some of the inputs.

On the other hand, to see that k-DL (even for $k = 1$) may sometimes be stronger than polynomially explainable classes, we look at the set of all monomials. This set is in 1-DL but is not learnable by the algorithm of Theorem 1, as each positive example can be “explained” by exponentially many monomials. This class is learnable however in terms of CNF (i.e., by Theorem 2). Similarly, the dual example, of all disjunctions, is also in 1-DL and is not learnable by Theorem 2 (but is learnable by Theorem 1). By combining these two examples (e.g., by considering the set of all functions of the type $x_{i_1} \vee \ldots \vee x_{i_m} \vee (y_{j_1} \wedge \ldots \wedge y_{j_p})$) we get a family of functions which is in 1-DL and is not learnable neither by the algorithm of Theorem 1 nor by the algorithm of Theorem 2 (as by fixing all the x’s to 0 we can get all the monomials on the y’s, and on the other hand, by fixing all the y’s to 0 we can get all the disjunctions on the x's). It may be the case, however, that these functions are still in some "polynomially explainable" class, using another representation (other than DNF or CNF).

We believe that the approach used in this paper will be found useful in tackling other problems as well. For example, Angluin (Angluin, 1980) considered the problem of learning *pattern languages*[8], where a pattern p is a string consists of bits ($\{0,1\}$) and variables, and its language, $L(p)$, is the set of all strings that can be obtained from p by substituting a string (in $\{0,1\}^*$) for each of the variables (for example, $L(x00xy)$ contains the string 1100110 which is obtained by substituting $x = 11, y = 0$). (Schapire, 1990) showed the hardness of learning *pattern languages*. (Kearns & Pitt, 1989) showed how to PAC-learn such languages, assuming that the number of variables is constant (though each variable may appear many times in the pattern) and with some limitations on the underlying distribution. Using the technique presented in this paper we can show how to learn (in the mistake-bound and also in the "statistical-queries" model) the language $L = L(p_1, p_2, \ldots, p_t) = \cup_{i=1}^{t} L(p_i)$ (i.e., L is the language of all strings that can be obtained from *any* of the patterns), where t is polynomial in n, and the total number of occurrences of variables in each pattern p_i is constant (note that the patterns here are more restricted than those considered in (Kearns & Pitt, 1989); however, we are not restricted to a single pattern but rather allow a collection of patterns). Loosely speaking, this is possible since given a positive example, we can enumerate the polynomially-many (in the length of the example) patterns of the above form from which the example can be obtained. To do so, given a length n string which is a positive example, we go over all the $O(n^{2c})$ possibilities to choose c (non-overlapping) substrings of it (we choose a substring by choosing a starting point and an end point). For each of these choices there are c^c possibilities to choose a variable name for each substring. Then, we enumerate only the patterns in which all the substrings that correspond to the same variable are the same (i.e., the substitution is consistent). As long as c is fixed, this gives a polynomial algorithm.

Jerrum (Jerrum, 1991) showed that learning *translation-invariant* DNF (when the learner is required to use the same representation as the output representation) is NP-hard. Note that the notion of a *translation-invariant* term and our notion of *dynamic* pattern are closely related; that is, the dynamic version of our visual learning problem is a subclass of translation-invariant DNF. In contrast to the general translation-invariant DNF, this subclass is still PAC-learnable.

In (Valiant, 1985) a hierarchical approach for learning DNF was discussed in which one learns a collection of monomials, in a supervised or unsupervised manner, and only then learns a DNF formula as a disjunction over this set of monomials. The method developed in this paper can be described as an instance of this approach, in which the set of "interesting" monomials, explaining the data seen, can be generated efficiently. It is interesting to note that in particular, all classes learnable within this framework (e.g. in the context of membership queries, (Blum, 1992, Blum & Rudich, 1992, Aizenstein & Pitt, 1992, Blum et al., 1994).) are shown here to be learnable even in the presence of noise.

Acknowledgments

We wish to thank Les Valiant for helpful discussions and comments. We thank the referees for helpful suggestions that improved the presentation of this work.

Eyal Kushilevitz was supported by research contracts ONR-N0001491-J-1981 and NSF-CCR-90-07677. Dan Roth was supported by NSF grant CCR-92-00884 and by DARPA AFOSR-F4962-92-J-0466.

Notes

1. We assume that each pixel can be either *black* or *white*. The results can be extended to handle more values in a straightforward way.
2. This should not be confused with the approach of *explanation based learning (EBL)* (see, e.g., (Mitchell, Keller, & Kedar-Cabelli, 1986)). In the EBL framework, a learner receives a single example, and tries to generalize it in a way that can be justified by deduction from the prior knowledge the learner has about the domain.
3. with a polynomial penalty in the complexity of the algorithms; This is because a pattern in the dynamic case can be replaced by a polynomial number (in the size of the picture) of patterns in the static case (e.g., $O(n^2)$ if only translations are considered and the size of the picture is $n \times n$).
4. Note that if $\mathcal{A}$ was guaranteed only to give a monomial that appears in *some* representation of f then this bound is false (as it could be the case that the "true" monomials in different executions of $\mathcal{A}$ belong to different representations of f). This explains the seemingly too strong requirement of the definition.
5. This class can be learned also in a different way, using the observation that every function in this class has a polynomial number of satisfying assignments.
6. This bound can be slightly improved to $(2n)^c$ using the axis-parallel property.
7. Alternatively, the shape can be the set of all those squares for which at least 50% of the area is contained in the polygon or any other way, consistent with the way digitization is made, namely, the method by which real-world pictures are converted to $n \times n$ pictures.
8. There is no relation between this notion of pattern and the notion of pattern used in our work.

