Electronic Colloquium on Computational Complexity, Report No. 114 (2017)

Proper Learning of k-term DNF Formulas from Satisfying Assignments

Maciej Liśkiewicz, Matthias Lutter, and Rüdiger Reischuk

Institut für Theoretische Informatik, Universität zu Lübeck, Germany
{liskiewi,lutter,reischuk}@tcs.uni-luebeck.de

Abstract. In certain applications there may only be *positive* samples available to to learn concepts of a class of interest, and this has to be done *properly*, i. e. the hypothesis space has to coincide with the concept class, and without false positives, i. e. the hypothesis always has be a subset of the real concept (one-sided error). For the well studied class of k-term DNF formulas it has been known that learning is difficult. Unless RP = NP, it is not feasible to learn k-term DNF formulas properly in a distribution-free sense even if both positive and negative samples are available and even if false positives are allowed.
This paper constructs an efficient algorithm that for arbitrary fixed k, if samples are drawn from distributions like uniform or q-bounded ones, properly learns the class of k-term DNFs without false positives from positive samples alone with arbitrarily small relative error.

Keywords: algorithmic learning, learning from positive samples, q-bounded distributions, k-term DNF formulas

1 Introduction

Boolean formula in DNF is a common way for humans to describe a concept over binary attributes. In particular, using DNF seems to be more natural than expressing a concept in CNF and, in contrast to CNF representation, DNF allows simple generation of attribute values of belonging objects. Since humans learn quickly such concepts – usually from positive examples only – one could have expected that they are also algorithmically learnable.

A systematic study on learnability of concepts described by DNF formulas was initiated by Valiant in his seminal paper [29] formalizing the Probably Approximately Correct (PAC) model of learning. The question whether DNF formulas can be learned efficiently in the PAC and related models, has been then the subject of a considerable amount of research, to name just a few [22, 23, 28, 21, 11, 13, 20, 26, 30, 1, 3, 16, 27, 9, 5, 7]. However, despite intensive research it remains an important open problem whether this class can be learned in polynomial time – even under the uniform distribution. The fastest known algorithm so far for learning (polynomial size) DNF formulas in the standard distribution-free PAC model is due to Klivans and Servedio [16]. It runs in time $2^{\tilde{\mathcal{O}}(n^{1/3})}$ for formulas with n variables. For the case that samples are drawn uniformly from $\{0,1\}^n$ Verbeurgt [30] has shown that DNF can be learned in time roughly $n^{\mathcal{O}(\log n)}$.

On the other hand, negative results have shown that in a distribution-free PAC model it is not feasible to learn even k-term DNF formulas properly (the hypothesis

ISSN 1433-8092

space has to coincide with the concept class) for every fixed $k \geq 2$ unless $RP = NP$. Learning k-term DNF concepts for $k \geq 4$ remains infeasible even if allowing as hypothesis $f(k)$-term DNF for any function $f(k) \leq (2k-3)$ [23].

In this paper we investigate proper learning without false positives for DNF formulas from positive samples. This setting has originally been studied by Valiant in [29] posing the following problem: *"The question as to whether monotone DNF expressions can be learned from [positive] EXAMPLES alone is open. A positive answer would be especially significant."* In that paper an efficient learner has been constructed for 1-term DNFs, i. e. for monomials, even for the nonmonotone case. Valiant's underlying model is the standard (ϵ, δ)-PAC model with the following additional constraints:

(a) the algorithm has to learn a target formula $\varphi(x)$ given only satisfying assignments x that are produced with probability $D(x)/D(\varphi^{-1}(1))$, where D is an (unknown) distribution on the set $\{0,1\}^n$ of all possible assignments,
(b) the hypothesis h generated belongs to the same class as φ (proper learnability),
(c) h does not have false positives, i. e. $h^{-1}(1) \subseteq \varphi^{-1}(1)$,
(d) h achieves small error probability for false negatives, i. e. with probability at least δ it holds $D(h^{-1}(0) \cap \varphi^{-1}(1)) \leq \epsilon$.

Notice the importance of requiring both the learning from positive samples alone and the proper learnability. For example, in some applications, negative samples are difficult or impossible to collect while positive samples may be obtained easily. Similarly, in machine learning settings, getting natural counter-examples in the training phase may be problematic. Proper learning plays a significant role, especially in the context of DNFs. It has been well known that k-term DNF formulas can be learned efficiently from positive samples if k-CNF formulas as hypotheses are allowed. However, such a k-CNF representation may be useless in specific applications. Also learning with one-sided error is a important property required in certain cases. For example, the combination of proper learning from positive samples without false positives shows up in steganography (see [19] or [10] for more discussion).

In [22] Natarajan gives a complete characterization of concept classes that are (polynomial time) learnable in the model defined above. His result implies that even monotone DNFs, bounded k-DNFs, and k-term DNFs, for $k \geq 2$, are *not* efficiently proper PAC-learnable without false positives from positive samples. One property of the learning model that makes it very difficult is the requirement that the learner has to perform well for *every* probability distribution D. Thus, an adversary may pick a different probability distribution for the samples for every concept which assigns high probabilities to the most "difficult" assignments.

If one wants to overcome these impossibility results and restricts the set of possible distributions an extreme case would be to fix a single distribution, naturally the uniform one on $\{0,1\}^n$, as it has been done in several subsequent papers. Here, for an arbitrary value $q \geq 1$ we will consider the quite large family of *q-bounded* distributions D where the ratio of the probabilities is bounded by q, that means $\max_x D(x) \leq q \cdot \min_x D(x)$.

Flammini, Marchetti-Spaccamela, and Kučera have proven that 2-term DNFs are proper learnable in polynomial time from satisfying assignments for q-bounded

distributions [12]. Recently De, Diakonikolas, and Servedio [7] have shown that DNF formulas have efficient learning algorithms using uniformly distributed positive samples, but instead of a k-term DNF hypothesis the learner outputs a *sampler* for $\varphi^{-1}(1)$ only that may have two-sided errors.

There are also positive results for monotone DNF (MDNF) formulas, i. e. those without negated variables, for the uniform distribution. By results of Sakai and Maruoka [24] we know that proper learning is possible for log-term MDNF formulas from positive assignments alone. Moreover, the class of k-term MDNFs can even be learned for q-bounded distributions [25, 17]. However, the nonmonotone case has remained open.

This paper addresses this problem and considers a harder requirement than condition (d) for the error probability of false negatives, which in particular for k-term DNFs makes more sense. Considering the *absolute error* probability $D(h^{-1}(0) \cap \varphi^{-1}(1))$ is the most commonly used definition in the literature on learning from positive samples, also in the setting allowing two-sided error where one has also to take into account $D(h^{-1}(1) \cap \varphi^{-1}(0))$ (see e. g. [22, 8, 9]). Instead, we will estimate the *relative error* probability

(d') $D(h^{-1}(0) \cap \varphi^{-1}(1)) \,/\, D(\varphi^{-1}(1))$,

that is with respect to satisfying assignments only. This criterion has already been used in learning from positive samples, for example in [28]. Moreover, this definition essentially corresponds to the *total variation distance* used in [7] to measure the distance between the uniform distribution $\mathcal{U}_{\varphi^{-1}(1)}$ over the satisfying assignments for φ and the distribution S of the sampler learned defined as $d_{\mathrm{TV}}(S, \mathcal{U}_{\varphi^{-1}(1)}) := \frac{1}{2} \sum_{x \in \{0,1\}^n} |S(x) - \mathcal{U}_{\varphi^{-1}(1)}(x)|$. As noted in [7], this measure is more appropriate than the standard one. For example, in uniform-distribution learning the constant 0 function is an acceptable hypothesis for every function φ with $|\varphi^{-1}(1)| \leq \epsilon\, 2^n$. In contrast, the condition (d') gives a reasonable error measure also for small $|\varphi^{-1}(1)|$. For a detailed discussion of error measures for DNF see the next section.

1.1 Our Contribution

The main result of this paper says that for the family of q-bounded distributions for every fixed k there exists an efficient learner for k-term DNF formulas with properties (a)–(d').

