ELSEVIER

Theoretical Computer Science 230 (2000) 1–37

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

P-sufficient statistics for PAC learning *k*-term-DNF formulas through enumeration

B. Apolloni *, C. Gentile

Dipartimento di Scienze dell' Informazione, Università di Milano, Via Comelico 39-41, I-20135 Milan, Italy

Received June 1996; revised May 1998
Communicated by B. Rovan

Abstract

Working in the framework of PAC-learning theory, we present special statistics for accomplishing in polynomial time proper learning of DNF boolean formulas having a fixed number of monomials. Our statistics turn out to be *near* sufficient for a large family of distribution laws – that we call *butterfly distributions*. We develop a theory of *most powerful learning* for analyzing the performance of learning algorithms, with particular reference to trade-offs between power and computational costs. Focusing attention on sample and time complexity, we prove that our algorithm works as efficiently as the best algorithms existing in the literature – while the latter only take care of subclasses of our family of distributions. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Computational complexity; Boolean function; Concept learning; Learning by examples; Sufficient statistics; Nonparametric statistics

Contents

* Corresponding author. E-mail: apolloni@hermes.mc.dsi.unimi.it.

PII: S0304-3975(98)00215-1

1. Introduction

Intuitively, learning in the probably approximately correct (PAC) model [35] amounts to constructing a routine c, not yet available in our software library, on the sole basis of (a) a rough description of the routine functionality, and (b) a set of examples of how the routine has to behave. For instance, for an unknown boolean function c we are given:
(a1) a set of boolean variables $\{x_1, \ldots, x_n\}$ that are the arguments of c.
(a2) a vague description of c. Thus, in the case of *disjunctive normal forms* (DNF), c is assumed to be representable by a boolean formula which is a disjunction of k terms, each term being a conjunction of some of the above variables, or their negations.
(b) a set of input–output pairs of c that constitute examples of what c computes. Say, for $x_1 = 1, x_2 = 0, \ldots, x_n = 1$ c computes 0, for $x_1 = 1, x_2 = 1, \ldots, x_n = 1$ c computes 1, and so forth.

Our aim is to approximate c by a boolean function h that is *similar* to c, in the following sense: upon replacing c by h , the probability of getting a wrong output in the next computations is very small.

More formally, the main ingredients of a PAC learning procedure are as follows:

- a probability space $(\boldsymbol{X}, \mathscr{F}, P)$, where
  $\boldsymbol{X}$ is the set of the possible outcomes of a source of random data,
  $\mathscr{F}$ is a σ-algebra on $\boldsymbol{X}$,
  P is a (possibly unknown) probability measure defined over $\mathscr{F}$;
- a family $\boldsymbol{C}$ of subsets c of $\boldsymbol{X}$, belonging to $\mathscr{F}$. Every c is called a (*target*, *concept*) and $\boldsymbol{C}$ is a *concept class*;
- a *labelled* random *sample* $\boldsymbol{X}_m^c$ consisting, for each c, of pairs

$$(X_i, \chi_c(X_i)), \quad i = 1, \ldots, m,$$

called *examples*. Here $X_1, \ldots, X_m$ are randomly chosen elements of $\boldsymbol{X}$, and $\chi_c : \boldsymbol{X} \mapsto \{0, 1\}$ denotes the *indicator function* of c.

The task of learning amounts to the computation of a statistic on $\boldsymbol{X}_m^c$, which identifies a region h in $\mathscr{F}$ yielding a good estimate of (an accurate hypothesis on) c. The probability measure P_{error} of the symmetric difference $c \div h$ between c and h is assumed to be the *accuracy parameter* ε of the estimate. By a *learning algorithm* we mean an algorithm A that generates hypotheses whose error probability P_{error} converges to 0 in probability.

While previous approaches were aimed at asymptotically perfect learning [34] (corresponding to the particular case $\varepsilon = 0$), Valiant's approach [35] allows us to compute

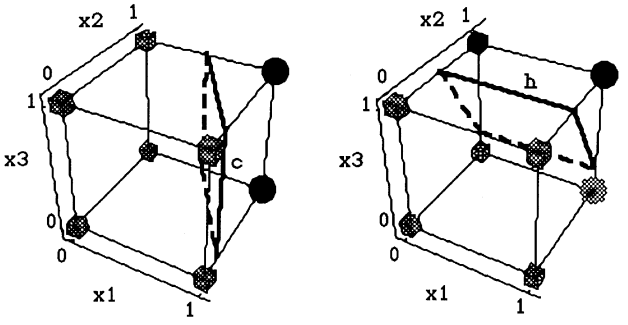

Fig. 1 *Learning the class* $\boldsymbol{C}$ of monomials having exactly two non-negated variables on the three-dimensional boolean cube.

$$c = x_1x_2;\ \boldsymbol{X}_m^c = \{((1,0,0),0);\ ((0,1,0),0);\ ((0,0,1),0);\ ((1,1,1),1);\ ((0,0,1),0);$$
$$((0,1,0),0);\ ((1,1,1),1);\ ((0,1,0),0);\ ((0,1,0),0);\ ((0,0,1),0)\}$$
$$h = x_2x_3;\ c \div h = \{(1,1,0);\ (0,1,1)\}$$

● = 1-labelled assignments = 0 -labelled assignments
○ =0 -mislabelled assignment; = 1-mislabelled assignment

sharp estimates of the sample size needed to achieve a preassigned accuracy ε and *confidence* δ as a function of certain combinatorial indices of the complexity of $\boldsymbol{C}$. By measuring the running time needed to compute the statistic h from the sample $\boldsymbol{X}_m^c$ we can make precise the notion of global feasibility of the learning task for $\boldsymbol{C}$.

In the example of Fig. 1, the black box point (0,1,1) and the grey sphere point (1,1,0) are *mislabelled* by the hypothesis $h = x_2x_3$. Let us equip the learning algorithm A with a fixed but otherwise arbitrary lexicographic order and stipulate that A outputs the first hypothesis within $\boldsymbol{C}$ that does not contradict the examples. The above *mislabelling* occurs because the points (0,1,1) and (1,1,0) are not members of the sample $\boldsymbol{X}_m^c$ used in learning the target concept $c = x_1x_2$. Thus the labels of (0,1,1) and (1,1,0) are unknown to A.

How big is our mistake in the above approximation? According to the strict philosophy of PAC learning, big mistakes are not frequent. Indeed our mislabelling only can have a big impact if the assignments (0,1,1) or (1,1,0) were likely to be used as an input of the unknown monomial. To fix ideas, assume c is a key classifier in a database, and the keys (0,1,1) and (1,1,0) never appear in the ten records randomly drawn from it. Then we may reasonably expect that (0,1,1) and (1,1,0) are rather infrequent keys in the database. If records are drawn with the same distribution law, the probability of further meeting (0,1,1) and (1,1,0) and consequently misclassifying them is small. Summing up, it is unlikely to draw a labelled sample that misses points of relevant probability, which yields a large probability P_{error}. In Section 2 we shall recall techniques allowing us to manage these probabilities.

As is well known, every boolean function has an equivalent DNF reduction. A DNF formula is probably the most common tool used by human beings for giving an immediate description of a set of objects by specifying a number of very simple alternative properties possessed by those objects. In particular, k-term-DNF formulas like in the example above might represent the class of all the realistic above descriptions, dealing with a preassigned number k of such alternatives. Thus, results on the learnability of these formulas have both computational and philosophical consequences.

Adopting the usual dichotomy *polynomial = feasible* vs. *non-polynomial = unfeasible*, it turns out that PAC learning of k-term-DNF is a very hard task: here the main difficulties are not due to the size of the labelled sample, but rather, to the problem of *computing* a hypothesis from a labelled sample. Let us parametrize our class by the number n of propositional variables. Then from a purely statistical point of view, sample size m will be sufficient, provided m polynomially depends on the relevant parameters n, $1/\varepsilon$ and $1/\delta$. However, unless NP $= R$ [5], the problem of computing an accurate hypothesis h from the given m examples is hard (in fact, NP-hard) for every preassigned $k \geqslant 2$ [27]. Stated otherwise, up to well-established conjectures on the status of NP-hard problems, no learning algorithm exists whose running time is polynomial in n, $1/\varepsilon$ and $1/\delta$.

These negative results hold when we are faced with the problem of representing hypotheses by k-term-DNF formulas (as is done in *proper learning*). One might decide to give up this natural way of describing the hypotheses looking for other representations, e.g., using less concise k-CNF formulas (conjunction of clauses with at most k propositional variables). Then more powerful descriptional tools become available – allowing polynomial-time learning algorithms [27]. However, this does not constitute a shortcut to proper learning: for, CNF-to-DNF conversion is typically unfeasible. The same happens if we assume hypotheses to be described by $\mathbf{O}(n^{k-1})$-term-DNF(n) formulas, or by decision lists (for more information see [9] and [28], respectively.)

Looking for proper learning results, researchers have made a great deal of effort towards computational feasibility by *reducing* the generality of the problem. This leads in a natural way to consider *special classes* of sample distributions, that are both well-shaped for designing efficient learning algorithms and sufficiently expressive to formalize actual learning problems of the real world.

As proved in [22], if no restrictions are made on sample distributions, then learning remains NP-hard, even assuming the concept class to consist of monotone formulas (formulas where the negation symbol never occurs), or else, μ-formulas (formulas where each propositional variable occurs at most once). On the other hand, k-term-μDNF formulas are properly and polynomially learnable when *positive* (1-labelled) examples, as well as *negative* (0-labelled) examples, separately follow the *uniform* distribution [22]. This is the so-called *two-distribution* model. Yet for uniform distributions, monotone $\mathbf{O}(\log(n))$-term-DNF formulas are properly and polynomially learnable by positive examples only [31], while interesting results dealing with non-proper learning algorithms for k-term- DNF formulas are given in [25, 39].

Naturally enough, the homogeneity and symmetry properties of the uniform distribution allow us to easily predict and control relevant parts of the target formula. As a result, certain terms can be disregarded during the computation of the final hypothesis. Moreover, constant probability spread over the whole sample space corresponds to maximum entropy distribution – the latter seldom applying in everyday operational contexts.

Therefore, various generalizations of the uniform distribution assumption have been proposed in the literature aimed at preserving the mentioned benefits through some sort of smoothness constraints. For instance, in the so-called *q-bounded* distributions, the ratio of the measure of any two points in the sample space is bounded by a quantity q – possibly different from 1. See [15, 30] for proper learnability results for k-term-DNF formula in this setting. Weaker results (for the purpose of our paper) are obtained in [19] for *product* distributions. In this case, homogeneity comes from the assumption that the coordinates of the sample space points are stochastically independent. However, it should be remarked that learning in this context is not proper.

All distributions mentioned so far are *p-smooth*, in the sense that, for any pair of points of unit Hamming distance, the ratio of their measures is suitably bounded. The algorithm proposed for these distributions in [32] properly PAC learns F_k-term formulas. By definition, F_k-term formulas constitute the class formed by arbitrary boolean functions of k many monomials; hence, in the subfamily of k-term-DNF formulas one further assumes that monomials are connected by disjunction. These formulas cannot be properly learnt by the algorithm of [32] in its basic version. Moreover, two separate polynomially sized samples of positive and negative examples are required.

Also the present paper pursues the aim of extending the family of treatable distributions for learning k-term-DNF formulas. As we shall see, our extension includes p-smooth distributions, when we work within the conventional PAC framework requiring the learning algorithm to compute hypotheses with arbitrarily small accuracy ε and confidence δ in feasible time. If we relax the PAC learnability requirements by binding ε and δ to be not smaller than the inverse of any polynomial in the number n of boolean attributes, then we are also able to include families of uniform distributions on *subsets* of the sample space. This weaker version of learnability looks reasonable in many operational contexts when the overall running time is bounded by a polynomial in n anyway.

The distinguishing feature of the family of *butterfly* distributions proposed in this paper is the following. Let us imagine the 2^n different assignments to the variables $x_1, \ldots, x_n$ in the sample space $\boldsymbol{X} = \{0,1\}^n$ as a stack of elastic wires. Let us squeeze the stack in correspondence with some of those variables, previously grouped in the center of the wires. We obtain a butterfly-shaped picture representing the support of the monomial specified by such variables. Let us associate with the butterfly the probability measure of this support. By squeezing the stack in correspondence with further variables we obtain a new smaller butterfly, whose probability measure does not have any abrupt jumps: for, the ratio between the probability measures of the two butterflies is suitably bounded. In addition, if the measure of the first butterfly is negligible, no assumptions

are made on this ratio. This bound, along with the number of additional squeezed positions parameterize the members of the butterfly family.

Using this family of distributions we are able to hit the target of performing proper PAC learning of k-term-DNF formulas in polynomial time for a constant $k \geqslant 1$. Our algorithm uses labelled examples from a unique sample space.

Summing up ideas from [1, 8], and arguing along the lines of [4], we can describe our main idea as follows:

Let c be an unknown formula that we want to learn within a given class $\boldsymbol{C}$. We first enumerate all formulas of $\boldsymbol{C}$. We then draw a set of examples of how c behaves on random inputs and, for each listed formula, we use this set to test if the formula can be considered an accurate estimate of c in the PAC model. This procedure provides a learning algorithm if, eventually, precisely one formula is accepted as a hypothesis. Further, we say that the procedure is feasible if the cardinality of the enumerated list is polynomial. This second condition will not be satisfied by the problem considered in this paper. Therefore we resort to enumerating *discriminant* subsets of the sample space, possibly containing the supports of the monomials of c. In order to fulfill the first condition we draw some special kind of sufficient statistics. In this respect, the nice properties of our butterfly distributions, are to the effect that the frequencies with which the examples fall into the mentioned discriminating subsets *nearly* constitute (in a technical sense which will be made precise) a sufficient statistic for the distribution law of the probability error P_{error}. For stressing the fact that a polynomial number of them is sufficient for accomplishing the whole learning procedure, we call these statistics *p-sufficient*.

In this probabilistic framework we design a polynomial time *statistical query* (SQ) [20] algorithm that translates into a PAC algorithm requiring polynomial resources too. Specifically, we develop two versions of the algorithm, respectively dealing with monotone and non-monotone formulas. Both versions use polynomial resources for a suitable range of the parameters of the butterfly family. These resource bounds are preserved under various kinds of errors, possibly occurring in the labelled sample.

Our bounds are as good as those proven for other algorithms that were developed for proper learning our present target class under some *near*-uniform distributions. An exception is given by [31] only in regards of time complexity. However, our algorithm still requires polynomial resources when applied to further useful distribution laws.

To make a comparison between the statistical performance of our algorithm and the ideal fully enumerative learning procedure, we extend to learning statistics results of point estimation theory. We establish the uniqueness of optimal learning procedure (in the mean square error sense), and we discuss possible trade-offs between optimality and feasibility of learning algorithms.

Our paper is organized as follows: in Section 2 we define our learning problem; after introducing all relevant background material concerning the SQ model, we formalize the notion of relaxed PAC learning and sketch a theory on the most powerful

learning algorithms. In Section 3 we describe the class of distribution laws and recall some results from discriminant theory. Our algorithm is presented in Section 4, together with an analysis of its performance. We collect a few concluding remarks in Section 5.

2. Preliminaries

Let us start from the formal definition of PAC learning. To complete the formal framework previously introduced, we must describe some representation issues.

A distinguishing feature of the PAC model is the evaluation of the learning task in terms of its computational complexity. For this purpose it is useful to parameterize learnability results both by (1) the dimension of the domain and (2) some natural descriptional complexity of the current target concept:

(1) We think of $\boldsymbol{X}$ as a parameterized family of domains $\boldsymbol{X} = \bigcup_n \boldsymbol{X}_n$, where, for instance, $\boldsymbol{X}_n = \mathbb{R}^n$, the n-dimensional Euclidean space, or $\boldsymbol{X}_n = \{0,1\}^n$, the n-dimensional boolean hypercube. In such cases n can be roughly interpreted as the description length of the examples which are supplied to the learning algorithm.