References

Aizenstein, H. & L. Pitt. (1991). Exact learning of read-twice DNF formulas. In *Proceedings of the IEEE Symp. on Foundation of Computer Science*, number 32, pages 170–179, San Juan.

Aizenstein, H. & L. Pitt. (1992). Exact learning of read-k disjoint DNF and not-so-disjoint DNF. In *Proceedings of COLT '92*, pages 71–76.

Angluin, D. (1980). Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21(1):46–62.

Angluin, D., M. Frazier, & L. Pitt. (1992). Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164.

Angluin, D. & P. Laird. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Aslam, J. A. & S. E. Decatur. (1993). General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 282–291.

Basri, R. (1994). Private Communication.

Bender, M. & D. Roth. (1994). Learning human motion as DNF formulae. (Unpublished).

Blum, A. (1992). Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386.

Blum, A., R. Khardon, A. Kushilevitz, L. Pitt, & D. Roth. (1994). On learning read-k satisfy-j DNF. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory*, pages 110–117. (Submitted for publication).

Blum, A. & S. Rudich. (1992). Fast learning of k-term DNF formulas with queries. In *Proceedings of Twenty-Fourth ACM Symposium on Theory of Computing*, pages 382–389.

Blumer, A., A. Ehrenfeucht, D. Haussler, & M. K. Warmuth. (1987). Occam's razor. *Information Processing Letters*, 24:377–380.

Bshouty, N. H. (1993). Exact learning via the monotone theory. In *Proceedings of the IEEE Symp. on Foundation of Computer Science*, pages 302–311, Palo Alto, CA.

Decatur, S. E. (1993). Statistical queries and faulty PAC oracles. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*, pages 262–268.

Hancock, T. (1991). Learning 2μ DNF formulas and $k\mu$ decision trees. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 199–209.

Haussler, D., M. Kearns, N. Littlestone, & M. K. Warmuth. (1991). Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161.

Jackson, J. (1994). An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings of the IEEE Symp. on Foundation of Computer Science*. To Appear.

Jerrum, M. (1991). Simple translation-invariant concepts are hard to learn. Technical Report CSR-12-91, University of Edinburgh, Department of Computer Science.

Kearns, M. (1993). Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 392–401.

Kearns, M. & M. Li. (1993). Learning in the precence of malicious error. *Siam Journal of Computing*, 22(4).

Kearns, M., M. Li, L. Pitt, & L. G. Valiant. (1987). On the learnability of boolean formulae. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 285–295.

Kearns, M. & L. Pitt. (1989). A polynomial-time algorithm for learning k-variable pattern languages from examples. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 57–71.

Kushilevitz, E. & Y. Mansour. (1993). Learning decision trees using the fourier spectrum. *Siam Journal of Computing*, 22(6):1331–1348. Earlier version appeared in Proc. 23rd Ann. IEEE Symp. on Foundations of Computer Science, 1991.

Li, M. & P. M. B. Vitanyi. (1989). A theory of learning simple concepts under simple distributions and average case complexity for the universal distribution. In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pages 34–39.

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

Littlestone, N. (1989). *Mistake bounds and logarithmic linear-threshold learning algorithms*. Ph.D. thesis, U. C. Santa Cruz.

Mitchell, T.M., R.M. Keller, & S.T. Kedar-Cabelli. (1986). Explanation Based Learning. *Machine Learning*, 1(1):47–80.

Pillapakkamnatt, K. & V. Raghavan. (1993). Read twice DNF formulas are properly learnable. Technical Report TR-CS-93-59, Vanderbilt University, Computer Science Department. To appear, Proceedings of the 1st European Conference on Computational Learning Theory (EuroColt 93).

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, 2(3):229–246.

Schapire, R. E. (1990). Pattern languages are not learnable. In *Proceedings of COLT '90*, pages 122–129.

Shackelford, G. & D. Volper. (1988). Learning k-DNF with noise in the attributes. In *First Workshop on Computatinal Learning Theory*, pages 97–103.

Shvaytser, H. (1990). Learnable and nonlearnable visual concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):459–466.

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

Valiant, L. G. (1985). Learning disjunctions of conjunctions. In *Proceedings of the International Joint Conference of Artificial Intelligence*, pages 560–566.

Received November 23, 1993
Accepted August 25, 1994
Final Maunscript November 8, 1995