The major challenge already occurs for the uniform distribution since false positives cannot be tolerated at all. Our solution works in two phases taking two separate batches of samples. In the first phase k-term DNF formulas are learned with very high accuracy and without false positives using k-CNF representations.

In the second phase, we construct a set of *maximal monomials* that should cover most of the area of this k-CNF formula. The number of candidates for these monomials can be extremely large. Therefore, we design a mechanism to select a suitable subset. This subset may still contain many more than k monomials. Finally, we apply tests with a second sequence of positive samples to select a subset of size at most k as final hypothesis.

Comparing our learner with the result in [7] for learning a k-term DNF φ from uniformly distributed satisfying assignments, note that their algorithm learns a sampler for a distribution S with $d_{\mathrm{TV}}(S, \mathcal{U}_{\varphi^{-1}(1)}) \leq \epsilon$, while our method returns a k-term DNF formula h, with

$$D(h^{-1}(0) \cap \varphi^{-1}(1)) \,/\, D(\varphi^{-1}(1)) \;=\; \Pr[h(x) \neq \varphi(x) \mid \varphi(x) = 1] \leq \epsilon \;,$$

both with probability $\geq 1 - \delta$ (recall $h^{-1}(1) \subseteq \varphi^{-1}(1)$). Such a representation h can always be used as an efficient sampler to generate the uniform distribution $\mathcal{U}_{h^{-1}(1)}$ (see [14]). In addition, the total variation distance between this distribution and the uniform distribution $\mathcal{U}_{\varphi^{-1}(1)}$ satisfies:

$$d_{\mathrm{TV}}(\mathcal{U}_{h^{-1}(1)}, \mathcal{U}_{\varphi^{-1}(1)}) \;=\; \Pr_{x \in \mathcal{U}_n}[h(x) \neq \varphi(x) \mid \varphi(x) = 1],$$

where $\mathcal{U}_n$ denotes the uniform distribution on $\{0,1\}^n$. Thus, our algorithm is more general than the method of [7] and it is not clear whether it can be extended to q-bounded distributions, however, it is faster.

As a negative result, we show that it is impossible to learn unrestricted DNF formulas without false positives. For q-bounded distributions learning n-term DNF formulas requires an exponential number of positive samples regardless of the hypothesis space. An overview of the current state of knowledge concerning DNF learning is given in Table 1.

concept class	distribution-free	uniform / q-bounded
1-term DNF (monomials)	yes [29]	yes [29]
2-term DNF	no [22]	yes [12]
k-term DNF	no [22, 23]	yes (Theorem 1)
log-term DNF	no [23]	open
unrestricted DNF	no [23]	no (Theorem 2)

Table 1: Positive and negative results for proper learning of DNF formulas from positive samples over several distributions in polynomial time. The negative results of [23] (unless $RP = NP$) for k-term, with $k \geq 3$, log-term, and unrestricted DNFs hold even for learning from positive and negative samples.

1.2 Related Work

Learning of DNF formulas is too vast for a detailed survey here, so we focus on the most related results for settings with samples drown from specific distributions including the most prominent research area – the uniform-distribution DNF learning.

If the number of terms of the DNFs may grow arbitrarily, due to Verbeurgt [30] we know that n-term DNF formulas over the uniform distribution (both with positive and negative samples) can be learned using a polynomial number of samples in quasi-polynomial time $n^{\mathcal{O}(\log n)}$. However, the hypothesis space has to be extended to $(n \cdot t)$-term DNF with t depending on the sample complexity.

For monotone DNFs faster algorithms are known. Sakai and Maruoka [24] gave a polynomial-time algorithm for log-term MDNFs under uniform distribution and

Bshouty and Tamon [6] extended these result to q-bounded distributions and for learning formulas including $\mathcal{O}(\log^2(n)/\log\log(n))$-term MDNFs. Servedio [27] has improved these results showing that k-term MDNF formulas are learnable to accuracy ϵ in time polynomial in $(k \cdot \log(nk/\epsilon))^{\log(k/\epsilon)\cdot\log(1/\epsilon)}$ and n. In particular, this algorithm can learn $\mathcal{O}(2^{\sqrt{\log n}})$-term MDNFs in polynomial time to any constant accuracy.

In [18] Linial, Mansour, and Nisan have introduced a powerful Fourier transform-based learning technique to learn under the uniform distribution. Particularly, it allows learning depth d circuits in time roughly $n^{\log^d n}$. This technique has been used intensively in subsequent years to prove important results on learning of DNFs under the uniform distribution. In [20] Mansour, based on Fourier analysis, gave an $O(n^{\log\log n})$ time bounded learning algorithm for DNFs under the uniform distribution in the PAC model in which the learner can also ask *membership queries*. Finally, Jackson [13] has designed a polynomial time learning algorithm for DNFs in this model. In contrast, Angluin and Kharitonov [3] have shown that for the distribution-free setting membership queries do not help to PAC-learn DNF.

The rest of this paper is organized as follows. The next section gives the basic facts about monomials needed later and a formal definition of the learning model. The learning algorithm will be presented in section 3. Its analysis follows in the next sections. Section 7 gives the lower bound. We conclude with a question whether efflcient learning of k-term DNF formulas can be extended to the case of small growing k.

2 Preliminaries and Definitions

Let us start with some basic definitions. In the following, n will always denote the number of Boolean variables $x_1, \ldots, x_n$ and $\mathcal{X} = \{0,1\}^n$ the set of possible assignments to these variables and D be a distribution over $\mathcal{X}$. D is called *q-bounded* for $q \geq 1$ if

$$\max\{D(x)\} \;\leq\; q \cdot \min\{D(x)\} \;.$$

Let $\mathcal{D}_q$ denote the family of all q-bounded distributions. Note that for $q = 1$ we get the uniform distribution on $\mathcal{X}$.

For a Boolean formula φ let $\mathtt{sat}(\varphi) := \varphi^{-1}(1) = \{x \in \mathcal{X} \mid \varphi(x) = 1\}$ denote the set of assignments that satisfy φ, which will also be called the *support* of φ. When learning from positive assignments without false positives the q-bounded condition can even be relaxed. The ratio q only has to hold for $x \in \varphi^{-1}(1)$ since the probability on $\varphi^{-1}(0)$ is totally irrelevant. Considering $\mathcal{X}$ as an n-dimensional cube the support of a monomial M over $x_1, \ldots, x_n$ will always be a subcube. We may assume that no monomial contains a literal twice or becomes *trivial* by containing a variable and its complement. Two monomials M and M' are considered *identical* if $\mathtt{sat}(M) = \mathtt{sat}(M')$ – that means they have the same set of literals. A monomial of length ℓ, that means consisting of ℓ variables, has a support of size $2^{n-\ell}$.

For a k-term DNF φ consisting of monomials $M_1, \ldots, M_k$ of length $\ell_1, \ldots, \ell_k$ the size of $\varphi^{-1}(1)$ can be most $2^n \sum 2^{-\ell_i}$. If the distribution D on $\mathcal{X}$ is q-bounded

then $D(M_i^{-1}(1)) \leq q\ 2^{-\ell_i}$. Omitting all $M_i^{-1}(1)$ in a hypothesis h for φ for all M_i with $D(M_i^{-1}(1)) \leq \epsilon/2k$ can generate an absolute error of at most $\epsilon/2$. Thus, in order to satisfy condition (d) it suffices to learn only those monomials M_i in φ with $\ell_i \leq \log(2\ q\ k/\epsilon)$. However, for bounded k the number of such monomials is polynomially bounded which significantly simplifies the learning problem. In the extreme case, where every ℓ_i is large the DNF formula identical 0 already achieves small absolute error at most ϵ. Thus, to exclude this trivial solution and to get a much better precision it makes sense to consider the relative error as defined in (d').

Definition 1. *A monomial M is* shorter *(with respect to its length) than a monomial M' – also called* larger *(with respect to its support) – if M consists of less literals than M' – which by excluding trivial monomials is equivalent to $|\mathtt{sat}(M)| > |\mathtt{sat}(M')|$.*

On the other hand, a proper submonomial *of M is obtained by removing some literals from M, thus enlarging the support.*

Note that a proper submonomial of M is always shorter and larger, but a shorter monomial does not have to be a submonomial. The following notion plays important role in our main algorithm.