(2) According to the previous point, we set $\boldsymbol{C} = \bigcup_n \boldsymbol{C}_n$, where $\boldsymbol{C}_n$ is a concept class over $\boldsymbol{X}_n$. We assume that the concepts of $\boldsymbol{C}$ are described in terms of a *representation* $\boldsymbol{R} = \langle R, L\rangle$ consisting of a set of strings L and a mapping R from $\boldsymbol{C}$ to 2^L that associates each concept c of $\boldsymbol{C}$ with a set of strings in L (i.e., with a set of different ways of representing c through $\boldsymbol{R}$). We denote by $\mathscr{L}_R(c)$ the length of the shortest string in L representing c. We simply write $\mathscr{L}(c)$ when $\boldsymbol{R}$ is clear from the context.

Here we assume that the domain points $X \in \boldsymbol{X}$ and the concepts $c \in \boldsymbol{C}$ are efficiently encoded with some standard scheme (e.g., [17]). For instance , for boolean functions n is the number of boolean arguments of c and $\mathscr{L}(c)$ can be assumed to be the circuit complexity (see, e.g., [40]) of c. The specific representation conventions we adopt for the learning problem considered in this paper are described in Section 2.1. Throughout the present paper n will be considered as a *known* parameter of the learning problem.

The pair of indices n and $\mathscr{L}_{\boldsymbol{R}}(c)$ is used in general to define learning according to the following definition.

Definition 1 (*Valiant* [35]; *Blumer et al.* [10]). Let us consider a concept class $\boldsymbol{C} = \bigcup_n \boldsymbol{C}_n$ on an n-indexed probability space $(\boldsymbol{X}_n, \mathscr{F}_n, P_n)$, where P_n belongs to a family $\boldsymbol{P}_n$ of probability measures on $\mathscr{F}_n$, possibly the family of *all* the probability measures on $\mathscr{F}_n$. Let $\boldsymbol{R} = \langle R, L\rangle$ be a representation of $\boldsymbol{C}$ and $\{\boldsymbol{X}_m^c\}$ be the set of *labelled samples of size m* for a given $c \in \boldsymbol{C}$. Let $P_n^{(m)}$ be the measure on the m-fold product probability space $(\boldsymbol{X}_n^{(m)}, \mathscr{F}_n^{(m)}, P_n^{(m)})$. A *PAC learning algorithm* A is a function $\mathsf{A} : \{\boldsymbol{X}_m^c\} \to L$ such that for every $0 < \varepsilon, \delta \leqslant 1$ there is a non-negative integer m° such that for every $c \in \boldsymbol{C}_n$ and $P \in \boldsymbol{P}_n$, for every sample $m \geqslant m^\circ$, denoting $h = \mathsf{A}(\boldsymbol{X}_m^c)$, the probability $P_{\text{error}} = P_n(\chi_c(X) \neq \chi_h(X))$ is bounded by: $P_n^{(m)}(P_{\text{error}} < \varepsilon) > 1 - \delta$.

If the above-mentioned function exists the class $\boldsymbol{C}$ is said to be *PAC learnable* (*learnable*, for brevity) w.r.t. $\boldsymbol{P}$ *using* $\boldsymbol{R}$. If m° is polynomial in $1/\varepsilon$, $1/\delta$ and n we say that the *sample complexity* of A is polynomial. If A has running time polynomial in the same arguments plus $\mathscr{L}(c)$, then $\boldsymbol{C}$ is called *polynomially learnable* and A is said to be a *polynomial time learning* algorithm for $\boldsymbol{C}$ w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$.

In the special case where $\boldsymbol{P}$ is the family of *all* the probability measures on $\mathscr{F}$ it is customary to call A a *distribution-free* learning algorithm for $\boldsymbol{C}$.

For brevity, from now the n-indexing of $\boldsymbol{C}$ and the probability space $(\boldsymbol{X}, \mathscr{F}, P)$ will be understood. In what follows, we will consider such representations that every string of L represents a concept of $\boldsymbol{C}$ (*proper* learning). In this case basic theorems [3, 11] state that, in order to learn, all that A needs is to observe a sufficient number of labelled examples of the goal function and output a contradiction-free hypothesis. These theorems introduce the notion of consistent hypothesis, namely a hypothesis that gives the examples the same labels of the target concept. Moreover, these theorems yield a very narrow gap between the lower and upper bound on the number of examples that have to be observed for a distribution-free learning of a class $\boldsymbol{C}$. These bounds depend on the following well-known combinatorial complexity index of $\boldsymbol{C}$.

Definition 2 (*Vapnik* [37]). Let $\boldsymbol{C}$ be a concept class and S a finite subset of $\boldsymbol{X}$, $\Pi_{\boldsymbol{C}}(S) = \{S \cap c | c \in \boldsymbol{C}\}$, and $\#\Pi_{\boldsymbol{C}}(S)$ its cardinality. The *Vapnik–Chervonenkis dimension* of $\boldsymbol{C}$ (shortly, $d_{\mathrm{VC}}(\boldsymbol{C})$) is the largest integer d such that $\max\{\#\Pi_{\boldsymbol{C}}(S) \,|\, \#S = d\} = 2^d$. If no such d exists, then $d_{\mathrm{VC}}(\boldsymbol{C})$ is assumed to be infinite. [1]

The next theorem deals with sample complexity bounds for the subfamily of *well-behaved* concept classes. This is a mild additional measurability property of a concept class introduced by Shai Ben-David (see [11]). As a matter of fact, almost all concept classes of concrete interest, including the one concerned in this paper, enjoy this property.

Theorem 1. *For a well-behaved concept class* $\boldsymbol{C}$ *on a probability space* $(\boldsymbol{X}, \mathscr{F}, P)$ *and a labelled sample* $\boldsymbol{X}_m^c$ *for* $c \in \boldsymbol{C}$, *let a* consistent hypothesis *be an* $h \in \boldsymbol{C}$ *such that* $\chi_c(X) = \chi_h(X)$ *for each element* X *of the sample. Then, for every* (*possibly unknown*) *probability measure* P *on* $\mathscr{F}$, *for* $0 < \varepsilon, \delta < 1/2$, *if* $m \geqslant max\{4/\varepsilon \log(2/\delta), 5.5 d_{\mathrm{VC}}(\boldsymbol{C})/\varepsilon\}$ *any function* $\mathsf{A} : \{\boldsymbol{X}_m^c\} \to \boldsymbol{C}$ *that outputs a consistent hypothesis is a distribution-free learning algorithm for* $\boldsymbol{C}$ [3].

For $0 < \varepsilon \leqslant 1/8$, $0 < \delta < 1/100$, *if* $d_{\mathrm{VC}}(\boldsymbol{C}) \geqslant 2$ *and* $m < \max\{((1-\varepsilon)/\varepsilon)\log(1/\delta), (d_{\mathrm{VC}}(\boldsymbol{C}) - 1)/32\varepsilon\}$ *then there exists a probability measure* P *on* $\boldsymbol{X}$ *such that no function of* $\boldsymbol{X}_m^c$ *is a learning algorithm for* $\boldsymbol{C}$ [14].

As shown by Vapnik and Chervonenkis in their pioneering paper [38], we can divide the learning job into two subtasks: an algorithmic one aimed at discovering a

[1] Since we parameterize a concept class $\boldsymbol{C} = \bigcup_n \boldsymbol{C}_n$ by n, the reader should consider $d_{\mathrm{VC}}(C)$ as the function of n $d_{\mathrm{VC}}(\boldsymbol{C}_n)$.

consistent hypothesis and a statistic one aimed at inferring the probability measure of the symmetric difference $c \div h$ from the fact that the frequency with which the sampled points fall there is zero.

The SQ model introduced by Kearns [20] arises quite naturally from this viewpoint. Here the learning algorithm uses the statistical properties of the labelled samples directly. In this sense the SQ model introduces a metaphor of the oracle STAT(c,P) with which the learning algorithm interacts by asking queries of the form $\langle \psi, \alpha \rangle$, where ψ is a function from $\{(X, \chi_c(X))\}$ to $\{0,1\}$ and $\alpha \in [0,1]$ is an accuracy parameter. The query $\langle \psi, \alpha \rangle$ is intended as a request for the value $P_\psi = P(\psi(X, \chi_c(X)) = 1)$. In other words, we assume that some property ψ of the random variable $(X, \chi_c(X))$ is relevant to our learning task and we check the probability P_ψ. From a labelled sample we might obtain a confidence interval for P_ψ. In its place STAT(c,P) returns an approximation P'_ψ such that $|P_\psi - P'_\psi| < \alpha$ with certainty. Consequently, the SQ model looks for an algorithm that assures $\mathrm{P}(c \div h) < \varepsilon$. The model is formalized in the following definition.

Definition 3 (*Kearns* [20]). Let $\boldsymbol{C}$ be a concept class on $(\boldsymbol{X}, \mathscr{F}, P)$, $P \in \boldsymbol{P}$, represented by $\boldsymbol{R} = \langle R, L \rangle$. $\boldsymbol{C}$ is polynomially learnable from statistical queries w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$ if there exists an algorithm A and polynomials $p(,,)$, $q(,,)$ and $r(,,)$ such that for every $c \in \boldsymbol{C}$, $P \in \boldsymbol{P}$, and $0 < \varepsilon \leqslant 1$ the following holds:

If A is given inputs ε, n and $\mathscr{L}(c)$ and access to STAT(c,P) then

(1) for every query $\langle \psi, \alpha \rangle$ made by A, ψ can be evaluated in time $q(1/\varepsilon, n, \mathscr{L}(c))$ and $1/\alpha$ is bounded by $r(1/\varepsilon, n, \mathscr{L}(c))$;

(2) A has running time bounded by $p(1/\varepsilon, n, \mathscr{L}(c))$ and outputs a hypothesis h, represented through $\boldsymbol{R}$, such that $P(c \div h) < \varepsilon$.

It is not surprising that almost every concept class that is polynomially PAC learnable is also SQ learnable. However, one can exhibit such learning tasks that show that the last model is less powerful than the PAC model [20].

An SQ learning algorithm translates into a PAC learning algorithm that is robust to noisy examples. This, of course, gives rise to larger deviations of the estimators we employ. We consider two kinds of errors.

(a) *Classification noise* (Angluin and Laird [2]). Here we assume that the indicator function of c changes into a Bernoullian variable of parameter $\chi_c(X)(1-\eta) + (1 - \chi_c(X))\eta$, with $0 \leqslant \eta < 1/2$, for each X independently from the other items in the sample. In other words, we might imagine that the sampled point $(X, \chi_c(X))$ is sent to us by means of an unreliable messenger who tosses a coin with bias η and flips the label of X whenever it turns heads.

This labelling error is called *classification noise*. Every concept class that is PAC learnable is also learnable with any fixed amount η of classification noise less than the information theoretic limit $= \frac{1}{2}$ [2]. A polynomial dependency on $(1 - 2\eta_b)^{-1}$ is added to the sample complexity [2, 21], where η_b is a known upper bound on η.

(b) *Malicious errors* (Kearns [21]; Valiant [36]). There are two changes with respect to the previous model. When tossed heads, the messenger:

- can corrupt the values of both X and $\chi_c(X)$ in the sampled item;
- can behave as a *malicious adversary*. He can decide how to modify X and whether or not to flip $\chi_c(X)$, according to a strategy based on the knowledge of ε, δ, n, c, P and the internal state of A.

The best upper bound on the error rate η (the coin bias, as above) that preserves learnability equals $\varepsilon/(1+\varepsilon)$ [21, 2], for minimal assumptions on the concept class (also satisfied by k-term-DNF formulas).

Focusing on oracles with finite query repertory, the robustness of the SQ model vs. these errors is stated by the following theorems.[2]

Theorem 2 (Kearns [20]). *Let $\boldsymbol{C}$ be a concept class on $(\boldsymbol{X}, \mathscr{F}, P)$, $P \in \boldsymbol{P}$, represented by $\boldsymbol{R}$. If $\boldsymbol{C}$ is polynomially learnable w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$ in the SQ model by algorithm A, then $\boldsymbol{C}$ is polynomially PAC learnable w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$ with classification noise rate $\eta \leqslant \eta_b < \frac{1}{2}$. In greater detail, let us call* query space *Q the set of queries ψ that can be asked the oracle and α a* lower *bound on the approximation error requested by A on the asked queries. Then for finite Q there is an algorithm that learns $\boldsymbol{C}$ in polynomial time w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$, whose sample complexity is $\mathbf{O}(1/(\alpha(1-2\eta_b))^2 \log(\#Q/\delta) + 1/\varepsilon^2 \log(1/(\delta\alpha(1-2\eta_b))))$.*

Theorem 3 (Decatur [13]). *Let $\boldsymbol{C}$ be a concept class on $(\boldsymbol{X}, \mathscr{F}, P)$, $P \in \boldsymbol{P}$, represented by $\boldsymbol{R}$. If $\boldsymbol{C}$ is polynomially learnable w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$ in the SQ model, then $\boldsymbol{C}$ is polynomially PAC learnable w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$ with malicious error rate $\eta = \gamma\alpha$, where $0 \leqslant \gamma < 1$ and α is defined as in Theorem 2. In more detail, for a finite query space Q, there exists an algorithm that learns $\boldsymbol{C}$ in polynomial time w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$, whose sample complexity is $\mathbf{O}(1/(\alpha-\eta)^2 \log(\#Q/\delta))$;*

In order to cover a larger set of learning problems occurring in practice, in the following definition we provide a weaker version of learnability. Owing to the asymptotic character of the definition, for definiteness, we shall explicitly mention the n-index.

Definition 1′. Let us consider a concept class $\boldsymbol{C} = \bigcup_n \boldsymbol{C}_n$ on an n-indexed probability space $(\boldsymbol{X}_n, \mathscr{F}_n, P_n)$, where $P_n \in \boldsymbol{P}_n$, a family of probability measures on $\mathscr{F}_n$. Let $\boldsymbol{R} = \langle R, L\rangle$ be a representation for $\boldsymbol{C}$. A *poly-relaxed* (PAC) *learning algorithm* A is a function $\mathsf{A}: \{\boldsymbol{X}_m^c\} \to L$ such that *for every* polynomial $p(.)$ and $1 \geqslant \varepsilon, \delta \geqslant 1/p(n)$, there is a non-negative integer m° such that for every $c \in \boldsymbol{C}_n$ and $P_n \in \boldsymbol{P}n$, for every sample size $m \geqslant m^\circ$, denoting $h = \mathsf{A}(\boldsymbol{X}_m^c)$, the probability $P_{\text{error}} = P_n(\chi_c(X) \neq \chi_h(X))$ is bounded by: $P_n^{(m)}(P_{\text{error}} < \varepsilon) > 1-\delta$, *for n large enough.*

[2] In later sections we will make use of Theorems 2 and 3 in the form quoted in the main text. As a matter of fact, they stay true even for infinite Q, once we use $d_{\text{VC}}(Q)$ instead of $\log \#Q$ in the sample complexity bounds. See [20] for details.

If m° is polynomial in n we say that the *sample complexity* of A is polynomial. If A has running time polynomial in n and $\mathscr{L}(c)$, then we say that $\boldsymbol{C}$ is *polynomially poly-relaxedly* (*PAC*) *learnable* w.r.t. $\boldsymbol{P}$ *using* $\boldsymbol{R}$ and we say that A is a *polynomial time poly-relaxed* (*PAC*) *learning* algorithm for $\boldsymbol{C}$ w.r.t. $\boldsymbol{P}$ using $\boldsymbol{R}$.