Definition 2. *Let ψ be a Boolean formula and $x \in \mathtt{sat}(\psi)$. A monomial M is* (ψ, x)-maximal *if $x \in \mathtt{sat}(M) \subseteq \mathtt{sat}(\psi)$ and there is no proper submonomial of M fulfilling this condition.*

Let $\mathcal{S}(\psi, x)$ denote the set of all (ψ, x)-maximal monomials. For an integer $c > 0$ we call a subset $\mathcal{S}' \subseteq \mathcal{S}(\psi, x)$ a (ψ, x, c)-set *if $|\mathcal{S}'| = \min\{c, |\mathcal{S}(\psi, x)|\}$, that means $\mathcal{S}'$ contains c many maximal monomials or all if their number is less than c.*

A k-term DNF formula φ is a disjunction of at most k monomials. φ is called *non-redundant* if it does not contain a monomial M such that removing M from φ does not change $\mathtt{sat}(\varphi)$, in particular there are no identical or trivial monomials. Throughout the whole paper we consider only non-redundant DNF formulas. The support of a k-term DNF is the union of at most k subcubes.

A k-CNF formula ψ is given by a conjunction of clauses each containing at most k literals. We may assume that no clause contains a literal more than once or a variable and its negation (a trivial clause). Also no clause should be empty which would make ψ unsatisfiable. Similarly to DNF formulas, a CNF formula fulfilling these properties is called *non-redundant* and we will consider only such CNF formulas. The support of a clause with k literals is the complement of a subcube of dimension $n - k$. This implies that the support of a k-CNF formula ψ is the intersection of such complementary subcubes. It may become arbitrarily complex. This geometric intuition will be helpful for the following investigations.

In this paper we consider the family of concept classes

$$\{\mathtt{sat}(\varphi) \mid \varphi \text{ is a } k\text{-term DNF formula over } \mathcal{X}\}$$

and proper learning of these classes from positive samples as discussed in the introduction, i. e. the learner seeing only satisfying assignments has to output a k-term

DNF formula that is *close* to φ and has no false positives (condition (a)–(d')). Formally, we make the following

Definition 3. *A learner* learns a concept class $\mathcal{C}$ from positive samples without false positives *with respect to q-bounded distributions if given the parameter q for the distributions and error parameters ε, δ, for every pair (C, D) of a concept $C \in \mathcal{C}$ and a distribution $D \in \mathcal{D}_q$, getting positive samples x for C that are chosen independently with probability $D(x)/\sum_{y \in C} D(y)$ its output hypothesis h satisfies $h \subseteq C$, and with probability at least $1 - \delta$, $D(C \setminus h) \,/\, D(C) \leq \epsilon$.*

A concept class $\mathcal{C}$ can be learned efficiently for bounded distributions if a learner exists with running time polynomial in $(1/\varepsilon, 1/\delta, n, q)$.

3 The Main Algorithm

Throughout this paper φ will always denote a k-term DNF formula representing the concept to be learned.

In [12], Flammini et al. presented a method for proper learning such concepts from positive *and* negative samples: In a first phase candidate monomials are selected based on positive samples such that every monomial that is part of φ and is satisfied by enough of these positive samples becomes a member of this set of candidates. But their number in general is larger than k. In the second phase, combinations of at most k candidate monomials are tested against a set of positive and negative samples. If such a combination fulfills a specific error bound then it becomes the output. It has been shown that with high probability this yields an approximately correct hypothesis.

In the following we describe an alternative approach that can handle the lack of negative samples. Getting only positive samples our learner learns properly, i. e. computes a k-term DNF formula as a hypothesis, and such a hypothesis does not include false positives, but is still close to the real concept.

We start by constructing a k-CNF formula that represents a first batch of positive samples. Then the main technical difficulty is to construct appropriate monomials from this CNF formula. This strategy in pseudo code is listed as Algorithm 1.

It has been shown how to learn a k-term DNF formula φ without false positives by using as hypothesis space k-CNF formulas [23, 29, 4]. In this case $((2n)^{k+1} - \ln \delta)/\varepsilon$ positive samples are needed. The learner starts with the conjunction of all possible non-tautological clauses of length at most k, of which there are at most $(2n)^{k+1}$. Then clauses not satisfied by a positive sample are deleted. This strategy is used in step 4 of `Learn-`k`-Term-DNF`. Fig. 1 illustrates such a hypothesis ψ for a target formula φ consisting of 3 monomials.

Our first innovation constructs candidate monomials for φ by extracting monomials from ψ. We choose monomials M with $\mathtt{sat}(M) \subseteq \mathtt{sat}(\psi)$ as large as possible. Generally, for $k \geq 3$ it is NP-hard to find a single satisfying assignment for a k-CNF formula. But here we already know a number of satisfying assignments, namely the positive samples used to create ψ. For this purpose, we define a criterion for potential candidate monomials generated from ψ and a sample $x \in \mathtt{sat}(\psi)$: M has to be (ψ, x) maximal. An illustration is given in Fig. 1 (Right).

Algorithm 1: `Learn-`k`-Term-DNF`$(\varepsilon, \delta, k, q, EX)$

Input: parameters ε, δ, k, q
sampling oracle EX for a target k-term DNF φ
Output: k-term DNF hypothesis φ'

1 $\varepsilon_1 \longleftarrow \varepsilon\, q^{-1}\, k^{-1}\, 2^{-(3k+1)}$;
2 $N_1 \longleftarrow \varepsilon_1^{-1}\, ((2n)^{k+1} + \ln(2/\delta))$;
3 draw N_1 samples $E = (e_1, \ldots, e_{N_1})$ using EX;
4 learn k-CNF formula ψ using samples in E;
5 $\mathcal{M} \longleftarrow \emptyset$;
6 **for** $j \longleftarrow 1$ **to** N_1 **do**
7 | generate a $(\psi, e_j, 2^k - 1)$-set $\mathcal{M}_j$;
8 | $\mathcal{M} \longleftarrow \mathcal{M} \cup \mathcal{M}_j$;
9 **end**
10 reduce $\mathcal{M}$ to the $(2^k - 1)$-shortest monomials;
11 $N_2 \longleftarrow 48\, \varepsilon^{-2}\, \ln(2^{k^2+2}/\delta)$;
12 draw N_2 samples $S = (s_1, \ldots, s_{N_2})$ using EX;
13 **foreach** subset W of $\mathcal{M}$ of size at most k **do**
14 | $\varphi_W := \bigvee_{M \in W} M$;
15 | **if** φ_W misclassifies less than $3\varepsilon N_2/4$ samples of S
| **then return** $\varphi' := \varphi_W$;
16 **end**
17 **return** $\varphi' := \bigvee_{M \in W} M$ for some $W \subseteq \mathcal{M}$ of size k;

In step 7 of `Learn-`k`-Term-DNF` for every positive sample e_i used to learn the k-CNF formula ψ has to select several (ψ, e_i)-maximal monomials, but at most $2^k - 1$ many. While it is not difficult to provide an algorithm which finds a single (ψ, e_i)-maximal monomial, the task to construct a $(\psi, e_i, 2^k - 1)$-set efficiently seems to be highly nontrivial. In Section 4 we present a strategy for solving this problem and provide its analysis.

After finishing the for-loop over all positive samples $e_1, e_2, \ldots$ (steps 6–9), the learner might get a very large set of maximal monomials $\mathcal{M} = \bigcup_j \mathcal{M}_j$, where $\mathcal{M}_j$ denotes the $(\psi, e_j, 2^k - 1)$-set for a sample e_j. Thus, a method is needed to reduce their number. To obtain a smaller number of candidates to continue with we try to prune the set $\mathcal{M}$ without losing too many satisfying assignments. To this aim every monomial of the unknown k-term DNF formula φ that has a large support should become a candidate monomial. On the other hand, monomials with a small support might be removed without losing much accuracy. This selection performed in step 10 reduces the number of maximal monomials in $\mathcal{M}$ to $2^k - 1$.

Finally, to construct the resulting k-term DNF formula φ', the learner chooses a subset of k monomials in $\mathcal{M}$ such that the formula φ' consisting of these monomials misclassifies a small fraction of fresh samples. This learner achieves the following properties.

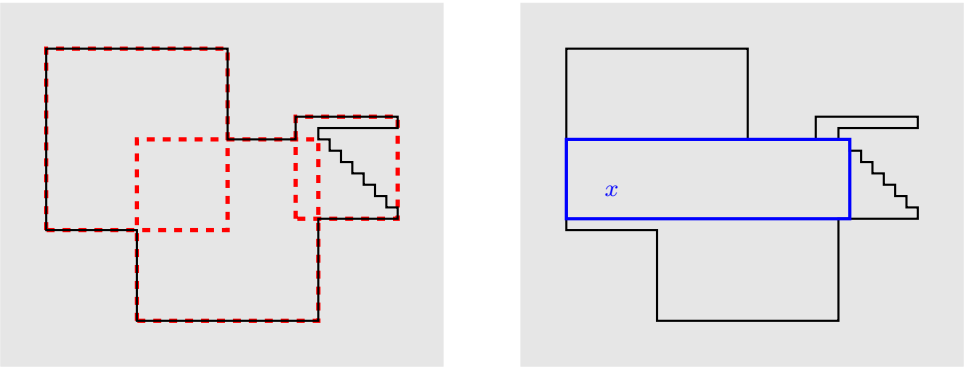

Fig. 1: A running example for $k = 3$. The grey region illustrates the set $\mathcal{X}$ of all possible assignments.
Left: The support of a target 3-term DNF φ is indicated by dashed red rectangles. The support of the 3-CNF hypothesis ψ that approximates φ is surrounded by the solid black line. On the right a scattered region (like a staircase) is generated that requires many rectangles to be covered.
Right: The blue rectangle denotes a (ψ, x)-maximal monomial that is bounded to the right by the scattered region.

Theorem 1. *For every constant k,* `Learn-k-Term-DNF` *learns k-term DNF formulas without false positives over q-bounded distributions with sample size bounded by*

$$\sigma(\varepsilon, \delta, n, k, q) \;:=\; \varepsilon^{-1}\, q\, k\, 2^{3k+1} \left((2n)^{k+1} + \ln(2/\delta)\right) + 48\varepsilon^{-2}\, \ln\left(2^{k^2+2}/\delta\right)$$

and time complexity polynomial with respect to $(1/\varepsilon, 1/\delta, n, q)$.

The procedure for generating maximal monomials from a CNF formula will be discussed in Section 4. In Section 5 we will show that it suffices to select only $2^k - 1$ maximal monomials. Finally, in Section 6 the correctness proof and complexity analysis of the whole algorithm will be presented to complete the proof of Theorem 1.

4 Generating Maximal Monomials for a CNF Formula

Let ψ be a non-redundant k-CNF formula over variables $x_1, \ldots, x_n$ with a satisfying assignment x and c a natural number. First, let us consider Algorithm 2 that finds a single (ψ, x)-maximal monomial. It will be used as a subroutine in Algorithm 3 to compute a (ψ, x, c)-set.

`SingleMaxMonomial` starts by removing all literals from ψ that are not satisfied by x. Then each variable appears only in one form – either positive or negative. Beginning with the constant monomial $M \equiv 1$, we gradually add literals to M that are left in the truncated version of ψ until $\mathtt{sat}(M) \subseteq \mathtt{sat}(\psi)$ becomes true.

Lemma 1. *For fixed k and an arbitrary k-CNF formula ψ and $x \in \mathtt{sat}(\psi)$ the algorithm* `SingleMaxMonomial`(ψ, x) *outputs a (ψ, x)-maximal monomial.*

Algorithm 2: `SingleMaxMonomial`(ψ, x)

Input: k-CNF formula ψ; assignment $x \in \mathtt{sat}(\psi)$
Output: some (ψ, x)-maximal monomial M

```
1  remove every literal from ψ that is not satisfied by x;
2  M ⟵ 1;
3  while true do
4  |  while there exists a clause in ψ consisting of a single literal ℓ do
5  |  |  M ⟵ (M ∧ ℓ);
6  |  |  remove all clauses from ψ that contain ℓ;
7  |  end
8  |  if ψ is empty then return M;
9  |  select an arbitrary literal ℓ from ψ;
10 |  remove ℓ from every clause in ψ;
11 end
```

Its run time is bounded by a polynomial $p_k(n)$ depending only on the number of variables in ψ.
Moreover, for every (ψ, x)-maximal monomial M one can select a sequence of literals in line 9 such that the output becomes M.

Proof. The number of clauses in ψ is bounded by $(2n)^k$. At most n different literals are left after performing line 1, and each iteration of the external while-loop removes at least one further literal. Thus, the total number of operations is bounded by $\mathcal{O}(n^{k+1})$.

If `SingleMaxMonomial`(ψ, x) returns M then we have to show that it is a (ψ, x)-maximal monomial. After performing line 1 the algorithm composes M only from literals that are satisfied by x. Thus, $x \in \mathtt{sat}(M)$.

A literal ℓ is added to M (line 5) only if ℓ is a single literal left in a clause K. We will show that this guarantees the maximality of the resulting monomial M. If ℓ were removed from M then the assignment $\bar{x}$ obtained from x by negating ℓ and the values of all variables that do not appear in M does not satisfy the initial formula ψ. This holds since all the other literals initially being in K have been removed – either in the first step because not being satisfied by x or in line 10. The later ones are complemented in $\bar{x}$ because they do not show up in M, and thus cannot satisfy K. Thus, there is no proper submonomial M' of M with $\mathtt{sat}(M') \subseteq \mathtt{sat}(\psi)$.

The algorithm terminates when ψ is empty. This means that every clause of the original CNF formula is satisfied by the literals in M. Thus, $\mathtt{sat}(M) \subseteq \mathtt{sat}(\psi)$.

Finally, we prove that for every (ψ, x)-maximal monomial $\hat{M}$ there exists a run of the algorithm such that its output M in line 8 equals $\hat{M}$. A maximal monomial $\hat{M}$ can only contain literals that appear in ψ, otherwise it could be shortened and would not be maximal. The condition $x \in \mathtt{sat}(\hat{M})$ implies all literals in $\hat{M}$ must be satisfied by x, thus none will be removed in line 1.

Fact 1. *Every clause K of ψ has at least one literal ℓ that belongs to $\hat{M}$.*

Proof. A similar argument as above shows that otherwise $\texttt{sat}(\hat{M}) \subseteq \texttt{sat}(\psi)$ would be violated. □

Fact 2. *For every literal ℓ of $\hat{M}$ there exists at least one clause $K = (\ell \vee \ell_1 \vee \cdots \vee \ell_j)$ in ψ such no other literal ℓ_i in K belongs to $\hat{M}$, too.*

Proof. Otherwise $\hat{M}$ would not be a maximal monomial because one could remove ℓ from $\hat{M}$. □

Now consider the following run of `SingleMaxMonomial`:

in line 9 always select as ℓ a literal that does not appear in $\hat{M}$.

Since every clause has at least one literal of $\hat{M}$, removing literals not in $\hat{M}$ will end up in nonempty clauses that have only literals belonging to $\hat{M}$. By Fact 2, every literal ℓ of $\hat{M}$ has a clause where it is the unique one of $\hat{M}$. Thus, such a clause will force ℓ to be selected for the output monomial M. Finally, after all literals of $\hat{M}$ have been selected, all clauses have become empty and been removed. This is exactly the point where the algorithm terminates with M being identical to $\hat{M}$. This completes the proof of Lemma 1. □

A k-CNF formula ψ may have many maximal monomials for a given satisfying assignment x. To find a subset of size c one could apply `SingleMaxMonomial` iteratively performing a depth-first search over its choices of literals in line 9. However, this strategy runs into the problem that different choices may lead to the same maximal monomial. Thus, the search time until enough distinct monomials have been found could be extremely large. Algorithm 3 solves this problem by pruning redundant branches at an early stage. It follows only those branches that actually shrink the set of remaining maximal monomials.