The reader can easily restate Definition 3 and Theorems 2 and 3 in terms of this relaxed notion of learnability, just letting ε and δ be the inverse of any polynomial and absorbing the polynomiality in $1/\varepsilon$ and $1/\delta$ into the polynomiality in n.

Of course, any polynomial time learning algorithm for $\boldsymbol{C}$ w.r.t. $\boldsymbol{P} = \cup_n \boldsymbol{P}_n$ is also a polynomial time poly-relaxed learning algorithm for $\boldsymbol{C}$ w.r.t. $\boldsymbol{P}$. In Section 3 we will point out interesting cases where only this relaxed notion of learnability takes place.

2.1. The learning problem

First of all, let us introduce some notation. Given a set of n propositional variables $V = \{x_1, \ldots, x_n\}$, a *literal* l_j is a non-negated or a negated propositional variable. We denote by x_i^c and l_j^c the negated variable x_i and literal l_j, respectively.

A *monomial* (or *term*) $\mu = \bigwedge_{l_i \in set(\mu)} l_i$ is the conjunction of a set set(μ) of literals. Both $a \wedge b$ and ab denote the conjunction of two literals or two terms a and b. An assignment of values $\boldsymbol{v} = (v_1, \ldots, v_n)$ to V satisfies μ if for each l_i in μ either $l_i = x_i$ and $v_i = 1$ or $l_i = x_i^c$ and $v_i = 0$.

A k-term-DNF(n) formula c is the disjunction of at most k terms on V. The parameterized class k-term-DNF is defined as k-term-DNF $= \bigcup_n k$-term-DNF(n). We will usually denote terms by t_i and their union either as $c = t_1 + t_2 + \cdots + t_r$ or as $c = t_1 \cup t_2 \cup \cdots \cup t_r$ or as $\{t_1, t_2, \ldots, t_r\}$, depending on various algebraic and algorithmic interpretations of the formula. c is satisfied by $\boldsymbol{v}$ if at least one monomial is satisfied by $\boldsymbol{v}$. c is *monotone* when no literal represents a negated variable.

For a given probability distribution P on $\boldsymbol{X}_n = \{0,1\}^n$, and a subset $B \subseteq \boldsymbol{X}_n$, P($B$) denotes the measure of B w.r.t. P. For subsets $B_1, B_2 \subseteq \boldsymbol{X}_n$ we denote the measure $P(B_1 \cap B_2)$ by $P(B_1, B_2)$.

For any set of literals $L = \{l_1, \ldots, l_k\}$ we define $\bar{\mu}_L = \bigwedge_{l_i \in L} l_i^c$; for any set $B \subseteq \boldsymbol{X}_n$ we denote by $B_{\downarrow L}$ the set of points of B satisfying all the literals of L. Moreover, we call *restriction* of B on L any set $B_{\downarrow M}$ with $M \subseteq L$. We denote that a set B' is such a restriction by $B' \leqslant_L B$.

When no ambiguity arises, we will abuse these notations by identifying a boolean formula with its support, i.e., with the set of assignments of values $\boldsymbol{v}$ on which the output is 1. Thus:

- $\boldsymbol{v} \in \mu$ means that $\boldsymbol{v}$ satisfies μ;
- $P(c)$ is the probability of the set of assignments making c true;
- $\mu_{\downarrow L}$ is the set of assignments satisfying both μ and all the literals of L;
- $\mu' \leqslant_L \mu$ means $\mu' = \mu \wedge (\bigwedge_{l_i \in M} l_i)$, for some $M \subseteq L$.

We want to learn the class $\boldsymbol{C}$ of k-term-DNF formulas through a representation $\boldsymbol{R}$ that associates with each concept c the string representing c in the above settled

notation. We will exhibit an algorithm A that outputs hypotheses h according to Definition 1 (Definition 1′) for each n (n large enough) and a constant $k \geqslant 1$ using small samples and short running times. For this algorithm we prove that its sample and time complexities are polynomial functions of n, $1/\varepsilon$ and $1/\delta$ (polynomial functions of n). In fact $\mathscr{L}(c) = \mathbf{O}(n)$ in the adopted representation; therefore, its contribution to the task complexity disappears.

As mentioned in the introduction, by virtue of Theorem 1 learning this class requires a polynomial sample size, since $d_{\mathrm{VC}}(k\text{-term-DNF}(n)) = \mathbf{O}(n)$. But the time needed to distribution-free learn this class is presumably non-polynomial (under standard complexity-theoretic assumptions), as shown in [27]. Therefore to make the job feasible we come to learn w.r.t. a special, although wide, family of probability distributions on $\boldsymbol{X}$. This allows us to find special statistics that: (i) are quickly computable and (ii) are highly efficient when the sample comes from the selected family of distributions. We will use these statistics as building blocks of the learning algorithm.

For a given $c \in \boldsymbol{C}$, the output of this algorithm is not a quantitative variable but a domain h of the σ-algebra $\mathscr{F}$, which satisfies a constraint on the probability measure $P(c \div h)$. Searching for this domain within a given class is essentially a computational task, where results from Computer Science can help. Checking that the probability constraint is satisfied is a statistical inference problem. Its main formal step is the determination of the minimum sample size that guarantees a fixed approximation threshold on the estimate of $P(c \div h)$ with a preassigned confidence. Now, this size depends on the employed estimator which, in turn, identifies the class of domains within which we look for our hypothesis. The last job may be feasible or not, depending on the complexity of this class. The trade-off between sample size and computational complexity which naturally arises can be outlined as follows.

In principle, for a given c, we may plan to enumerate all the items h of $\boldsymbol{C}$ and consider the related symmetric differences $c \div h$. For each of these domains we have that the frequency φ with which the sample items fall into them is a *Uniformly Minimum Variance Unbiased Estimator* (UMVUE) of $P(c \div h)$. Therefore for any fixed h, $\varphi = 0$ is a good critic region for testing the statistical hypothesis $P(c \div h) < \varepsilon$ (i.e., h is a good approximation of c) against the complementary hypothesis. Then we expect that a procedure based on this test, for example "pick the first h with $\phi = 0$", is extremely cheap w.r.t. the sample size, still remaining in the scope of Theorem 1. Nevertheless this procedure has high computational costs because the number of concepts in $\boldsymbol{C}$ is large.

Our short-cut consists in grouping h′s through minimal sufficient statistics [24]. This allows us to state lower bounds for the distribution law of $P(c \div h)$ within the group. The number of these statistics as well as the groups of hypotheses is *polynomial*, thanks to the clustering functionality of the *discriminants* (Section 3.2) that are the arguments of these statistics. The prefix 'p' in the name *p-sufficient statistics* we gave them stems from this feature.

In order to appreciate the quality of the resulting algorithm, below we state some lemmas and theorems concerning certain quality indices, mainly connected to the capacity of drawing the information content of the sample.

2.2. A side theory for the most powerful learning algorithms

Let us start by formally specifying some definitions and lemmas from Statistics Theory [29].

Definition 4. Let Y be a random variable of density $f(y) \in \Phi(y)$, and let the sample $Y_1, \ldots, Y_m$ be drawn from Y. A set of statistics $\{S_i = g_i(Y_1, \ldots, Y_m), i = 1, \ldots, n\}$ is a set of *joint sufficient statistics* if and only if the conditional distribution $f_{Y_1,\ldots,Y_m|S_1,\ldots,S_n}(y_1, \ldots, y_m|s_1, \ldots, s_n)$ of $Y_1, \ldots, Y_m$ given that $S_i = s_i$, i=1,...,n, does not depend on which $f(y)$ within $\Phi(y)$ generated the sample.

Definition 5. Let Y be a random variable of density $f(y) \in \Phi(y)$, and let the sample $Y_1, \ldots, Y_m$ be drawn from Y. Let $Z = g(Y_1, \ldots, Y_m)$ be a given function. A set of statistics $\{S_i = g_i(Y_1, \ldots, Y_m), i = 1, \ldots, n\}$ is a set of *joint sufficient statistics for* Z if and only if for any z the conditional distribution $f_{Z|S_1,\ldots,S_n}(z|s_1, \ldots, s_n)$ of Z given that $S_i = s_i$, $i = 1, \ldots, n$, does not depend on which $f(y)$ within $\Phi(y)$ generated the sample.

Definition 6. A set of joint sufficient statistics $S_1, \ldots, S_n$ (for Z) is sufficient and *minimal* if it is a function of any other set of joint sufficient statistics (for Z).

Definition 7. A set of joint sufficient statistics (for Z) is *complete* if and only if the expected value $\mathrm{E}(h(S_1, \ldots, S_n))$ of any function $h(.)$ equals zero $\forall f(y) \in \Phi(y)$, only if $P(h(S_1, \ldots, S_n) = 0) = 1 \ \forall f(y) \in \Phi(y)$.

Focusing on the variance of the estimator of $P(c \div h)$ as a relevant quality parameter of a learning algorithm, we define a UMVULP as follows:

Definition 8. Given a concept class $\boldsymbol{C}$ on $(\boldsymbol{X}, \mathscr{F}, P)$, a *learning procedure* $\mathscr{A}$ is a *family of statistics* on $\{\boldsymbol{X}_m^c\}$. $\mathscr{A}$ is an *unbiased* procedure if for each $c \in \boldsymbol{C}$ and $t \in (0,1)$ there exists an $m(t)$ such that for each $h \in \mathscr{A}((X_1, \chi_c(X_1)), \ldots, (X_{m(t)}, \chi_c(X_{m(t)})))$ we have $\mathrm{E}(P(c \div h)) < t$. $\mathscr{A}$ is a *Uniform Minimum Variance Unbiased Learning Procedure* (UMVULP) if: (i) $\mathscr{A}$ is unbiased and (ii) for each $\mathscr{A}'$ and each $h' \in \mathscr{A}'((X_1, \chi_c(X_1)), \ldots, (X_m, \chi_c(X_m))), m \leqslant m(t)$ such that $E(P(c \div h)) = E(P(c \div h'))$, we have $\mathrm{V}(P(c \div h)) \leqslant V(P(c \div h'))$, where V(.) denotes the variance.

Theorem 4. *Let* $\boldsymbol{C}$ *be a concept class on* $(\boldsymbol{X}, \mathscr{F}, P)$, *with* P *from the class of all absolutely continuous or all purely discrete distribution functions on* $\boldsymbol{X}$.

If $d_{\mathrm{VC}}(\boldsymbol{C}) < \infty$ *then there exists a unique learning procedure for* $\boldsymbol{C}$ *with output* $\{h\} \subseteq \boldsymbol{C}$, *which is symmetric in its arguments and unbiased for every* P *within the above class. This procedure is the unique UMVULP.*

Proof. In [16, 23] it has been shown that in the probabilistic space $(\boldsymbol{Y}, \mathscr{F}', P')$ with the same properties, any function $G(Y)$ that has an unbiased estimator independent of P' has a unique estimator that is unbiased and symmetric in the sample items. This estimator has uniform minimum variance among all unbiased estimators, i.e., it is symmetric and UMVUE. These results directly apply to the probabilistic space whose elements are the pairs $(X, \chi_c(X))$, when we want to estimate $\mathrm{E}(P(c \div h))$ through $P(c \div h)$, where P is a function that incidentally coincides with the unknown distribution law of X.

Indeed, $P(c \div h)$ is symmetric in the sample items if and only if h is such. Moreover, by Theorem 1, the finiteness of $d_{\mathrm{vc}}(\boldsymbol{C})$ implies the existence of a learning procedure $\mathscr{A}$ and a function $m(t)$ such that $h \in \mathscr{A}((X_1), \chi_c(X_1)), \ldots, (X_{m(t)}, \chi_c(X_{m(t)}))$ is symmetric and $\mathrm{E}(P(c \div h)) < t$, for all $t \in (0,1)$. Thus $\mathscr{A}$ is unbiased and $P(c \div h)$ is a symmetric and unbiased estimator of $\mathrm{E}(P(c \div h))$, and it is the unique UMVUE as well. □

Since the unbiasedness of $\mathscr{A}$ implies the finiteness of $d_{\mathrm{VC}}(\boldsymbol{C})$, in the distribution-free case if $\boldsymbol{C}$ is learnable then it is learnable through the UMVULP. For specific distribution laws the following holds.

Theorem 5. *Let $\boldsymbol{C}$ be a concept class on $(\boldsymbol{X}, \mathscr{F}, P)$ and $\mathscr{A}$ be a learning procedure with output $\{h\}$ such that $\mathrm{E}(P(c \div h) < t$ for $t \in (0,1)$, $h \in \mathscr{A}(\boldsymbol{X}_m^c)$ and $m = m(t)$ is a proper sample size, with P within a class Φ. Let $S_i = g_i(Y_1, \ldots, Y_m), Y_j = (X_j, \chi_c(X_j)), i = 1, \ldots, n, j = 1, \ldots, m$ be a set of joint sufficient complete statistics for $P(c \div h)$ such that $P(c \div h) = g(S_1, \ldots, S_n)$. Then $\mathscr{A}$ is the unique UMVULP for $\boldsymbol{C}$.*

Proof. (1) Let h' be as in Definition 8. Then we have $\mathrm{E}(P(c \div h)) = \mathrm{E}(P(c \div h'))$.

(2) Let us assume that $P(c \div h') = g'(S_1, \ldots, S_n)$. Then by the completeness of $\{S_i = g_i(Y_1, \ldots, Y_m), i = 1, \ldots, n\}$ we have $P(c \div h) = P(c \div h')$ with probability 1. Therefore in this case we have $\mathrm{V}(P(c \div h)) = \mathrm{V}(P(c \div h'))$.

(3) Let us assume that there is no g' such that $P(c \div h') = g'(S_1, \ldots, S_n)$. Now $P(c \div h) = g(S_1, \ldots, S_n) = \mathrm{E}(P(c \div h) | S_1, \ldots, S_n) = \mathrm{E}(P(c \div h') | S_1, \ldots, S_n)$ for the completeness of $\{S_i = g_i(Y_1, \ldots, Y_m), i = 1, \ldots, n\}$. From the Rao-Blackwell theorem [41] on $\mathrm{E}(P(c \div h) | S_1, \ldots, S_n)$ we have $\mathrm{V}(P(c \div h)) \leqslant V(P(c \div h'))$. □

Definition 9. Given a concept class $\boldsymbol{C}$ on $(\boldsymbol{X}, \mathscr{F}, P)$, a learning procedure $\mathscr{A}$ is the *uniformly cheapest* one if and only if, for each c, ε, δ, and minimum value $m(\varepsilon, \delta) = m$ such that $P^{(m)}(P(c \div h) < \varepsilon) > 1 - \delta$ for each $h \in \mathscr{A}(\boldsymbol{X}_m^c)$, no $\mathscr{A}' \neq \mathscr{A}$ and $m'(\varepsilon, \delta) = m'$ with $m' \leqslant m$ exist such that $P^{(m')}(P(c \div h') < \varepsilon) > 1 - \delta$ for some $h' \in \mathscr{A}'(\boldsymbol{X}_{m'}^c)$.

Is the UMVULP also the uniformly cheapest procedure? In general no. Actually, we can state that the UMVULP is a family of PAC learning algorithms, due to the bound on $\mathrm{E}(P(c \div h))$. Moreover, we see by inspection that if the distribution law of $P(c \div h)$ satisfies some regularity conditions for each $h \in \mathscr{A}(\boldsymbol{X}_m^c)$, as in the case of Gaussian or Beta distribution, then we can rely on the UMVULP as the cheapest family of PAC algorithms.