Proposition 1. *For a given k-CNF formula ψ, a satisfying assignment $x \in \texttt{sat}(\psi)$, and a positive integer c,* `MaxMonomials`*(ψ, x, c) returns a (ψ, x, c)-set. Its run time is bounded by $\mathcal{O}\left(p_k(n) \cdot n^c\right)$.*

Thus, to find the monomials in step 7 of `Learn-`k`-Term-DNF` we call `MaxMonomials` with parameters $(\psi, e_j, 2^k - 1)$.

In the rest of this section we will prove Proposition 1. Let ψ, x and c be as specified above. Lines 6–12 of `MaxMonomials` correspond to the deterministic part of `SingleMaxMonomial`. Lines 13–23 implement a pruned depth-first search over the nondeterministic decisions used to remove a literal. In lines 13–17 the algorithm computes a subset $\mathcal{L}$ of removable literals that are potentially useful for further examination. The global variable $\mathcal{M}_{\text{global}}$ is used to cancel the search as soon as enough distinct monomials have been found. The branches of the search are examined by recursive calls in line 20. The last parameter of the function `DFS`, the sequence *removed_literals*, is used to facilitate the analysis of the algorithm and plays no other role. An implementation of `MaxMonomials` can omit this parameter.

At the leaf level when the recursion stops only a single monomial is returned. Thus, eventually the algorithm has either visited the complete recursion tree or

Algorithm 3: `MaxMonomials`(ψ, x, c)

Input: k-CNF formula ψ; satisfying assignment $x \in \mathtt{sat}(\psi)$;
requested number of monomials $c > 0$

Output: a (ψ, x, c)-set $\mathcal{M}$

remove every literal from ψ that is not satisfied by x;
$\mathcal{M}_{\text{global}} \longleftarrow \emptyset$;
return `DFS`$(\psi, 1, \langle\,\rangle)$;

Function `DFS`$(\psi, M, \mathit{removed\_literals})$

`// termination test`
while there exists a clause in ψ consisting of a single literal ℓ **do**
  $M \longleftarrow (M \wedge \ell)$;
  remove all clauses from ψ that contain ℓ
end
if ψ is empty **then**
  $\mathcal{M}_{\text{global}} \longleftarrow \mathcal{M}_{\text{global}} \cup \{M\}$;
  return $\{M\}$
end

`// recursion preprocessing`
$\mathcal{M} \longleftarrow \emptyset$;
select a shortest clause K of ψ and an arbitrary literal ℓ of K;
let ψ' be obtained from ψ by removing ℓ from each clause of ψ ;
call `SingleMaxMonomial`(ψ', x) to get a (ψ', x)-maximal monomial M' ;
$\mathcal{L} \longleftarrow$ all literals in $M' \wedge \ell$;

`// recursion`
foreach $\ell \in \mathcal{L}$ **do**
  let ψ' be obtained from ψ by removing ℓ from each clause of ψ;
  $\mathcal{M} \longleftarrow \mathcal{M} \cup$ `DFS`$(\psi', M, \langle \mathit{removed\_literals}, \ell\rangle)$;
  if $|\mathcal{M}_{\mathit{global}}| \geq c$ **then return** $\mathcal{M}$;
end
return $\mathcal{M}$;

the cardinality of $\mathcal{M}_{\text{global}}$ achieves the value c and the procedure stops. Hence, `MaxMonomials` will never output more than c monomials. An example of the recursion tree is depicted in Fig. 2. It remains to show that our pruning strategy does not exclude maximal monomials.

Assume `MaxMonomials` is called with c larger than the total number of maximal monomials. Consider a fixed, but arbitrary run ρ on this instance. In this case the run will inspect the complete recursion tree. The sequence r of *removed_literals* that is the third argument of a recursive call of `DFS`$(\psi, M, \langle r\rangle)$ will be used to uniquely identify the other arguments. Thus, $\psi_r := \psi$ and $M_r := M$, and the whole triple

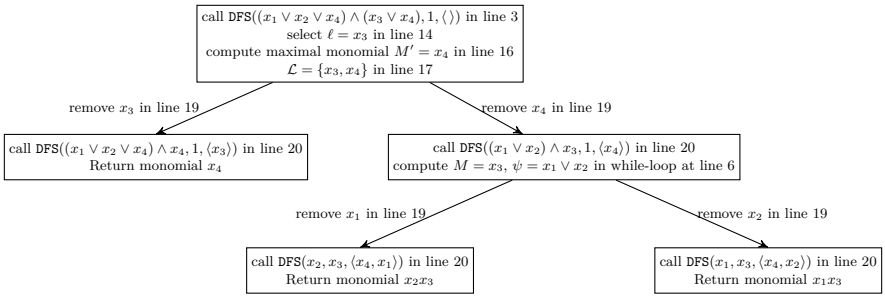

Fig. 2: A recursion tree of `MaxMonomials`(ψ, x, c) on input $\psi = (x_1 \vee x_2 \vee x_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4)$, $x = 1111$, and $c = 7$. This tree corresponds to the choice of the literal $\ell = x_3$ in line 14 and computing the monomial $M' = x_4$ in line 16 in the initial call of `DFS in line 3`. In this run the algorithm finds all (ψ, x)-maximal monomials: x_4, x_2x_3, and x_1x_3. Note that the last parameter of each call of `DFS`, sequence *removed_literals*, includes all edge literals which lay on the path from the root to the corresponding call.

$\psi_r, M_r, \langle r\rangle$ is denoted by $P\langle r\rangle$. In the initial call `DFS`$(\psi, 1, \langle\ \rangle)$ the sequence r of *removed_literals* is empty and this call is abbreviated by `DFS`$(P\langle\ \rangle)$. For the set of literals $\mathcal{L} = \{\ell_1, \ldots, \ell_t\}$ computed in line 17, `DFS`$(P\langle\ \rangle)$ initiates recursive calls of `DFS` with parameters $P\langle\ell_1\rangle, \ldots, P\langle\ell_t\rangle$. Note that the formula ψ' in line 19 is obtained from ψ by removing ℓ_i, and the second parameter M is the result of the deterministic part of Algorithm 2 on input (ψ, x).

Furthermore, if r defines a sequence of literals that corresponds to a call `DFS`$(P\langle r\rangle)$ and if $\mathcal{L} = \{\ell'_1, \ldots, \ell'_{t'}\}$ are computed during `DFS`$(P\langle r\rangle)$ in line 17, then `DFS`$(P\langle r\rangle)$ initiates the recursive calls `DFS`$(P\langle r, \ell'_1\rangle), \ldots,$ `DFS`$(P\langle r, \ell'_{t'}\rangle)$ parameterized by the sequences $\langle r, \ell'_i\rangle$. Accordingly, every $P\langle r, \ell'_i\rangle$ is determined by the choice of ℓ'_i in line 18. We define the notions child, ancestor and leaf of a call `DFS`$(P\langle r\rangle)$ in an obvious way.

Let `DFS`$(P\langle r\rangle)$ denote the set of monomials that is returned by the function call. Particularly, `DFS`$(P\langle\ \rangle)$ is the set of monomials computed by `MaxMonomials`. For a sequence r of literals let `AllMon`(r) denote the set of all (ψ, x)-maximal monomials that do not contain any literal of r.

Lemma 2. *For all parameters $P\langle r\rangle$ of the run ρ holds* `DFS`$(P\langle r\rangle) =$ `AllMon`(r).

Proof. If `MaxMonomials` examined the entire tree without pruning the property would hold due to Lemma 1. Thus, we have to show that it is sufficient to follow the literals in $\mathcal{L}$, which are the literals of $M' \wedge \ell$. The monomial $M' \wedge M$ does not contain ℓ, so the path to M' is not cut off at this stage. Since M' is a maximal monomial, for every other maximal monomial $\tilde{M} \in$ `AllMon`(r) there must be a literal of M' that is not contained in $\tilde{M}$. Thus, for every maximal monomial, there exists a path of removing-decisions that is not cut off at the current stage. The arguments above can be applied to every level of the recursion tree. Thus, every element of `AllMon`(r) is treated by `DFS`$(P\langle r\rangle)$. □

Hence, $\texttt{MaxMonomials}(\psi, x, c)$ with c large enough finds every (ψ, x)-maximal monomial.