Hence the UMVULP is a good landmark in designing learning algorithms. As a matter of fact, any batch procedure, where we refer to the whole sample regardless of any ordering, is UMVULP in the distribution-free case. Concerning the two well-known algorithms proposed in [35] the one for learning k-CNF is symmetric w.r.t. the sample. On the contrary, a hypothesis generated by the k-term-DNF learning algorithm strongly depends on the order of the sample items, thus proving not optimal in the above sense even if the special oracle required by the algorithm is available.

Our ideal enumerative procedure is UMVULP. But, as we have already said, it is too expensive. The procedure we propose tries to barter the dependence on the distribution law[3] to some minimality of the employed statistics. Namely the statistics we use are *near sufficient,* in the sense that they allow us to approximate only from below the distribution law of $P(c \div \mathscr{A}(\boldsymbol{X}_m^c))$.

3. Butterfly distributions and related sufficient statistics

Let us consider a sample space constituted by the boolean hypercube $\boldsymbol{X}_n = \{0,1\}^n$ and let us represent its points as n bit strings, namely the 2^n different assignments to the variables $x_1,\ldots,x_n$.

The support of a monomial can be considered to have a butterfly shape, according to the figuration mentioned in the introduction. A non-empty intersection of two monomials gives rise to a new butterfly with a longer squeezed string. In this paper we look for a family of distribution laws for which this intersection constitutes a suitable representation of both the monomials, in the sense that it allows us to draw sufficient statistics w.r.t. the probability measures of both these monomials.

3.1. Butterfly distributions

A peculiarity of our distributions is that if the sets of variables that are grouped in the center of two butterflies differ only by a small fraction, then the ratio between the measures of the two butterflies is nicely bounded. Formally:

Definition 10. Let $n,k \in \mathbb{N}$ and $q* = \text{poly}(n)$ (i.e., polynomial in n). We say that the distribution P on $\boldsymbol{X}_n$ is a *butterfly distribution* with parameters k and $q*$ (for short, P is but($k,q*$)) if there exists a function $\alpha \leqslant 1/\text{spoly}(n)$ (i.e., superpolynomial in n), $\alpha \in [0,1]$, such that for every *monomial* μ on $V = \{x_1,\ldots,x_n\}$ and *for each* set I of *at most* k literals, if $\mu_{\downarrow I} \neq \emptyset$ and $P(\mu) \geqslant \alpha/k$ then $P(\mu_{\downarrow I}) \geqslant P(\mu)/q*$ for n large enough.

Basically, the distributions belonging to the butterfly family are those that allow us to check (from one side) the relevance of a monomial just by checking the probability

[3] Indeed each SQ learning procedure is symmetric in the examples, but the unbiasedness proof of our learning procedure is restricted to special distribution laws.

measure of some of its subsets. This family is not narrow. Here we list some relevant members and non-members.

Examples of butterfly distributions

(a) *p-smooth distribution* [32]

For a given $0 < p \leqslant 1$, a distribution P on $\boldsymbol{X}_n$ is said to be *p-smooth* if for any $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \boldsymbol{X}_n$ of Hamming distance one it holds that $\mathrm{P}(\boldsymbol{v}_1) \geqslant pP(\boldsymbol{v}_2)$, where $p = 1/\mathrm{poly}(n)$. Now, given an arbitrary monomial μ and a set I of at most k literals with $\mu_{\downarrow I} \neq \emptyset$, let us consider for each $\boldsymbol{v} \in \mu_{\downarrow I}$ the set A_v of all points of $\boldsymbol{X}_n$ differing from $\boldsymbol{v}$ only in the assignments to some literals of I; their Hamming distance from $\boldsymbol{v}$ is at most k, their number is at most 2^k. Consequently, for each $\boldsymbol{v}' \in A_{\boldsymbol{v}}$, $P(\boldsymbol{v}) \geqslant p^k P(\boldsymbol{v}')$ and $P(\boldsymbol{v}) \geqslant p^k P(A_{\boldsymbol{v}})/2^k$. Finally, since the $\mathrm{A}_{\boldsymbol{v}}$'s induce a partition of μ, we have that $P(\mu_{\downarrow I}) \geqslant P(\mu)(p/2)^k$, which, simply putting $\alpha = 0$,[4] proves that P is but $(k, (2/p)^k)$ for a constant k.

(b) *q-bounded distribution* [6]

For a given $q \geqslant 1$, a distribution P on $\boldsymbol{X}_n$ is said to be *q-bounded* if, for any $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \boldsymbol{X}_n, P(\boldsymbol{v}_1) \leqslant qP(\boldsymbol{v}_2)$. A q-bounded distribution is a p-smooth distribution. In particular P is but $(k, q2^k)$ for $q2^k = \mathrm{poly}(n)$ and $\alpha = 0$.

(c) *Product distribution* [19]

A set $\boldsymbol{X}_n$ is said to have a *product distribution* P if $(\boldsymbol{X}_n, \mathscr{F}, P)$ is described by a random vector $\boldsymbol{B} = (B_1, \ldots, B_n)$, where $B_i, i = 1, \ldots, n$, are independent Bernoullian variables of parameters r_i, where $0 < r \leqslant r_i \leqslant 1 - r < 1$. A product distribution is a p-smooth distribution. In particular P is but$(k, q*)$ for $q* \leqslant (\min\{r, 1-r\})^{-k} = \mathrm{poly}(n)$ and $\alpha = 0$.

Since we have set $\alpha{=}0$ in all previous examples, the reader may be reasonably wondering about the use of α in the definition of butterfly distributions. A main property of these distributions is that they allow zero probability holes in $\boldsymbol{X}_n$, provided that the probability is fairly distributed among the remaining points. When the probability $P(\mu)$ of a monomial μ is very close to 0 (say, $P(\mu) = 1/2^n$ with n large), it may happen that, while μ does contain points of positive probability, a restriction $\mu_{\downarrow I}$ does not, even if $\mu_{\downarrow I} \neq \emptyset$. The requirement $\alpha \leqslant 1/s\ \mathrm{poly}(n)$ is meant as a suitable bound on α, to bound the measure of these odd monomials *from above*. Indeed, all we need to check according to Definition $1'$ is the *existence* of a function s that goes to ∞ faster than any polynomial. For this s we can set $\alpha(n) \leqslant 1/s(n)$ in Definition 10. In later sections we will relate α to the accuracy parameter ε by means of inequalities of the form "$\alpha(n) \leqslant \varepsilon$". This ε can be reasonably set to $1/\mathrm{poly}(n)$, according to the notion of poly-relaxed learning algorithms introduced in Definition $1'$. The rest of this section illustrates the point.

(d) *Uniform distribution on subsets of* $\boldsymbol{X}_n$. The uniform distribution on the whole $\boldsymbol{X}_n$ is a special case of both product and q-bounded distributions. Here we consider some special restrictions of the support of the random variable.

[4] Throughout the paper we will write "$\alpha{=}0$" as an abbreviation for "$\alpha(n) = 0 \forall n$" ; whereas, every *inequality* involving $\alpha(n)$, such as $\alpha(n) \leqslant \varepsilon$ should be understood to hold for *n large enough*.

(d.1) Let P be the *uniform* distribution over the odd-parity points of $\boldsymbol{X}_n$ (i.e., the points that have an odd number of components set to one). The number of such points is 2^{n-1}. Let $n > k$ and consider the family of monomials μ of $l < n-k$ literals. Each μ contains exactly 2^{n-l-1} points of odd parity, thus its measure is $2^{n-l-1}/2^{n-1} = 2^{-l}$. For a set I of $k' \leqslant k$ literals, if $\mu_{\downarrow I} \neq \emptyset$ then $P(\mu_{\downarrow I}) \geqslant 2^{n-l-k}$. Thus if $n > k$ then P is but $(k, 2^k)$ for $\alpha(n)/k > 2^{-(n-k)}$.

(d.2) Let $\boldsymbol{P}_\lambda$ be the family of distributions that are *uniform* over the points of $\boldsymbol{X}_n$ having at most $\lfloor n\lambda \rfloor$ components set to one and zero elsewhere, with $0 < \lambda < 1/2$. In the appendix we show the following :

Lemma 1. *For each $\lambda \in (0, 1/2)$ and $P \in \boldsymbol{P}_\lambda$ if n is chosen so large that for a constant k we have $\lfloor n\lambda \rfloor > k > 1$, then P is* but$(k, \mathbf{O}(n^{2k}))$ *for* $[en/(k-1)]^{k-1}\lambda^{\lfloor n\lambda \rfloor} < \alpha(n)/k$, *where e is the base of the natural logarithm.*

Examples of non-butterfly distributions

(e) Let P be the probability distribution on $\boldsymbol{X}_n$ that assigns measure 1/2 to the point $v* = (0, 0, 0, \ldots, 0)$ and measure $1/(2^{n+1}-2)$ to the remaining points. Let $\mu = x_1^c x_2^c x_3^c \cdots x_{n-1}^c$. Obviously $P(\mu) \geqslant 1/2$, as $v* \in \mu$. Assume $k \geqslant 2$ (which implies $1/2 \geqslant \alpha(n)/k$). If $I = \{x_n\}$, the monomial $\mu_{\downarrow I} = \mu \wedge x_n = x_1^c x_2^c x_3^c \cdots x_{n-1}^c x_n$ has measure $P(\mu_{\downarrow I}) = 1/(2^{n+1} - 2)$. Therefore $P(\mu_{\downarrow I})/P(\mu) \leqslant 1/(2^n - 1)$, where $2^n - 1$ is not a polynomial function in n. We conclude that if $k \geqslant 2$ then P is not a butterfly distribution.

A further negative example will be supplied later in Section 3.2.

Even from a strictly operational point of view, the butterfly family does not appear trivially reducible to another already studied distribution family. In particular, let us consider the following *dominance* relation.

Definition 11 (*Benedek and Itai* [7]). Given two probability spaces $(\boldsymbol{X}, \mathscr{F}, P_1)$, $(\boldsymbol{X}, \mathscr{F}, P_2)$ and $\gamma \geqslant 1, P_1$ is said to *γ-dominate* P_2 if and only if for every $A \in \mathscr{F}$ we have $P_2(A) \leqslant \gamma P_1(A)$.

It is easy to prove that the SQ learnability of a concept class $\boldsymbol{C}$ in a given probability space $(\boldsymbol{X}, \mathscr{F}, P)$ extends to any $P'\gamma$-dominated by P, for γ polynomial in the relevant parameters of the class $\boldsymbol{C}$ and the SQ algorithm. On the other hand, the following lemma – whose proof is reported in the appendix – states that not all the members of the butterfly family can be fairly dominated by distributions from among those mentioned in this section.

Lemma 2. *For each $\lambda \in (0, 1/2)$, $P \in \boldsymbol{P}_\lambda$ and n large enough, P is not γ-dominated by a p-smooth distribution, for any p and any $\gamma = \mathrm{poly}(n)$.*

Remark 1. Note that $\boldsymbol{P}_\lambda$ is not a purely artificial distribution class. If we consider monotone formulas, for instance, $P \in \boldsymbol{P}_\lambda$ becomes the uniform distribution law of the *incomplete assignments* with λ limiting the number of assigned values.

For our specific learning problem, a broad interpretation of the role played by α, k and $q*$ in butterfly distribution is the following: k is an upper bound on the number of terms of the actual target formula; q^* and α represent the interaction between the distribution on $\boldsymbol{X}_n$ and the class of k-term-DNF(n) formulas. With reference to examples (a)–(c), we call the interaction *fair* when we do not need to introduce a lower bound on α to get a butterfly distribution. Stated otherwise, we can set $\alpha = 0$. With reference to examples (d.1), (d.2), we call the interaction *unfair* when we do need to impose a lower bound on α. This latter kind of interaction bounds the accuracy P_{error} obtainable by our algorithm from *below*. However this lower bound, which is a sort of admittance fee for many distribution laws, can still be considered reasonable, since it is of the form $1/\text{poly}(n)$. Here we get polynomial time algorithms only in the poly-relaxed sense.

3.2. Discriminants and notation

In line with some ideas that appear for instance in [1, 8, 15], we present the following definitions and results:

Definition 12. Given a k-term-DNF(n) $c = t_1 + t_2 + \cdots + t_{k'}, k' \leqslant k$, a *discriminant*[5] for a term t_i is a collection of at most $k' - 1$ literals $L = \{l_j, j = 1, \ldots, k'' \leqslant k' - 1\}$ without complementary pairs such that for every $t_r \neq t_i$ there is a literal of t_r belonging to L and no literal of t_i is in L.

For example, if $c = t_1 + t_2 + t_3$, $t_1 = x_1x_2x_3^c$, $t_2 = x_3x_4x_5$, $t_3 = x_1x_4$, a discriminant for t_1 is $L = \{x_4, x_5\}$, another one is $L = \{x_4\}$.

Remark 2. The discriminant provides a mechanism for focusing on a single term of the target formula. Let L^c be the set of the negations of the literals in L. If L is a discriminant of a term t_i in c and, during sampling $\boldsymbol{X}_n$, we draw a point $\boldsymbol{v}$ such that $\boldsymbol{v} \in c_{\downarrow L^c}$, then this point satisfies only term t_i in c. Indeed, every term in c shares a literal with L, except for t_i . For instance, if $c = x_1 + x_2^c$, $L = \{x_2^c\}, \boldsymbol{v}_1 = (1,1), \boldsymbol{v}_2 = (1,0)$, then $\boldsymbol{v}_1 \in c_{\downarrow L_c}$ and it satisfies only the first term, $\boldsymbol{v}_2$ does not spur any term of c.

Thus, we can associate with each discriminant L of t a monomial $\bar{\mu}_L$, i.e., a butterfly whose squeezed part contains the binary values satisfying the negations of the literals in L. By definition, $t_{\downarrow L^c} = \bar{\mu}_L \cap t \neq \emptyset$ and it is a butterfly too whose squeezed part is at most $k - 1$ bits longer than the squeezed part of t. If the probability distribution P on $\boldsymbol{X}_n$ is a butterfly, we might deduce a lower bound on $P(t_{\downarrow L^c})$ through $P(t)$. This is the key point of the learning algorithm we will present in the next section. In fact, we will show a statistic for inferring about $P(t_{\downarrow L^c})$. This statistic proves sufficient for stating lower bounds on the distribution function of the variable $P(c \div h)$.

[5] The word 'discriminant' comes from Angluin's paper [1] where she defined a slightly different notion of it. By abuse of terminology we use the same word here.

In the sequel we will also make use of the two following weaker notions of discriminant.

Definition 13. Given a k-term-DNF(n) c, a *subformula* c' of c is a disjunction of terms in c. A (*weak*) *discriminant* L of c' is a collection of at most $k-1$ literals without complementary pairs, such that for every term t_r in c and not in c' there is a literal of t_r belonging to L, and no literal of L appears in a term of c'. L is a *false discriminant* for c if it is a set of at most k literals without complementary pairs such that for every term t_r of c there is a literal in t_r belonging to L.

Note that according to our notational conventions for $\bar{\mu}_L$, Definitions 12 and 13 can be restated through the following conditions:
– discriminant: $\bar{\mu}_L$ has $\leqslant k-1$ literals, $\bar{\mu}_L \neq \emptyset$ and $t_{\downarrow_{L^c}} = c_{\downarrow_{L^c}}$;
– weak discriminant: $\bar{\mu}_L$ has $\leqslant k-1$ literals, $\bar{\mu}_L \neq \emptyset$ and $c'_{\downarrow_{L^c}} = c_{\downarrow_{L^c}}$;
– false discriminant: $\bar{\mu}_L$ has $\leqslant k$ literals, $\bar{\mu}_L \neq \emptyset$ and $c_{\downarrow_{L^c}} = \emptyset$.