Lemma 3. *For all parameters $P\langle r\ell\rangle$ of ρ it holds $\texttt{DFS}(P\langle r\ell\rangle) \subsetneq \texttt{DFS}(P\langle r\rangle)$.*

Proof. A child $P\langle r\ell\rangle$ of $P\langle r\rangle = (\psi', M, \langle r\rangle)$ fulfills $\ell \in \mathcal{L}_{\langle r\rangle}$, where $\mathcal{L}_{\langle r\rangle}$ denotes the value assigned to $\mathcal{L}$ in line 17 during $\texttt{DFS}(P\langle r\rangle)$. There are two types of literals in $\mathcal{L}_{\langle r\rangle}$. The first one are literals of the maximal monomial $M'_{\langle r\rangle}$ computed in line 16. Removing such a literal ℓ implies $\texttt{DFS}(P\langle r\ell\rangle) \subsetneq \texttt{DFS}(P\langle r\rangle)$, because $M'_{\langle r\rangle} \wedge M$ is in $\texttt{AllMon}\,(r)$, but not in $\texttt{AllMon}\,(r\ell)$.

The other type is a literal ℓ selected from a shortest clause $K_{\langle r\rangle}$ in line 14. Note that ψ' in line 15 does not contain any literal of r. We construct a maximal monomial containing ℓ, but none of the literals in r. Let us run $\texttt{SingleMaxMonomial}(\psi', x)$ with the following rule.

> Whenever an arbitrary literal has to be removed and $K_{\langle r\rangle}$ is still in ψ', remove a literal of $K_{\langle r\rangle}$ from ψ' that is not ℓ.

Because $K_{\langle r\rangle}$ is a shortest clause, no other clause will contain exactly one literal before $K_{\langle r\rangle}$ does. Thus, $K_{\langle r\rangle}$ will not be removed from ψ' until it contains exactly one literal. The last literal of $K_{\langle r\rangle}$ will be ℓ. Hence ℓ must be added to the maximal monomial. When we combine the result with M, we get a maximal monomial containing ℓ, but no literal from r, implying $\texttt{AllMon}\,(r\ell) \subsetneq \texttt{AllMon}\,(r)$. □

Note that the depth of the recursion tree is bounded by $\min\{n, |\,\texttt{AllMon}\,(\langle\,\rangle)\,|\}$, the minimum of n and cardinality of all (ψ, x)-maximal monomials. Otherwise there must be a node that violates Lemma 3.

Lemma 4. *For all parameters $P\langle r\rangle$ of ρ such that $P\langle r\rangle$ is the root of a subtree of depth h it holds $|\texttt{DFS}(P\langle r\rangle)| \geq h$.*

Proof. Otherwise there must be a node that violates Lemma 3. □

Lemma 5. *If the whole recursion tree has depth at least h then after n^h recursive calls of* DFS *at least one root of a subtree with depth exactly h has been visited.*

Proof. We construct a run where the root of such a subtree is visited as late as possible.

The number of branches per node is bounded by n. At most $n-1$ complete subtrees of height $h-2$ are added as children to the root and one larger subtree preventing the root and every node in the $n-1$ complete subtrees from being a root of a subtree of height h. The last child of the root can have at most $n-2$ complete subtrees of height $h-2$ as children and one larger subtree, and so on.

The maximal height of the whole tree is bounded by n. So, we can repeat this procedure at most $n-h$ times before the root of a subtree of height h must occur. Every subtree of height $h-2$ has less than n^{h-2} nodes. Hence, the total number of nodes that have been visited before the final subtree is reached must be less than n^h. □

These properties imply

Lemma 6. *For every positive integer c,* `MaxMonomials`(ψ, x, c) *returns a (ψ, x, c)-set after at most $\mathcal{O}(n^c)$ recursive calls.*

5 Bounding the Subset Size for Maximal Monomials

In this section we show that with high probability for every monomial M of the original k-term DNF formula, `Learn-`k`-Term-DNF` draws at least one sample x in line 3 such that every $(\psi, x, 2^k - 1)$-set covers almost all satisfying assignments of M. Figure 3 illustrates the idea for the case of the 3-term DNF target formula φ and the 3-CNF hypothesis ψ presented in Fig. 1. For the sample x on the left that satisfies the monomial of φ marked in red, there are seven (ψ, x)-maximal monomials – two of them are drawn in green and blue – that cover only a small portion of satisfying assignments of the red monomial. However, x on the right has only two maximal monomials – one being identical to the red monomial itself.

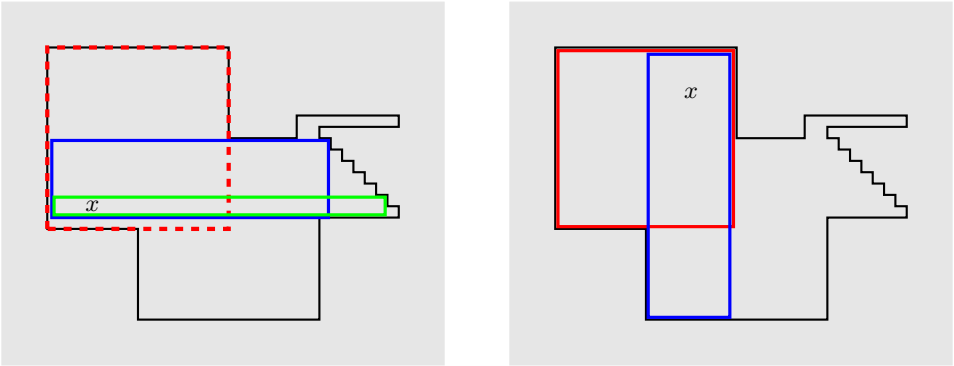

Fig. 3: The 3-term DNF formula φ and the 3-CNF hypothesis ψ of Fig. 1 with positive samples x and corresponding maximal monomials.

In the following let $\varphi = M_1 \vee \cdots \vee M_k$ be a k-term DNF formula, where we may assume that the monomials are ordered by increasing length, and $\varepsilon_1 = \varepsilon\, q^{-1}\, k^{-1}\, 2^{-(3k+1)}$ as defined in Algorithm 1.

Definition 4. *Let g denote the function $g(\varphi, q, k) := q\, 2^k |\,\mathtt{sat}(\varphi)|$.*
For $\gamma > 0$ we call a monomial M_i of φ γ-large *if $|\,\mathtt{sat}(M_i)| \geq \gamma\, g(\varphi, q, k)$.*

Proposition 2. *With probability at least $1 - \delta/2$, for every $(\varepsilon_1 2^{2k})$-large monomial M_i of φ holds:* `Learn-`k`-Term-DNF` *in line 3 draws at least one sample $e \in \mathtt{sat}(M_i)$ such that*
a) the number of (ψ, e)-maximal monomials is at most $2^k - 1$, and
b) there exists a (ψ, e)-maximal monomial M_i' with $\mathtt{sat}(M_i) \subseteq \mathtt{sat}(M_i')$.

To prove this claim let us first consider the case that the k-CNF formula ψ is equivalent to the unknown k-term DNF formula φ. In general as illustrated above,

$\mathtt{sat}(\psi)$ may cover only a part of the satisfying region of a monomial in a scattered way and there could exist many (ψ, x)-maximal monomials.

Definition 5. *For a non-empty subset $I = \{i_1, \ldots, i_p\} \subseteq \{1, \ldots, k\}$ of indices of the monomials of φ, a (φ, x)-maximal monomial M is called* (φ, x, I)-maximal *if $\mathtt{sat}(M) \subseteq \mathtt{sat}(M_{i_1} \vee \cdots \vee M_{i_p})$ and after removing any M_{i_j} from the right side this inclusion fails.*

Lemma 7. *For every φ, I and x, a (φ, x, I)-maximal monomial $M_{I,x}$ is unique. If $y \in \mathtt{sat}(M_{i_1} \vee \cdots \vee M_{i_p})$ also possesses a maximal monomial $M_{I,y}$ then $M_{I,y} = M_{I,x}$.*

Proof. Suppose $M_{I,y} \neq M_{I,x}$. This implies the existence of at least one literal ℓ such that ℓ occurs in exactly one of the two monomials. Without loss of generality, let ℓ occur in $M_{I,x}$.

$M_{I,x}$ is a (φ, x)-maximal monomial, so there exists at least one $i \in I$ such that the monomial M_i in φ contains ℓ. Further, there must exist some $j \in I$ such that M_j in φ does not contain ℓ, because otherwise $M_{I,y}$ would have to contain ℓ. So, there is a non-empty strict subset $J \subset I$ such that, for every $j \in I$, M_j does not contain ℓ, if and only if $j \in J$.

Let $s \in \mathtt{sat}(M_{I,y}) \setminus \mathtt{sat}(\ell)$ be an assignment. If $\overline{\ell}$ occurred in any M_i with $i \in I$, $M_{I,x}$ would not be (φ, x, I)-maximal, because i could be removed from I. From $\mathtt{sat}(M_{I,y}) \subseteq \mathtt{sat}(M_{i_1}, \ldots, M_{i_p})$ it follows that there must exist some $j \in J$ such that $s \in \mathtt{sat}(M_j)$. If we flip the bit belonging to ℓ in s, we get $s_{\overline{\ell}}$. Notice that $s_{\overline{\ell}} \in \mathtt{sat}(M_j)$, because M_j contains neither ℓ nor $\overline{\ell}$.

Such a monomial M_j, with $j \in J$, that is satisfied by s and $s_{\overline{\ell}}$ can be found for any assignment $s \in \mathtt{sat}(M_{I,y}) \setminus \mathtt{sat}(\ell)$. Thus, any assignment that satisfies $M_{I,y}$ also satisfies some M_j with $j \in J$. So one has $\mathtt{sat}(M_{I,y}) \subseteq \mathtt{sat}(M_{j_1} \vee M_{j_2} \vee \cdots \vee M_{j_{|J|}})$, with $j_1, j_2, \ldots, M_{j_{|J|}} \in J \subset I$, and $M_{I,y}$ would not be (φ, y, I)-maximal. This is a contradiction. It follows that $M_{I,y} = M_{I,x}$. □

This implies that the number of different (φ, x, I)-maximal monomials over all $x \in \mathtt{sat}(\varphi)$ and nonempty $I \subseteq \{1, \ldots, k\}$ is bounded by $2^k - 1$. Next we will derive a bound on the number of satisfying assignments for those maximal monomials that intersect potentially scattered regions of φ.

Lemma 8. *For $d \in \mathbb{N}$ let $u(d)$ be the number of monomials M_i with $|\mathtt{sat}(M_i)| \geq 2^d$ and $\varphi_d = M_1 \vee \cdots \vee M_{u(d)}$ the conjunction of these large monomials (remember the monomials are ordered by length). For the remaining small monomials $M_{u(d)+1}, \ldots, M_k$ and a Boolean formula χ_d with $\mathtt{sat}(\chi_d) \subseteq \mathtt{sat}(M_{u(d)+1} \vee \cdots \vee M_k)$ define the following:*

1. $\psi_d := \varphi_d \vee \chi_d$,
2. $\mathcal{M}^{[d]} := \{M \mid M$ *is a* (ψ_d, x)*-max. monom. for some* $x \in \mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d)\}$,
3. $\xi_d := \bigvee_{M \in \mathcal{M}^{[d]}} M$.

Then, $|\mathtt{sat}(\xi_d)| \leq 2^{d+k-1}$.

Proof. If $\mathtt{sat}(\chi_d) \subseteq \mathtt{sat}(\varphi_d)$ then $|\mathtt{sat}(\xi_d)| = 0$ since $\mathcal{M}^{[d]} = \emptyset$. Otherwise, consider an assignment $x \in \mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d)$. The formula $\varphi_d(x)$ with $\mathtt{sat}(\varphi_d(x)) \supseteq \mathtt{sat}(\varphi_d)$

is derived from φ_d by removing every literal that supports x. Thereby, x does not become a satisfying assignment of $\varphi_d(x)$. Further, let M^x denote the monomial with $\mathtt{sat}(M^x) = \{x\}$ and define $\psi_d(x) := \varphi_d(x) \vee M^x$. To continue the proof we make the following claims.

Fact 3. *For any assignment $z \in \mathtt{sat}(\xi_d)$, there exists an assignment $x \in \mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d)$, such that every $(\psi_d(x), x)$-maximal monomial is satisfied by z.*

Let ξ'_d be a disjunction that for every assignment $x \in \mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d)$ contains an arbitrary $(\psi_d(x), x)$-maximal monomial $M_d(x)$. From Fact 3 we get the inequality $|\,\mathtt{sat}(\xi_d)| \leq |\,\mathtt{sat}(\xi'_d)|$.

Fact 4. *Every $(\psi_d(x), x)$-maximal monomial $M_d(x)$ in ξ'_d has at most $2^{u(d)}$ satisfying assignments.*

From these facts we can conclude the upper bound stated in Lemma 8 as follows:

$$\begin{aligned} |\,\mathtt{sat}(\xi_d)| \leq |\,\mathtt{sat}(\xi'_d)| \;&\leq \sum_{x \in \mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d)} |\,\mathtt{sat}(M_d(x))| \;\leq\; |\,\mathtt{sat}(\chi_d)|\, 2^{u(d)} \\ &\leq 2^d\,(k - u(d))\, 2^{u(d)} \;\leq\; 2^{d+k-1} \qquad \text{since } u(d) \leq k\ . \end{aligned}$$

It remains to prove Fact 3 and Fact 4.

Proof (Fact 3). Consider an arbitrary assignment $z \in \mathtt{sat}(\xi_d)$.

- **Case 1:** $z \notin \mathtt{sat}(\varphi_d)$.
  Because of $z \in \mathtt{sat}(\chi_d)$, with $x = z$ every $(\psi_d(x), x)$-maximal monomial is satisfied by z.
- **Case 2:** $z \in \mathtt{sat}(\varphi_d)$.
  ξ_d contains a monomial M with $z \in \mathtt{sat}(M)$. Furthermore, there must be at least one assignment in $(\mathtt{sat}(M) \cap \mathtt{sat}(\chi_d)) \setminus \mathtt{sat}(\varphi_d)$ due to the construction of ξ_d. From these assignments, we choose an assignment x with minimal Hamming distance to z.
  Let y be an arbitrary assignment that differs from x at exactly one position towards z. Note that $y \in \mathtt{sat}(M) \cap \mathtt{sat}(\varphi_d)$. Let M_j be a monomial in φ_d that is satisfied by y. During the construction of $\varphi_d(x)$, all literals that are satisfied by x are removed from M_j. Thus, the equivalent for M_j in $\varphi_d(x)$, $M_j(x)$, consists only of one literal $\ell(x, y)$ with $x \notin \mathtt{sat}(\ell(x, y))$ and $y \in \mathtt{sat}(\ell(x, y))$. Hence, no $(\psi_d(x), x)$-maximal monomial contains $\overline{\ell(x, y)}$.
  This argument can be repeated for any choice of y and any $\overline{\ell(x, y)}$ that satisfies x, but not z. Since every $(\psi_d(x), x)$-maximal monomial is satisfied by x, it is also satisfied by z. □

Proof (Fact 4). Let $M_d(x)$ be a $(\psi_d(x), x)$-maximal monomial of ξ'_d. It holds $\{x\} = \mathtt{sat}(M^x) \subseteq \mathtt{sat}(M_d(x))$. Thus, M^x only differs from $M_d(x)$ in some additional literals, $\ell_1, \ldots, \ell_r$. This implies $|\,\mathtt{sat}(M_d(x))| = 2^r\ |\,\mathtt{sat}(M^x)| = 2^r$.