Obviously, since the learning algorithm does not know c, it does not know the discriminants of the terms of c as well. However, we can state the following lemmas:

Lemma 3 (Blum and Rudich [8]; Flemmini et al. [5]). (*a*) *If a k-term-DNF*(n) *formula c is made only of prime implicants* (*i.e.*, *there is no term* t_i *in c that can be removed without changing* χ_c *and there is no literal that can be removed from any* t_i *in c without changing* χ_c) *then every term of c admits a discriminant.*

(*b*) *If* $c \neq \boldsymbol{X}_n$ *then c admits at least one false discriminant.*

Lemma 4. *For given k and n, the number of possible discriminants* (*even weak or false*) *for a term or a subformula of all the k-term-DNF* (n) *formulas is* $< (2en/k)^k$.

Proof. Any unordered selection of k out of $2n$ literals without complementary pairs is a possible discriminant. Then, the number of discriminants is less than $\sum_{l=0}^{k} \binom{2n}{l} < (2en/k)^k$ [33]. □

Before closing this section, let us present another example of a non-butterfly distribution:

(f) *Uniform distribution on the support of a k-term-DNF formula*

Given a k-term-DNF (n) formula $c = t_1 + t_2 + \cdots + t'_{k'}, k' \leqslant k$, such that $c \neq \boldsymbol{X}_n$, let us consider a probability distribution P that is *uniform* over all the points of $\boldsymbol{X}_n$ satisfying c. Let us focus on the monomial (without literals) $\mu = \boldsymbol{X}_n$ and on a false discriminant L of c. Obviously $P(\mu) = 1$. Now, $\mu_{\downarrow_{L^c}} \neq \emptyset$ and $c_{\downarrow_{L^c}} = \emptyset$, by definition. This means that no point of measure greater than zero satisfies $\mu_{\downarrow_{L^c}}$. Then, despite $P(\mu) = 1, P(\mu_{\downarrow_{L^c}}) = 0$ holds and no butterfly parameter can fill the 'infinite' multiplicative gap between these two probabilities.

4. The learning algorithm

Let us finally describe the k-term-DNF learning algorithm for butterfly distributions. The key hypothesis that makes the algorithm feasible is that the given distribution P on $\boldsymbol{X}_n$ is of but(k,q^*) kind, where k is an upper bound on the actual number of terms of the DNF to be learnt and q^* is a *known*[6] polynomial in n. The role of the function α is immaterial in the case of fair interactions, since we are allowed to set α to 0. By contrast, for unfair interactions α must be compared to the final accuracy parameter ε. The algorithm we present in this section accomplishes proper learning for both fair and unfair interactions: specifically, in the first case the learning task follows Definition 1, and in the second case Definition $1'$.

We will specialize the algorithm for both the non-monotone case (Section 4.2.1) and the monotone case (Section 4.2.2).

4.1. Survey of the algorithm

The structure of the algorithm is shown below, where the parts in square brackets apply only to the monotone case.

DISCRIMINANT_LEARNING
for each selection of $k' \leqslant k-1$ literals l_i without complementary pairs
  $L = \{l_1, \ldots, l_{k'}\}$
  if EMPTY (L) = 'ok'
    then $t_{\text{cand}}[, t_{\text{test}}]$ = INFER_SUBTERM (L);
h_{large}=PRUNE $(\{t_{\text{cand}}\})$;
h_{end}=FILTER $(h_{\text{large}}[, \{t_{\text{test}}\}])$;
return (h_{end});
end (DISCRIMINANT_LEARNING)

The algorithm starts enumerating all possible sets L of at most $k-1$ literals without complementary pairs. Viewing L as a candidate discriminant of a term of the target formula c, the subroutine EMPTY accepts L if it discriminates subformulas of non-negligible measure. Then INFER_SUBTERM cumulates, at the end of the enumeration, a set of monomials $\{t_{\text{cand}}\}[\cup\{t_{\text{test}}\}]$ that include all the relevant terms of c. This set is processed by subroutines PRUNE and FILTER for dropping exceeding monomials, leaving at most k terms in the final set h_{end}, namely the above relevant ones together with (possibly) some very small residual monomials, which do not affect the accuracy of the estimation.

PRUNE operates on every monomial suggested by INFER_SUBTERM and FILTER extracts the final hypothesis from the whole set of remaining terms. These subroutines

[6] In Section 4.3 we will remove the requirement that q* be known in advance.

act in a completely different way depending on whether the concept that they deal with is monotone or not.

Namely, the strategy followed in the latter case is just to clean $\{t_{\text{cand}}\}$ by removing those monomials t whose difference $t-c$ appears too high, and then to find in a purely combinatorial way a subset h of at most k terms that minimizes the measure of the symmetric difference $c \div h$.

On monotone c, we recognize that no restriction of terms of c is generated by any inference of our algorithm. This fact allows us to identify the terms of c as the minimal restrictions of the monomials in $\{t_{\text{cand}}\}$. Here special care must be paid in deciding about the relevance of the monomials, in order to have in $\{t_{\text{cand}}\}$, for each monomial μ, a term t of c such that t is a restriction of μ. This is achieved through the auxiliary set $\{t_{\text{test}}\}$.

The main job of most of these subroutines might be described as a series of queries to an oracle STAT(c,P) as described in Section 2.

4.2. Procedure description and analysis

We have two goals: (a) to show that the algorithm selects a right formula, (b) to prove that this job is polynomial. We will start by checking the job of the subroutines w.r.t. the first goal, dealing with non-monotone and monotone formulas in two different subsections. In the next section we will bound the resources used by the whole algorithm in both cases.

In regard to notation, in the pseudocode of the routines below we will write $x=$ query $(\langle \psi; \alpha \rangle)$ to denote the answer of STAT (c,P) to query $\langle \psi; \alpha \rangle$.

4.2.1. Non-monotone formulas

- EMPTY

The goal of EMPTY is to make INFER_SUBTERM consider only significantly large subformulas.

EMPTY(L)
$P' = \text{query}(\langle \boldsymbol{v} \in c_{\downarrow_{L^c}}; \varepsilon_1/2kq^* \rangle)$;
if $P' > \varepsilon_1/2kq^*$ **then return** ('ok')
else return ('wrong')
end (EMPTY)

Lemma 5. *Let P be a* but(k,q^*) *distribution on $\boldsymbol{X}_n$, t^* be a term of probability measure $\geqslant \varepsilon_1/k$ of a k-term-DNF (n) formula c and L be a discriminant of t^*. If $\alpha(n) \leqslant \varepsilon_1$ then* EMPTY (L) ='ok'.

Proof. Simply note that, since L is a discriminant of t^*, then $P(c_{\downarrow_{L^c}}) = P(t^*{}_{\downarrow_{L^c}}) \geqslant \varepsilon_1/kq^*$. Therefore $P' > \varepsilon_1/kq^* - \varepsilon_1/2kq^* = \varepsilon_1/2kq^*$. □

Table 1
The probability model of the oracle STAT(c,P) answering INFER_SUBTERM queries

	$x_i^c \notin L^c$		$x_i \notin L^c$		$x_i \in L^c$	$x_i^c \in L^c$
	$x_i^c \notin t^*$	$x_i^c \in t^*$	$x_i \notin t^*$	$x_i \in t^*$		
$P(t*_{\downarrow_{L^c}}, v_i = 1)$	$\geqslant \frac{\varepsilon_1}{kq*}$	0	?	$\geqslant \frac{\varepsilon_1}{kq*}$	$\geqslant \frac{\varepsilon_1}{kq*}$	0
$P(t*_{\downarrow_{L^c}}, v_i = 0)$	?	$\geqslant \frac{\varepsilon_1}{kq*}$	$\geqslant \frac{\varepsilon_1}{kq*}$	0	0	$\geqslant \frac{\varepsilon_1}{kq*}$

• INFER_SUBTERM
The inferential core of the procedure is realized as follows:

INFER_SUBTERM(L)
$T_{\text{cand}} = \{x_1, \ldots, x_n, x_1^c, \ldots, x_n^c\} - L$;
for $i = 1, \ldots, n$
  if $x_i^c \notin L$ **then** $P' = \text{query}(\langle \boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = 1; \varepsilon_1/2kq^* \rangle)$;
    if $P' \geqslant \varepsilon_1/2kq^*$ **then** $T_{\text{cand}} = T_{\text{cand}} - \{x_i^c\}$;
  if $x_i \notin L$ **then** $P' = \text{query}(\langle \boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = 0; \varepsilon_1/2kq^* \rangle)$;
    if $P' \geqslant \varepsilon_1/2kq^*$ **then** $T_{\text{cand}} = T_{\text{cand}} - \{x_i\}$;
$t_{\text{cand}} = \wedge_{l_i \in T_{\text{cand}} - L^c} l_i$
return (t_{cand});
end(INFER_SUBTERM).

Lemma 6. *Let P be a* but(k,q^*) *distribution on $\boldsymbol{X}_n$, t^* be a term of probability measure $\geqslant \varepsilon_1/k$ of a k-term-DNF (n) formula c, L be a discriminant of t^* and* $t_{\text{cand}} =$ INFER_SUBTERM(L). *If $\alpha(n) \leqslant \varepsilon_1$ then $t^* \leqslant_{L^c} t_{\text{cand}}$.*

Proof. According to the butterfly hypothesis, the probability model of Table 1 holds. In this table the probability of the cells labelled by '?' depends on the membership of the complementary literal in set (t^*).

If $l_i \in L$ then $l_i \notin \text{set}(t^*)$, by the definition of discriminant. If $l_i \notin L$ then the above model implies that, due to its accuracy, STAT (c,P) sharply discriminates about the membership of l_i in set (t^*) as long as $l_i \notin L^c$. Here we also note that, since L is a discriminant of t^* then $c_{\downarrow_{L^c}} = t^*{}_{\downarrow_{L^c}}$ and thus the events $\{\boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = j\}$ and $\{\boldsymbol{v} \in t^*{}_{\downarrow_{L^c}}, v_i = j\}$, $j = 0,1$, are equivalent. So we can correctly decide to exclude $x_i^c \notin (L \cup L^c)$ from t^* when $P' =$ query $(\langle \boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = 1; \varepsilon_1/2kq^* \rangle) \geqslant \varepsilon_1/2kq^*$, as is indicated in the test of the subroutine. Indeed, if $x_i^c \in$ set (t^*) then we would have $P(\boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = 1) = 0$ and therefore $P' < \varepsilon_1/2kq^*$. On the other hand, this threshold is not too high, since $x_i^c \notin \text{set}(t^*)$ implies $P(\boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = 1) \geqslant \varepsilon_1/kq^*$ and then $P' > \varepsilon_1/kq^* - \varepsilon_1/2kq^* = \varepsilon_1/2kq^*$. An analogous proof works for x_i.

From the last two columns of Table 1 we can see that if $l_i \in L^c$ we cannot decide the membership of l_i through INFER_SUBTERM queries. Therefore for that case we

adopt the strategy of removing all the literals belonging to L^c from T_{cand}, and then checking all the monomials coming from adding all the subsets of L^c to set(t_{cand}). □

• PRUNE

For a given discriminant L of t^*, set (t_{cand}) contains all the literals of t^*, except every $l_i \in L^c$. Therefore we add to the inferred monomials each restriction t_{res} of t_{cand} obtained by adding subsets of L^c to set (t_{cand}). Then we keep only those monomials whose difference t-c is small enough ($P' < \varepsilon_1/2k$).

PRUNE(h)
$h_{\text{large}} = \emptyset$;
for each $t_{\text{cand}} \in h$
  $L = \{l_1, \ldots, l_{k'}\}$ such that t_{cand}= INFER_SUBTERM(L);
  $L^c = \{l_1^c, \ldots, l_{k'}^c\}$;
  for each subset M of L^c
    $t_{\text{res}} = t_{\text{cand}} \wedge (\bigwedge_{l_i^c \in M} l_i^c)$
    P'=query($\langle \boldsymbol{v} \in t_{\text{res}} - c; \varepsilon_1/2k \rangle$);
    if $P' < \varepsilon_1/2k$ **then** $h_{\text{large}} = h_{\text{large}} \cup \{t_{\text{res}}\}$;
return (h_{large});
end(PRUNE)

Lemma 7. *If t_{res} is a term of c then $t_{\text{res}} \in$ PRUNE($\{t_{\text{cand}}\}$); if $P(t_{\text{res}} - c) \geqslant \varepsilon_1/k$ then $t_{\text{res}} \notin$PRUNE($\{t_{\text{cand}}\}$).*

Proof. By inspection, given the accuracy of the query and the fact that $P(t_{\text{res}} - c) = 0$ whenever t_{res} is a term of c. □

• FILTER

After PRUNE, h_{large} is a set of at most $2^{k-1}N_k$ terms, where N_k is the total number of discriminants allowed by the class of k-term-DNF (n). By Lemma 4, $N_k < (2en/k)^k$. In h_{large} we can find:
(1) all the terms of c having measure $\geqslant \varepsilon_1/k$;
(2) further terms t such that $P(t - c) < \varepsilon_1/k$.
Therefore, due to the items under (1) above, the problem of picking at most k terms that fulfill c is a set-covering problem that admits at least one solution with waste $< \varepsilon_1$. FILTER solves this problem in a purely enumerative way.

FILTER (h)
for each $t_{i1}, \ldots, t_{ik'} \in h, 0 \leqslant k' \leqslant k$
  $h_{\text{trial}} = t_{i1} + \cdots + t_{ik'}$;
  $P'(h_{\text{trial}}) = \text{query}(\langle \boldsymbol{v} \in c - h_{\text{trial}}; \varepsilon_1 \rangle)$;
$P'(h_{\text{end}}) = \min_{h_{\text{trial}}} P'(h_{\text{trial}})$;
return (h_{end})
end (FILTER)

Lemma 8. *Let P be a* but$(k, q*)$ *distribution on $\boldsymbol{X}_n$. Then for $\alpha(n) \leqslant \varepsilon_1$* FILTER (h_{large}) *returns a k-term-DNF(n) formula h_{end} such that $P(c \div h_{\text{end}}) < 4\varepsilon_1$ as hypothesis.*

Proof. Since for each trial h_{trial} is made up of at most k terms, we have that $P(h_{\text{trial}} - c) < \varepsilon_1$, by Lemma 7. Hence $P(h_{\text{end}} - c) < \varepsilon_1$.

Let h^* be the h_{trial} for which $P(c - h_{\text{trial}})$ is minimized. Since h_{large} contains all the terms t^* of c such that $P(t^*) \geqslant \varepsilon_1/k$ then $P(c - h^*) < \varepsilon_1$. The query accuracy allows us to write $|P'(h_{\text{trial}}) - P(c - h_{\text{trial}})| < \varepsilon_1$ for each h_{trial}, so $P(c - h_{\text{end}}) < P'(h_{\text{end}}) + \varepsilon_1 \leqslant P'(h^*) + \varepsilon_1 < P(c - h^*) + 2\varepsilon_1 < 3\varepsilon_1$.

Remark 3. It is easy to see that in the case of non-monotone formulas EMPTY is not necessary and PRUNE might come to add all the restrictions of the monomials suggested by INFER_SUBTERM. In fact FILTER is able to extract the final hypothesis even from the resulting larger set of monomials. Nevertheless the above optional actions actually allow us to reduce the search space of FILTER, thus adding efficiency to the whole procedure.

4.2.2. Monotone formulas

If we restrict ourselves to monotone DNF formulas, the learning task can be performed more efficiently.

Now a discriminant is a collection of at most $k - 1$ non-negated *variables*. As we will show in Corollary 1, this fact makes our inference tools never underestimate the relevant terms of c. However, *purely syntactical* rules have to be applied in order to remove further monomials not belonging to c from the set of the inferred terms. The resulting hypothesis h is always included in c, possibly missing some terms of negligible probability measure. As usual for monotone formulas, the absence of overestimating errors allows us to restrict the subject of our queries to the sole support of c (positive examples), if the examples are error-free. We still do need to refer to the whole sample space in the case of noisy examples.

In the sequel we give slightly modified versions of EMPTY and INFER_SUBTERM, and new versions of PRUNE and FILTER. Their names carry a suffix ".M " to denote that they refer exclusively to monotone formulas.

- EMPTY.M

Now the subroutine makes two tests, whose difference lies only in the threshold on the size of accepted subformulas.

EMPTY.M(L)
flag = 'wrong';
$P' =$ query$(\langle \boldsymbol{v} \in c_{\downarrow_{L^c}}; \varepsilon_1/8k^4(q*)^2 \rangle)$;
if $P' \geqslant 3\varepsilon_1/8k^4(q*)^2$ **then** *flag* = 'ok';
if $P' \geqslant 3\varepsilon_1/4k^2q*$ **then** *flag* = 'okk';
return(*flag*)
end (EMPTY.M)

Lemma 9. *Let P be a* but(k,q^*) *distribution on $\boldsymbol{X}_n$, c' a subformula of a k-term-DNF(n) formula c, L a discriminant of c'. Then for $\alpha(n)\leqslant\varepsilon_1/2k^3q^*$:*
(1) *if* EMPTY.M(L) = *'wrong' then* $P(c')<\varepsilon_1/2k^3q^*$;
(2) *if* EMPTY.M(L) = *'okk' then* $P(c')\geqslant\varepsilon_1/2k^2q^*$;
(3) *if* EMPTY.M(L) = *'ok' then* $P(c')\geqslant\varepsilon_1/4k^4(q^*)^2$;
(4) *if c' is a single term and* $P(c')\geqslant\varepsilon_1/k$ *then* EMPTY.M(L)= 'okk'.

Proof. Let $c'=t_{i1}+\cdots+t_{ik'}, k'\leqslant k,\ L=\{x_{jk''}\}, k''\leqslant k'-1$.
First of all, we note that, as L is a discriminant of c', $c_{\downarrow L^c}=c'_{\downarrow L^c}$.
(1) Assume that $P(c')\geqslant\varepsilon_1/2k^3q^*$. $c_{\downarrow L^c}=c'_{\downarrow L^c}=(t_{i1}+\cdots+t_{ik'})\wedge\bar{\mu}_L=t_{i1}\wedge\bar{\mu}_L+\cdots+t_{ik'}\wedge\bar{\mu}_L$. Since $P(c')\geqslant\varepsilon_1/2k^3q^*$ then there exists a term t^*_{ij} of c' such that $P(t^*_{ij})\geqslant\varepsilon_1/2k^4q^*$. Since P is but (k,q^*) with $\alpha(n)\leqslant\varepsilon_1/2k^3q^*$ we have $P(c_{\downarrow L^c})\geqslant P(t^*_{ij}\wedge\bar{\mu}_L)=P(t^*_{ij}\downarrow L^c)\geqslant\varepsilon_1/2k^4(q^*)^2$. Then $P'>\varepsilon_1/2k^4(q^*)^2-\varepsilon_1/8k^4(q^*)^2=3\varepsilon_1/8k^4(q^*)^2$ and EMPTY.M$(L)\neq$ 'wrong', a contradiction.
(2) Assume that $P(c')<\varepsilon_1/2k^2q^*$. Then, obviously, $P(c'_{\downarrow L^c})<\varepsilon_1/2k^2q^*$, hence $P'<\varepsilon_1/2k^2q^*+\varepsilon_1/8k^4(q^*)^2<5\varepsilon_1/8k^2q^*$ and EMPTY.M$(L)\neq$ 'okk', a contradiction.
(3) Assume that $P(c')<\varepsilon_1/4k^4(q^*)^2$. Then $P(c'_{\downarrow L^c})<\varepsilon_1/4k^4(q^*)$, hence $P'<\varepsilon_1/4k^4(q^*)^2+\varepsilon_1/8k^4(q^*)^2=3\varepsilon_1/8k^4(q^*)^2$ and EMPTY.M(L) $\neq$ 'ok', a contradiction.
(4) Since $P(c')\geqslant\varepsilon_1/k$ then $P(c'_{\downarrow L^c})\geqslant\varepsilon_1/kq^*$ and $P'>\varepsilon_1/kq^*-\varepsilon_1/8k^4(q^*)^2\geqslant7\varepsilon_1/8kq^*$ and EMPTY.M(L) = 'okk'. □

• INFER_SUBTERM.M
We will make two changes to the previous version:
(i) the case $x_i^c\notin L$ does not apply;
(ii) we produce two sets, $\{t_{\text{cand}}\}$ and $\{t_{\text{test}}\}$. Their difference lies both in the discriminants and in the threshold used to select their items. The use of the new set $\{t_{\text{test}}\}$ will be clear in FILTER.M.

INFER_SUBTERM.M(L)
$t_{\text{cand}}=\boldsymbol{X}_n$
$T_{\text{cand}}=\{x_1,\ldots,x_n\}-L$;
$T_{\text{test}}=\{x_1,\ldots,x_n\}-L$;
for $i=l,\ldots,n$
  if $x_i\notin L$ **then**
    $P'=\text{query}(\langle\boldsymbol{v}\in c_{\downarrow L^c}, v_i=0;\varepsilon_1/4k^3(q^*)^2\rangle)$;
    if $P'\geqslant\varepsilon_1/4k^3(q^*)^2$ **then** $T_{\text{cand}}=T_{\text{cand}}-\{x_i\}$;
    $P''=\text{query}(\langle\boldsymbol{v}\in c_{\downarrow L^c}, v_i=0;\varepsilon_1/8k^5(q^*)^3\rangle)$;
    if $P''\geqslant\varepsilon_1/8k^5(q^*)^3$ **then** $T_{\text{test}}=T_{\text{test}}-\{x_i\}$;
$t_{\text{test}}=\wedge_{l_i\in T_{\text{test}}}l_i$;
if EMPTY.M(L) = 'okk' **then** $t_{\text{cand}}=\wedge_{l_i\in T_{\text{cand}}}l_i$;

return $(t_{\text{cand}}, t_{\text{test}})$;
end (INFER_SUBTERM.M)

Lemma 10. *Let P be a* but(k, q^*) *distribution on $\boldsymbol{X}_n$, c' a subformula of probability measure $\geqslant \varepsilon_1/2k^2q^*$ of a monotone k-term-DNF(n) formula c and L a discriminant of c'. Then for $\alpha(n) \leqslant \varepsilon_1/2k^2q^*$ there exists a term t^* of c' such that $P(t^*) \geqslant \varepsilon_1/2k^3q^*$ and* set$(t^*) \supseteq$ set$(t_{\text{cand}}$ *generated by* INFER_SUBTERM.M$(L))$.

Proof. Since $P(c') \geqslant \varepsilon_1/2k^2q^*$, there exists a term t^* of c' such that $P(t^*) \geqslant \varepsilon_1/2k^3q^*$. Since P is but (k, q^*), with $\alpha(n) \leqslant \varepsilon_1/2k^2q^*$, then $P(c_{\downarrow_{L^c}}) \geqslant P(t*_{\downarrow_{L^c}}) \geqslant \varepsilon_1/2k^3(q^*)^2$. So, for every variable $x_i \notin$ set $(t^*), P' =$ query$(\langle \boldsymbol{v} \in c_{\downarrow_{L^c}}, v_i = 0; \varepsilon_1/4k^3(q^*)^2 \rangle)$ is large enough for deleting x_i from T_{cand}. Then set $(t^*) \supseteq$ set (t_{cand}). □

An analogous lemma holds for T_{test}. Let us summarize:

Corollary 1. *Let P be a* but(k, q^*) *distribution on $\boldsymbol{X}_n$, t^* a term of probability measure $\geqslant \beta$ of a monotone k-term-DNF (n) formula c and L a discriminant of t^*.*

- *If $\beta = \varepsilon_1/2k^2q*$ and $\alpha(n) \leqslant \beta$ then $t^* = (t_{\text{cand}}$ generated by* INFER_SUBTERM.M(L)).
- *If $\beta = \varepsilon_1/4k^4(q^*)^2$ and $\alpha(n) \leqslant \beta$ then $t^* = (t_{\text{test}}$ generated by* INFER_SUBTERM.M(L)).
- *In both cases no L generates a restriction of the concerned t^*.*

Proof. The claim follows directly from Lemma 6, since if L is a discriminant of a relevant term then the sole allowed restriction of $t_{\text{cand}}[t_{\text{test}}]$ is $t_{\text{cand}}[t_{\text{test}}]$ itself, and from Lemma 10, due to the features of discriminants of subformulas containing relevant terms. □

- PRUNE.M

PRUNE.M removes from $\{t_{\text{cand}}\}$ all monomials that can be found too large. Namely it compares each t_{cand} with all the remaining monomials of this set, deleting those that include (*even not strictly*) t_{cand}.

PRUNE.M(h)
for each $t \in h$
  for each $t' \in h - \{t\}$
  if set$(t) \supseteq$ set(t') **then** $h = h - \{t'\}$
return(h)
end(PRUNE.M)

Lemma 11. *Let P be a* but(k, q^*) *distribution on $\boldsymbol{X}_n$ and c be a monotone k-term-DNF(n) formula. Then, for $\alpha(n) \leqslant \varepsilon_1/2k^2q^*, h_{\text{large}} =$* PRUNE.M$(\{t_{\text{cand}}\})$ *can be partitioned into the following two sets:*
(1) *all the prime implicants of c of measure $\geqslant \varepsilon_1/k$,*
(2) *further monomials that include terms t^* of c such that $\varepsilon_1/2k^3q^* \leqslant P(t^*) < \varepsilon_1/k$.*

Proof. Without loss of generality, we assume that c is made up of prime implicants only.

The items under (1) above are guaranteed to belong to h_{large} by the fact that if $t^* \in c, P(t^*) \geqslant \varepsilon_1/k$ and L is a discriminant of t^*, then EMPTY.M(L) = 'okk' by Lemma 9. Thus $t^* \in \{t_{\text{cand}}$ generated by INFER_SUBTERM.M$\}$ does not get deleted in PRUNE.M by any restriction of t^* (Corollary 1).

As far as the remaining items of PRUNE.M($\{t_{\text{cand}}\}$) are concerned, Lemma 10 states that each t_{cand} inferred by INFER_SUBTERM.M includes some term t^* of c such that $P(t^*) \geqslant \varepsilon_1/2k^3q^*$. Due to Corollary 1 each monomial μ such that $P(\mu) \geqslant \varepsilon_1/k$ either belongs to c or is pruned by a $t^* \in c$. □

• FILTER.M

FILTER.M picks from $\{t_{\text{test}}\}$ the monomials of c of measure $\geqslant \varepsilon_1/2k^3q^*$ that are included in the monomials of the second part of h_{large} (see Lemma 11). Then FILTER.M adds the former monomials to and removes the latter monomials from h_{large}. The whole job is accomplished by simply comparing the monomials t of h_{large} with the monomials t' of $\{t_{\text{test}}\}$: each time set $(t') \supseteq$ set $(t), t$ is *replaced by* t' *in* h_{large}.

What remains at the end is just the final hypothesis the procedure DISCRIMINANT_LEARNING generates for c.

```
FILTER.M(h, h')
for each t ∈ h
  for each t' ∈ h'
    if set(t') ⊇ set(t) then t = t'
return(h)
end(FILTER.M)
```

Lemma 12. *Let P be a* but(k, q^*) *distribution on $\mathbf{X}_n$ and c be a monotone k-term-DNF(n) formula. For $\alpha(n) \leqslant \varepsilon_1/4k^4(q^*)^2$* FILTER.M *outputs as hypothesis a monotone k-term-DNF(n) formula h such that $P(c \div h) < \varepsilon_1$.*

Proof. By Corollary 1 and Lemma 9, for each t^* such that $t^* \in c$ and $P(t^*) \geqslant \varepsilon_1/2k^3q^*$ no restriction of t^* belongs to $\{t_{\text{test}}\}$. Hence each monomial μ of point 2 of Lemma 11 gets removed by FILTER.M and replaced by a single term included in μ and belonging to c. Therefore, by Corollary 1 on $\{t_{\text{cand}}\}$, all that is left in h after FILTER.M is the collection of *all* the terms of c of probability measure $\geqslant \varepsilon_1/k$, plus other terms of c of measure $\geqslant \varepsilon_1/2k^3q^*$. Thus $P(c - h) < \varepsilon_1, P(h - c) = 0$, hence $P(c \div h) < \varepsilon_1$. □

4.3. The cost of learning k-term-DNF(n)

In summary, we have proved that our DISCRIMINANT_LEARNING procedure computes accurate hypotheses both in the monotone and in the non-monotone case.

As already mentioned, accuracy is to be intended in the sense of Definition 1 or of Definition 1′ according to whether $\alpha = 0$ in the definition of but (k, q^*) or not. Theorem 6, as well as Corollaries 2 and 3 below, deal with the weaker notion of learnability by Definition 1′. Clearly enough, all these results can be reformulated as learnability results in the sense of Definition 1 for $\alpha = 0$ with no essential modifications: one simply makes head that the asymptotical behavior of the function α *does not affect* the complexity of our algorithms. Thus also in case of fair interactions we can establish these results with *exactly* the same complexity bounds.

Theorem 6. *Let* $\boldsymbol{P}$ *be the family of* but(k, q^*) *probability distributions on* $\boldsymbol{X}_n$, *with* q^* *known. Then the class of k-term-DNF formulas is polynomially poly-relaxedly SQ-learnable w.r.t.* $\boldsymbol{P}$ *using k-term-DNF representation, and the class of monotone k-term-DNF formulas is polynomially poly-relaxedly SQ-learnable w.r.t.* $\boldsymbol{P}$ *using monotone k-term-DNF representation.*

The time to evaluate the queries to the oracle, the inverse of the accuracy of its answers and the time complexity of the learning algorithm is bounded respectively by

– $\mathbf{O}(nk)$, $\mathbf{O}(kq^*/\varepsilon)$ *and* $\mathbf{O}(k2^{k^2}(e/k)^{k^2+k}n^{k^2+1}+(4e/k)^k n^{k+2})$ *in the non-monotone case, and*

– $\mathbf{O}(k)$, $\mathbf{O}(k^5(q^*)^3/\varepsilon)$, $\mathbf{O}(k(e/k)^k n^{k+2} + n^{2k+1}(e/k)^{2k})$ *in the monotone case.*

Proof. The three upper bounds are easily proved by inspection. In more detail:

• non-monotone formulas

By Lemma 8, the required accuracy ε is obtained by our algorithm for $\varepsilon_1 = \varepsilon/4$.

– query complexity. This parameter is $\mathbf{O}(k)$, $\mathbf{O}(k)$, $\mathbf{O}(n)$, $\mathbf{O}(nk)$ and, therefore, still $\mathbf{O}(nk)$ in the queries raised by EMPTY, INFER_SUBTERM, PRUNE, FILTER and the whole procedure, respectively.

– accuracy. Its tightest value is $\varepsilon/8kq^*$ (recall that $\varepsilon_1 = \varepsilon/4$).

– time complexity of the algorithm. Let us assume that *accessing* the STAT oracle requires constant time and that generating a discriminant requires time $\mathbf{O}(n)$. Let T_{Em} be the running time of EMPTY, T_{IS} that of INFER_SUBTERM, T_{Pr} that of PRUNE and T_{Fil} that of FILTER. Then for the global running time τ we have $\tau = \mathbf{O}[nN_k(T_{\mathrm{Em}} + T_{\mathrm{IS}}) + T_{\mathrm{Pr}} + T_{\mathrm{Fil}}]$, where $T_{\mathrm{Em}} = \mathbf{O}(k)$, $T_{\mathrm{IS}} = \mathbf{O}(nk)$, $T_{\mathrm{Pr}} = \mathbf{O}(nN_k 2^k)$ and $T_{\mathrm{Fil}} = \mathbf{O}\left(\sum_{i=0}^{k}\binom{N_k}{i}\right)\mathbf{O}(nk) = \mathbf{O}(k2^{k^2}(e/k)^{k^2+k}n^{k^2+1})$.

• Monotone formulas

By Lemma 12, the required accuracy ε is achieved by our algorithm for $\varepsilon_1 = \varepsilon$.

– query complexity. Now only the first two subroutines query the oracle. So the query complexity is $\mathbf{O}(k)$.

– accuracy. Now its tightest value is $\varepsilon/8k^5(q^*)^3$.

– time complexity of the algorithm. Upon the same assumptions and notations as in the non-monotone case, we have $\tau_{.\mathrm{M}} = \mathbf{O}[nN_k(T_{\mathrm{Em.M}} + T_{\mathrm{IS.M}}) + T_{\mathrm{Pr.M}} + T_{\mathrm{Fil.M}}]$, where $T_{\mathrm{Em.M}} = \mathbf{O}(k)$, $T_{\mathrm{IS.M}} = \mathbf{O}(nk)$, $T_{\mathrm{Pr.M}} = \mathbf{O}(nN_k^2)$ and $T_{\mathrm{Fil.M}} = \mathbf{O}(nN_k^2)$. □

Corollary 2. *Let* $\boldsymbol{P}$ *be the family of* but(k, q^*) *probability distributions on* $\boldsymbol{X}_n$, *with* q^* *known. Then the class of k-term-DNF formulas is polynomially poly-relaxedly PAC learnable w.r.t.* $\boldsymbol{P}$ *using k-term-DNF representation and the class of monotone k-term-DNF formulas is polynomially poly-relaxedly PAC learnable w.r.t.* $\boldsymbol{P}$ *using monotone k-term-DNF representation, with classification noise rate* η *such that* $0 \leqslant \eta \leqslant \eta_b < 1/2$ *by an algorithm that uses:*

- *In the case of non-monotone formulas,*

a sample of size

$$m = \mathbf{O}(k^4(q^*)^3/(\varepsilon^2(1-2\eta_b)^4)\log(n/\delta) + 1/\varepsilon^2 \log[kq^*/\varepsilon\delta(1-2\eta_b)]) \tag{1}$$

and a running time $\tau' = \mathbf{O}(kn^{k+1}m_1 + \tau m_2)$, *where* $m_1 = \mathbf{O}([k^4(q^*)^3/(\varepsilon^2(1-2\eta_b)^4)]\log(n/\delta))$, $m_2 = \mathbf{O}([k^2q^*/(\varepsilon(1-2\eta_b))]^2 \log(n/\delta) + 1/\varepsilon^2 \log[kq^*/\varepsilon\delta(1-2\eta_b)])$ *and* τ *is the same as in Theorem* 6.

- *In the case of monotone formulas,*

a sample of size $m = \mathbf{O}(k^{11}(q^*)^7/[\varepsilon^2(1-2\eta_b)^4]\log(n/\delta)+1/\varepsilon^2 \log[k^5(q^*)^3/\varepsilon\delta(1-2\eta_b)])$ *and a running time* $\tau'_{.M} = \mathbf{O}(kn^{k+1}m + \tau_{.M}m_2)$, *where* $m_2 = \mathbf{O}(k^{11}(q^*)^6/[\varepsilon^2(1-2\eta_b)^2]\log(n/\delta) + 1/\varepsilon^2 \log[k^5(q^*)^3/\varepsilon\delta(1-2\eta_b)])$, *and* $\tau_{.M}$ *is the same as in Theorem* 6.

Proof. We prove only the non-monotone case, the proof of the latter being similar. The claim follows from the following facts:

(i) the sample complexity is that of Theorem 2 plus a term which will be clear in a moment,

(ii) by Lemma 4, the cardinality of the query space Q is $\mathbf{O}(2^{k^2}(e/k)^{k^2+k}n^{k^2}+(4e/k)^k n^{k+1})$,

(iii) the accuracy α of STAT(c,P) is $\mathbf{O}(\varepsilon/kq^*)$,

(iv) the running time concerns two actions: (a) estimation of the actual labelling error rate η,(b) translation of the DISCRIMINANT_LEARNING procedure through the simulation of the oracle STAT(c,P).[7]

(a) *Estimation of* η. In [20] we read that Theorem 2 holds if accuracy $\Delta = \mathbf{O}(\alpha(1-2\eta_b)^2)$ is achieved with confidence $\delta/\#Q$ on the estimation of η.

Lemma 13. *Let* $\lambda_c(X)$ *be the noisy label of* X. *If* $c \neq \boldsymbol{X}_n$ *then for any set* L *of at most* k *literals such that* $\bar{\mu}_L \neq \emptyset$ *then* $P\tilde{}(\lambda_c(X) = 1 | X \in \bar{\mu}_L) \geqslant \eta$. *The equality occurs if and only if* L *is a false discriminant of* $c, P\tilde{}$ *accounting for the randomness of the noisy labelling.*

[7] Point (iv) has already been solved by Kearns in a general context where he has evaluated the sample complexity of Theorem 2. Here we propose a special procedure for estimating η, which is just tailored for our learning problem. This procedure is a bit more expensive in sample complexity than Kearns'. Therefore, we add this cost in (1) to the one reported in Theorem 2. Nevertheless, we prefer our estimate because it requires a much smaller computational time.

Proof. Recall that the labelling error is independent of X. Let L be any set of at most k literals such that $\bar{\mu}_L \neq \emptyset$. Then

$$\begin{aligned}
&P\tilde{}(X \in \bar{\mu}_L, \lambda_c(X) = 1)\\
&\quad = P\tilde{}(X \in \bar{\mu}_L, \lambda_c(X) = 1 | \lambda_c(X) \text{ is correct})\ (1-\eta)\\
&\qquad + P\tilde{}(X \in \bar{\mu}_L, \lambda_c(X) = 0 | \lambda_c(X) \text{ is incorrect})\ \eta\\
&\qquad + (P(X \in c_{\downarrow L^c})\eta - P(X \in c_{\downarrow L^c})\eta)\\
&\quad = P(X \in \bar{\mu}_L)\eta + P(X \in c_{\downarrow L^c})(1-2\eta).
\end{aligned}$$

So $P\tilde{}(\lambda_c(X) = 1 | X \in \bar{\mu}_L) = \eta + P(X \in c_{\downarrow L^c} | X \in \bar{\mu}_L)(1-2\eta)$, where the last term is 0 if and only if L is a false discriminant. Moreover, since we are dealing with a but (k, q^*) distribution, $P(X_i \in \bar{\mu}_L) \geqslant 1/q^*$ for each $\bar{\mu}_L \neq \emptyset$. Thus the conditional probability on the right side of the last equation cannot be made undefined by the impossibility of the conditioning event. □

Lemma 14. *Let $\hat{\eta}$ be an estimate of η, and $\varphi_{\boldsymbol{X}_m^c}(E)$ be the frequency of the event E in drawing a labelled sample $\boldsymbol{X}_m^c$. Let*

$$\hat{\eta} = \min_L \{\varphi_{\boldsymbol{X}_m^c}(\lambda_c(X) = 1 | X \in \bar{\mu}_L)\},$$

where the minimum is taken over all L such that L is a set of at most k literals with $\bar{\mu}_L \neq \emptyset$. Then for a sample size $m_1 = \mathbf{O}((k^4(q^*)^3/[\varepsilon^2(1-2\eta_b)^4])\log(n/\delta))$, $\hat{\eta}$ estimates η with accuracy $\Delta = \mathbf{O}(\alpha(1-2\eta_b)^2)$, with confidence $\delta/\#Q$, and it costs $\mathbf{O}(kn^{k+1}m_1)$ time units.

Proof. If $c \neq \boldsymbol{X}_n$ the claim follows directly from the use of Chernoff's bounds [12] in Lemma 13. If $c = \boldsymbol{X}_n$ then c does not admit false discriminants. Nevertheless, in this case a trivial estimate of η is $\hat{\eta} = 1 - \varphi_{\boldsymbol{X}_m^c}(\lambda_c(X) = 1)$, and the above accuracy can be obtained with a smaller sample and time bounds than in the case $c \neq \boldsymbol{X}_n$.

Thus, in the worst case ($c \neq \boldsymbol{X}_n$), for each subset L of at most k literals with $\bar{\mu}_L \neq \emptyset - \mathbf{O}((en/k)^k)$ in number – we need to check the membership of m_1 examples in $\bar{\mu}_L$. Each check costs k units of computation.

In total, the time necessary to estimate η is $\mathbf{O}(kn^{k+1}m_1)$. □

Proof of Corollary 2 (*conlusion*)

(b) Simulation of DISCRIMINANT LEARNING procedure. Theorem 2 assures that a sample of size m_2 is sufficient for computing an answer of STAT(c, P) to any instance from Q with accuracy $\alpha = \mathbf{O}(\varepsilon/kq^*)$ and confidence $\delta/\#Q$. Thus we essentially scale the resources appraised in Theorem 6 for obtaining the analogous costs of PAC learning by this quantity.

Summing up the cost of the two mentioned actions, we obtain the claim of the corollary.

Actually the estimation of η requires a different algorithm depending on whether $c \neq \boldsymbol{X}_n$ or not. We can start assuming the former case, and we possibly shift to the latter whenever we come to a conclusion that the error of the resulting hypothesis h is too large. The additional cost of this procedure is disregardable. □

Corollary 3. *Let* $\boldsymbol{P}$ *be the family of* but(k, q^*) *probability distributions on* $\boldsymbol{X}_n$, *with* q^* *known. Then*:

- *the class of non-monotone k-term-DNF formulas is polynomially poly-relaxedly PAC learnable w.r.t.* $\boldsymbol{P}$ *using k-term-DNF representation with malicious error rate* $\eta = \mathbf{O}(\varepsilon/kq^*)$ *by an algorithm that needs* $m' = \mathbf{O}((k^2q^*/\varepsilon)^2\log(n/\delta))$ *examples and* $\mathbf{O}(m'\tau)$ *running time, where* τ *is as in Theorem* 6;
- *the class of monotone k-term-DNF formulas is polynomially poly-relaxedly PAC learnable w.r.t.* $\boldsymbol{P}$ *using monotone k-term-DNF representation with malicious error rate* $\eta = \mathbf{O}(\varepsilon/k^5(q^*)^3)$ *by an algorithm that needs* $m'' = \mathbf{O}((k^{11}(q^*)^6/\varepsilon^2)\log(n/\delta))$ *examples and* $\mathbf{O}(m''\tau_{\mathrm{M}})$ *running time, where* τ_{M} *is as in Theorem* 6.

Proof. The proof can be obtained by an analogous use of Theorem 3 in place of Theorem 2, except for an estimate of the malicious error rate, which is unnecessary here. □

It is easy to show that under a uniform distribution and in presence of malicious errors, monotone monomials (1-term-DNF) can be learnt only for error rate $\eta = \mathbf{O}(\varepsilon)$.

The knowledge of q^* is crucial in INFER_SUBTERM for sizing both the tests and their accuracy.

The extension of the previous results to the case where q^* is unknown can be performed by a well-known *generate and test* scheme [26]. We start running the whole DISCRIMINANT_LEARNING algorithm on a low value of the unknown parameter. Then we progressively increase it until we get a positive answer from a procedure, which tests, with a proper set of new examples, the accuracy of the output hypothesis. The *average* sample and time complexity of this strategy – which might loop forever with probability 0 – essentially increases by a factor q^* compared to the case where this parameter is known in advance. For more detail see for instance [15].

4.4. Comparison with the state of the art

The efficiency ratings available in the literature about the concerned learning task are not easily comparable, since they refer to different featured goals, both in terms of distribution laws and concept representations. To our knowledge, the references closest to our framework are the following:

In [15] the authors describe an algorithm for learning non-monotone k-term-DNF(n) formulas working for q-bounded (and with minor changes product) distributions and samples made of both positive and negative examples. Its sample complexity is $\mathbf{O}((kq2^k/\varepsilon)\log(n/\delta))$ and its running time is $\mathbf{O}(((2n)^{k^2}q/\varepsilon)\log(n/\delta))$.

In [31] Sakai and Maruoka show a learning algorithm for monotone k-term-DNF(n) formulas working on positive examples only, with sample complexity $m = \mathbf{O}[(2^{8k}/\varepsilon)(r + \log(d/\delta))]$ and time complexity $\mathbf{O}(dm)$, where $r = \mathbf{O}[(1/\varepsilon + 2^k k^2)^2 \log(1/\delta)]$, $d = \mathbf{O}(nk^2 + k^3)$.

These kinds of results concern learning models where no (classification or malicious) errors corrupt the examples. Thus, for comparison purposes, we can rescale our results by replacing ε^2 with ε in the expressions for sample and time complexity of Corollaries 2 and 3. This is a known way [20] of passing from noisy to error-free examples.

Within this framework, since q-bounded and product distributions are butterfly distributions with $\alpha = 0$, we have proved essentially the same bounds for DISCRIMINANT-LEARNING in the non-monotone case as those proved for the algorithm in [15]. On the other hand, in Section 4.2.1 we worked with a wider family of distribution laws, even with respect to Definition 1.

To learn monotone formulas, when the sample is error-free, the PAC release of our proposed SQ algorithm can come to use only positive examples. This derives from the absence of overestimating errors, as mentioned in Section 4.2.2.

In comparison, Sakai and Maruoka's procedure has better time complexity, since its running time is poly $(2^k, n, 1/\varepsilon, \log(1/\delta))$ (note that it still preserves polynomiality for $k = \mathbf{O}(\log n)$), thanks to the stringent probabilistic assumptions they make about the sample space. This allows them to *infer* the discriminants of the relevant terms of the target formula c directly, instead of simply enumerating all of them. Namely, their algorithm progressively builds every candidate discriminant with variables appearing in some relevant term of c, and at the same time avoids adding *irrelevant* variables (that do not appear in any term of c). In the case of uniform distribution the separation between these two kinds of variables can be easily done, since now $P(\boldsymbol{v} \in c \downarrow_{\{x_i\}} | \boldsymbol{v} \in \mu_{\{x_i\}})$ is equal to 1/2 for an irrelevant variable x_i and is significantly less than 1/2 otherwise. But similar gaps do not exist upon weaker assumptions, neither, for instance, for q-bounded distributions.

Looking at related frameworks, we quote the paper [19] which deals with monotone $k\mu$-DNF formulas ($k\mu$-DNF(n) formulas are DNF(n) formulas, where each variable can appear at most k times where k is a fixed constant. Obviously monotone k-term-DNF(n) $\subseteq$ monotone $k\mu$-DNF(n)) and the recent paper [32] which deals with functions of k terms and shares some ideas with our approach. The former paper refers to product distributions while the latter to p-smooth distributions.

Both papers require two separate polynomially sized samples of positive and negative examples and are based on restriction techniques, but neither of them seems to be directly adaptable to get proper k-term-DNF learning results.

Even though the algorithms we proposed are neither the cheapest (see Definition 9) nor instances of a UMVULP (see Definition 8 and footnote 3), our approach is more statistically based. Our approach relies to a greater extent on statistical induction (queries) for finding relevant regions of the sample space. This makes us to focus our search for the terms of the target formula there, rather than to use the structure of the probability space for addressing the navigation toward these relevant regions in advance.

We remark that the algorithms designed until now for polynomial and proper learning of k-term-DNF formulas [15, 31] refer to uniform, q-bounded and product distributions. By Lemma 2, these distributions cannot dominate the whole family of butterfly distributions. Rather, if we rely on the relaxed notion of learnability in Definition 1′, a wide class of probability distributions prove feasible for learning k-term-DNF formulas. Consider, in particular, the distribution family of Section 3.1, part (d.2) and Remark 1. DISCRMINANT_LEARNING can be translated into a PAC algorithm that yields polynomial and proper poly-relaxed learning of monotone k-term-DNF formulas from samples of partial assignments uniformly drawn from a set of examples with a non-negligible number of assigned values.

5. Conclusions and open problems

We have proposed an algorithm that performs proper learning of k-term-DNF formulas in feasible time, under weak assumptions.

These assumptions concern the distribution law of the set of examples on which the learning is based. Accordingly, we have defined the family of *butterfly* distribution laws, along with a relaxed notion of learnability, covering many real learning instances. The butterfly family describes many useful probabilistic models and assures the feasibility of our algorithm, when its parameters lay in a suitable range.
The importance of the result is both in the use of k-term-DNF formulas in boolean algebra and in the wide applicability of the assumed context.
Our algorithm:

(i) is based on *near*- and p-sufficient statistics; it barters the variance minimality to the polynomial time computability of the estimator;
(ii) tolerates errors in line with the SQ algorithms;
(iii) has sample and time complexity as low as the best available bounds, with a sole exception under very stringent assumptions;
(iv) proves feasible in many probability spaces, according to a reasonable relaxed version of the PAC learning paradigm.

There are at least two plausible extensions of the method we have illustrated in this paper:

(i) treating larger classes of representations, typically $\mathbf{O}(\log n)$-term DNF;
(ii) dealing with larger distribution families.

Both problems are challenging. Such learnability results, which we would consider highly relevant, should exploit properties of the interaction between DNF formulas and underlying distributions that are somewhat subtler than those we have shown here.

The general aim of this paper has been to establish a connection between Computational Learning Theory (which is usually within the scope of Computer Science) and Statistics Theory. We point out that the hypotheses produced by learning algorithms are interesting non-parametric statistics, and we used the results of this branch of Statistics

Theory to improve results in Learning. In this sense we have stressed the importance of the minimality of sufficient statistics, defined the class of Uniform Minimum Variance Unbiased Learning Procedures, given some characteristics of this class and shown that for a given concept class there exists a unique UMVULP when P ranges within the class of all absolutely continuous or all purely discrete distribution functions.

Appendix A

This appendix contains the proofs of Lemmas 1 and 2, whose statements are recalled here with reference to the distribution family $\boldsymbol{P}_\lambda$ defined in part (d.2) of Section 3.1.

Lemma A.1. *Given the Boolean n-dimensional hypercube $\boldsymbol{X}_n$ and $\lambda \in (0, 1/2)$, if n is chosen so large that for constant k we have $\lfloor n\lambda \rfloor > k > 1$ then $P \in \boldsymbol{P}_\lambda$ is* but$(k, \mathbf{O}(n^{2k}))$ *for*

$$[en/(k-1)]^{k-1}\lambda^{\lfloor n\lambda \rfloor} < \alpha/k, \tag{A.1}$$

where e is the base of natural logarithms.

Proof. The cardinality of the support of P in $\boldsymbol{X_n}$ equals $\sum_{i=0}^{\lfloor n\lambda \rfloor} \binom{n}{i}$, where it is easy to see that

$$(1/\lambda)^{\lfloor n\lambda \rfloor} < \binom{n}{\lfloor n\lambda \rfloor} < \sum_{i=0}^{\lfloor n\lambda \rfloor} \binom{n}{i}. \tag{A.2}$$

Let t be a monomial on V of l literals of which u are non-negated and $z = l - u$ are negated variables. If $P(t) \geqslant \alpha/k$ then $u \leqslant \lfloor n\lambda \rfloor - k$. Indeed, let us assume $u > \lfloor n\lambda \rfloor - k$. This means that the number of points in the support of P satisfying t is such that

$$\sum_{i=0}^{\lfloor n\lambda \rfloor - u} \binom{n-l}{i} \leqslant \sum_{i=0}^{k-1} \binom{n-l}{i} < \left(\frac{e(n-l)}{k-1}\right)^{k-1},$$

where the last inequality derives from [33].

By (A.2) each item of the support of P has probability $< \lambda^{\lfloor n\lambda \rfloor}$, so (A.1) yields

$$P(t) < [e(n-l)/(k-1)]^{k-1}\lambda^{\lfloor n\lambda \rfloor} \leqslant [en/(k-1)]^{k-1}\lambda^{\lfloor n\lambda \rfloor} < \alpha/k,$$

a contradiction.

Let I be a set composed of *only one* literal not belonging to set(t). We will distinguish two cases.

Case 1: $z \leqslant n - \lfloor n\lambda \rfloor - 1$ (namely, $n - l - 1 \geqslant \lfloor n\lambda \rfloor - u$).
If the literal in I is a non-negated variable and $t_{\downarrow I} \neq \emptyset$ then

$$P(t_{\downarrow I})/P(t) = \frac{\sum_{i=0}^{\lfloor n\lambda \rfloor - u - 1} \binom{n-l-1}{i}}{\sum_{i=0}^{\lfloor n\lambda \rfloor - u} \binom{n-l}{i}}.$$

Now, set

$$A = \sum_{i=0}^{\lfloor n\lambda\rfloor-u-1} (i!(n-l-1-i)!)^{-1}, \quad B = \sum_{i=0}^{\lfloor n\lambda\rfloor-u} (i!(n-l-1-i)!)^{-1},$$
$$C = ((\lfloor n\lambda\rfloor - u)!(n-l-(\lfloor n\lambda\rfloor - u))!)^{-1}.$$

It is easy to see that $A/B \geqslant 1-(B/C)^{-1}$ and that

$$B/C = 1 + \sum_{i=0}^{\lfloor n\lambda\rfloor-u-1} \frac{(\lfloor n\lambda\rfloor - u)!(n-l-(\lfloor n\lambda\rfloor - u))!}{i!(n-l-i)!} > \frac{n-\lfloor n\lambda\rfloor+2}{n-\lfloor n\lambda\rfloor+1},$$

so that we obtain

$$P(t_{\downarrow I})/P(t) = \frac{A}{B(n-l)} > ((n-l)(n-\lfloor n\lambda\rfloor+2))^{-1} = \frac{1}{\mathbf{O}(n^2)}.$$

A similar algebra occurs to realize that if the literal in I is a negated variable and $t_{\downarrow I} \neq \emptyset$ then the above ratio is at least $(n-l)^{-1}$.

Case 2: $z \geqslant n - \lfloor n\lambda\rfloor$ (namely $n-l \leqslant \lfloor n\lambda\rfloor - u$).

In this case, no matter what the literal contained in I is and provided $t_{\downarrow I} \neq \emptyset$, we have

$$P(t_{\downarrow I})/P(t) = \frac{\sum_{i=0}^{n-l-1} \binom{n-l-1}{i}}{\sum_{i=0}^{n-l} \binom{n-l}{i}} = \frac{1}{2}.$$

Therefore in both cases $P(t_{\downarrow I})/P(t) = 1/\mathbf{O}(n^2)$ and this equality can be trivially extended to the case $I \subseteq \mathrm{set}(t)$.

In case $k \geqslant \#I > 1$, we can factorize the reduction from t to $t_{\downarrow I}$ through a series of $\#I$ reductions, each one involving a single literal of I. Here the hypothesis $\lfloor n\lambda\rfloor > k$ assures that each reduction produces a set whose probability is > 0: each reduction decreases $P(t_{\downarrow I})$ by a factor at most n^2, so we can globally state that $P(t_{\downarrow I}) \geqslant P(t)/q^*, q^* = \mathbf{O}(n^{2k})$, as claimed by the lemma. □

Lemma A.2. *For each $\lambda \in (0,1/2)$ and $P \in \boldsymbol{P}_\lambda$, if n is large enough P cannot be γ-dominated by a p-smooth distribution P_s, for any p and any $\gamma = \mathrm{poly}(n)$.*

Proof.

(i) The number of points of non-zero probability w.r.t. P is $\sum_{i=0}^{\lfloor n\lambda\rfloor} \binom{n}{i} \leqslant 2^{nH(\lambda)}$, where H is the binary entropy $H(\lambda) = -\lambda\log_2\lambda - (1-\lambda)\log_2(1-\lambda)$. Hence, for every point v of $\boldsymbol{X}_n$, if $P(v) > 0$ then $P(v) \geqslant 1/2^{nH(\lambda)}$.

(ii) There exists a $v_0 \in \boldsymbol{X}_n$ such that $P_s(v_0) \leqslant 1/2^n$, since $P_s()$ is always greater than zero. If P were γ-dominated by P_s then we would have $1/2^{nH(\lambda)} < P(v_0) \leqslant \gamma P_s(v_0) \leqslant \gamma/2^n$ which is false for any $\lambda \in (0,1/2), \gamma = \mathrm{poly}(n)$ and n large enough. □

Acknowledgements

We would like to thank anonymous reviewers for many valuable suggestions which greatly contributed to improving the presentation of this paper. In particular, we would like to thank one of them for pointing out an error in our original version of Definition 13 and for her/his comments that led us to explicitly introduce Definition 1' .

References

[1] D. Anguin, Learning k-term-DNF formulas using queries and counterexamples, Technical Report YALEU/RR-559, Yale University, Department of Computer Science, 1987.
[2] D. Anguin, P.D. Laird, Learning from noisy examples, Machine Learning 2 (2) (1988) 343–370.
[3] B. Apolloni, S. Chiaravalli, PAC learning of concept classes through the boundaries of their items, J. Theoret. Comput. Sci., 172 (1997) 91–120.
[4] B. Apolloni, C. Gentile, G. Mauri, Noisy oracles for learning k-term DNF formulas, Proc. AIIA 1994, pp. 38–41.
[5] J.L. Balcazar, J. Diaz, J. Gabarro, Structural Complexity I, Springer, Berlin, 1988.
[6] P.L. Bartlett, R.C. Williamson, Investigating the distribution assumptions in the PAC learning model, Proc. 4th Workshop on Comput. Learning Theory, Morgan Kaufmann, San Mateo, CA, 1991, pp. 24–32.
[7] G. Benedek, A. Itai, Dominating distributions and learnability, Proc. 5th Workshop on Comput. Learning Theory, Morgan Kaufmann, San Mateo, CA, 1992, pp. 253–264.
[8] A. Blum, S. Rudich, Fast learning of k-term-DNF formulas with queries, Proc. 24th Annual ACM Symp. on Theory of Computing, ACM Press, New York, 1992, pp. 382–389.
[9] A. Blum, M. Singh, Learning functions of k terms, Proc. 3rd Workshop on Comput. Learning Theory, Morgan Kaufmann, San Mateo, CA, 1990, pp. 144–153.
[10] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth, Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension, Proc. 18th ACM Symp. on Theory of Computing, ACM Press, New York, 1986, pp. 273–282.
[11] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, J. ACM 36 (1989) 929–965.
[12] A. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, Ann. Math. Statist., 23 (1952) 493–507.
[13] S.E. Decatur, Statistical queries and faulty PAC Oracles, Proc. 6th Workshop on Comput. Learning Theory, Morgan Kaufmann, San Mateo, CA, 1993, pp. 262–268.
[14] A. Ehrenfeucht, D. Haussler, M. Kearns, L. Valiant, A general lower bound on the number of examples needed for learning, Proc. 1988 Workshop on Comput. Learning Theory, Morgan Kaufmann, San Mateo, CA, 1988, pp. 139–154.
[15] M. Flammini, A. Marchetti-Spaccamela, L. Kucera, Learning DNF formulas under classes of probability distributions, Proc. 5th Workshop on Comput. Learning. Theory, Morgan Kaufmann, San Mateo, CA, 1992, pp. 85–92.
[16] D.A.S. Fraser, Nonparametric Methods in Statistics, Academic Press, New York, 1967.
[17] M.R. Garey, D.S. Johnson, Computer and Intractability, Freeman, San Fransisco, CA, 1979.
[18] Q.P. Gu, A. Maruoka, Learning monotone Boolean functions by uniformly distributed examples, SIAM J. Comput. 21(3) (1992) 587–599.
[19] T. Hancock, Y. Mansour, Learning monotone kμ-DNF formulas on product distributions, Proc. 4th Workshop on Comput. Learning Theory, Morgan Kaufmann, San Mateo, CA, 1991, pp. 179–183.
[20] M. Kearns, Efficient noise-tolerant learning from statistical queries. Proc. 25th Annual ACM Symp. on Theory, of Computing, ACM Press, New York, 1993, pp. 392–401.
[21] M. Kearns, M. Li, Learning in the presence of malicious errors, Proc. 20th Annual ACM Symp. on Theory of Computing, ACM Press, New York, 1988, pp. 267–280.
[22] M. Kearns, M. Li, L. Pitt, L. Valiant, On the learnability of boolean formulae, Proc. 19th ACM Symp. on Theory of Computing, ACM Press, NY, 1987, pp. 285–295.

[23] Lehmann E.L., Theory of Estimation, University of California Press, Berkeley, CA, 1950.
[24] E.L. Lehmann, H. Scheffé H. Completeness, similar regions, and unbiased estimation, Sankhya Ser. A 10 (1950) 305–340.
[25] N. Linial, Y. Mansour, N. Nisan, Constant depth circuits, Fourier transform, and learnability, Proc. 30th Annual Symp. on Foundation of Computer Science, IEEE Computer Soc. Press, Silver Spring, MD, 1989, pp. 574–579.
[26] Nilsson, Principles of Artificial Intelligence, Springer, New York, 1982.
[27] L. Pitt, L. Valiant, Computational limitations on learning from examples, J. ACM 35 (4) (1988) 965–984.
[28] R. Rivest, Learning decision-lists, Machine Learning 2 (3) (1987) 229–246.
[29] V.K. Rohatgi, An Introduction to Probability Theory and Mathematical Statistics, Wiley, New York, 1976.
[30] Y. Sakai, A. Maruoka, Learning k-term monotone Boolean formulae, Proc. 3rd Workshop on Algorithmic Learning Theory, 1992, pp. 197–207.
[31] Y. Sakai, A. Maruoka, Learning monotone log-term DNF formulas, Proc. 7th Workshop on Comput. Learn. Theory, Morgan Kaufmann, New Brunswick, NJ, 1994, pp. 165–172.
[32] Y. Sakay, E. Takimoto, A. Maruoka, Proper learning algorithm for functions of k-terms under smooth distributions, Proc. 8th Workshop on Comput. Learn. Theory, Morgan Kaufmann, Santa Cruz, CA, 1995, pp. 206–213.
[33] N. Sauer, On the density of families of sets, J. Combin. Theory Ser. A 13 (1972) 145–147.
[34] R. Solomonoff, A formal theory of inductive inference, Inform. Control 7 (1964) 1–22.
[35] L.G. Valiant, A theory of the learnable, Commun. ACM 27 (11) (1984) 1134–1142.
[36] L.G. Valiant, Learning disjunctions of conjunctions, Proc. 9th Int. Joint Conf. on A.I., vol. 1, Morgan Kaufmann, San Mateo, CA, 1985, pp. 560–566.
[37] V. Vapnik, Estimation of Dependences Based on Empirical Data, Springer, New York, 1982.
[38] V. Vapnik, A.Ya. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theoret. Prob. Its Appl. 16(2) (1971) 264–280.
[39] K. Verbeurgt, Learning DNF under the uniform distribution in quasi-polynomial time, Proc. 3rd Workshop on Comput. Learn. Theory, Morgan Kaufmann, San Mateo, CA, 1990, pp. 314–326.
[40] I. Wegener, The Complexity of Boolean Functions, Wiley-Teubner Series in Computer Science, 1987.
[41] S.S. Wilks, Mathematical Statistics, Wiley, New York, 1962.