Let $M^{x,\bar{\ell}}$ denote the monomial that is formed by replacing ℓ by $\bar{\ell}$ in M^x. We consider a literal ℓ_s, with $s \in \{1, \ldots, r\}$. It holds $\mathtt{sat}(M^{x,\overline{\ell_s}}) \not\subseteq \mathtt{sat}(M^x)$. From

$\mathtt{sat}(M^{x,\overline{\ell_s}}) \subseteq \mathtt{sat}(M_d(x)) \subseteq \mathtt{sat}(\varphi_d(x) \vee M^x)$, it follows that there exists at least one assignment satisfying $M^{x,\overline{\ell_s}}$ and $\varphi_d(x)$. Hence, there must be at least one monomial M_t in $\varphi_d(x)$, with $t \in \{1, \ldots, u(d)\}$, that is satisfied by the same assignment. Thus, M_t cannot contain any negated literal from $M^{x,\overline{\ell_s}}$.

Now assume that $r > u(d)$. Then there must be at least one monomial M_t in $\varphi_d(x)$, with $t \in \{1, \ldots, u(d)\}$, that shares a satisfying assignment with a monomial $M^{x,\overline{\ell_s}}$ as well as with $M^{x,\overline{\ell_{s'}}}$, where $s' \in \{1, \ldots, s-1, s+1, \ldots, r\}$. Thus, M_t cannot contain any negated literal from $M^{x,\overline{\ell_s}}$ or $M^{x,\overline{\ell_{s'}}}$, especially neither $\ell_{s'}$ nor ℓ_s. It follows that $\mathtt{sat}(M^x) \subseteq \mathtt{sat}(M_t)$. By $\mathtt{sat}(M_t) \subseteq \mathtt{sat}(\varphi_d(x))$, we get $\mathtt{sat}(M^x) \subseteq \mathtt{sat}(\varphi_d(x))$. The construction of ξ'_d implies $x \in \mathtt{sat}(M^x) \setminus \mathtt{sat}(\varphi_d(x))$. This is a contradiction. So, $r \leq u(d)$ and thereby $|\,\mathtt{sat}(M_d(x))| \leq 2^{u(d)}\ |\,\mathtt{sat}(M^x)| = 2^{u(d)}$.

□

This completes the proof of Lemma 8. □

Now let us estimate how well a k-CNF formula ψ can reconstruct the original monomials of the unknown k-term DNF φ to bound the size of the scattered regions χ_d.

Lemma 9. *Let $\psi = K_1 \wedge \cdots \wedge K_p$ be a k-CNF formula with clauses K_j and $\mathtt{sat}(\psi) \subseteq \mathtt{sat}(\varphi)$, D be a q-bounded distribution and $\gamma > 0$.*
If $D(\mathtt{sat}(\varphi) \setminus \mathtt{sat}(\psi))/D(\mathtt{sat}(\varphi)) < \gamma$ then $\mathtt{sat}(M_i) \subseteq \mathtt{sat}(\psi)$ for every γ-large M_i.

Proof. Choose a monomial M_i with $|\,\mathtt{sat}(M_i)| \geq \gamma\ 2^k\ q\ |\,\mathtt{sat}(\varphi)|$ and consider $x \in (\mathtt{sat}(M_i) \setminus \mathtt{sat}(\psi))$. There must be a clause K_j in ψ, such that $x \notin \mathtt{sat}(K_j)$. This clause K_j cannot contain any literal from M_i. So, K_j is not satisfied by at least $2^{-k}|\,\mathtt{sat}(M_i)|$ assignments that satisfy M_i. Even in the case that all these assignments are q-times more likely than any other assignment satisfying φ according to D, one has $D(\mathtt{sat}(\varphi \wedge \neg\psi))\ /\ D(\mathtt{sat}(\varphi)) \geq D(\mathtt{sat}(M_i \wedge \neg\psi))\ /\ D(\mathtt{sat}(\varphi)) \geq 2^{-k}\ |\,\mathtt{sat}(M_i)|\ /\ (q\ |\,\mathtt{sat}(\varphi)|) \geq \gamma$. Consequently, the assignment x cannot exist and the claim follows. □

Thus, if a CNF formula ψ approximates a k-term DNF formula φ quite well without false positives then every monomial of φ with large support is completely covered by ψ. Only monomials with small support may give rise to errors in the approximation.

Lemma 10. *Let φ_i equal φ without M_i. Then $|\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i)| \geq |\,\mathtt{sat}(M_i)| \cdot 2^{-k+1}$.*

Proof. Consider a monomial $M_j \neq M_i$ from φ. There must be at least one literal ℓ_j from M_j that is not contained in M_i. So, $\mathtt{sat}(M_i \wedge \overline{\ell_j}) \cap \mathtt{sat}(M_j) = \emptyset$ and $|\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(M_j)| \geq |\,\mathtt{sat}(M_i \wedge \overline{\ell_j})| \geq |\,\mathtt{sat}(M_i)| \cdot 2^{-1}$. Successively, we obtain $|\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i)| \geq |\,\mathtt{sat}(M_i)| \cdot 2^{-k+1}$. □

Now we are ready to prove the main result of this section.

Proof (Proposition 2). Let M_i be an $(\varepsilon_1 2^{2k})$-large monomial and φ_i as defined above. The algorithm starts by learning a k-CNF formula ψ. As already mentioned in Sect. 3

this can be done without false positives and with relative error bounds $(\varepsilon_1, \delta/2)$ using N_1 positive samples. Assume that the algorithm has learned such a ψ with $D(\mathtt{sat}(\varphi) \setminus \mathtt{sat}(\psi)) \;/\; D(\mathtt{sat}(\varphi)) < \varepsilon_1$, which happens with probability at least $1 - \delta/2$.

Consider the construction in Lemma 8 for $d = \lceil \log(\varepsilon_1 \; g(\varphi, q, k)) \rceil$ and χ_d as defined above. Then $\psi_d = \varphi_d \vee \chi_d = \varphi$ and ξ_d is the disjunction of all (φ, x)-maximal monomials for elements $x \in \mathtt{sat}(\varphi)$ that are not covered by the monomials of φ with support at least 2^d. Using Lemma 8 and Lemma 10 we can conclude

$$\begin{aligned} |\,\mathtt{sat}(\xi_d)| &\leq 2^{d+k-1} \;\leq\; \varepsilon_1 \; g(\varphi, q, k) \; 2^k \\ &\leq 2^{-k} \; |\,\mathtt{sat}(M_i)| \;<\; |\,\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i)| \; . \end{aligned}$$

Thus, we obtain

Claim 1: $\mathtt{sat}(M_i) \setminus (\mathtt{sat}(\varphi_i) \cup \mathtt{sat}(\xi_d)) \neq \emptyset$, that means every $(\varepsilon_1 2^{2k})$-large monomial has a satisfying assignment that is not covered by any other monomial of φ nor by any (φ, x)-maximal monomial for elements in $\mathtt{sat}(\varphi)$ that are not covered by monomials of support at least 2^d.

Next we show

Claim 2: The sample sequence E given to `Learn-`k`-Term-DNF` at the beginning contains an element $e \in \mathtt{sat}(M_i) \setminus (\mathtt{sat}(\varphi_i) \cup \mathtt{sat}(\xi_d))$.

If none of the e_j of E belongs to $\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i)$ the DNF formula φ_i would be satisfied by every sample in E. The formula ψ has to be learned without false positives. Since the learner cannot distinguish between the concepts φ and φ_i he must also learn φ_i without false positives, thus $(\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i)) \cap \mathtt{sat}(\psi) = \emptyset$. Thus, we get $\mathtt{sat}(M_i) \not\subseteq \mathtt{sat}(\psi)$ which contradicts Lemma 9. Hence, there exists some $e \in E$ that satisfies M_i, but no other M of φ.
If e does not satisfy ξ_d we are done.
Otherwise, fix an assignment $x \in \mathtt{sat}(M_i) \setminus (\mathtt{sat}(\varphi_i) \cup \mathtt{sat}(\xi_d))$ and assume that every $e_j \in E \cap (\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i))$ satisfies ξ_d, thus there exists a monomial $M(e_j)$ in ξ_d that is satisfied by e_j. For each such monomial $M(e_j)$ there must exist a literal ℓ_j in $M(e_j)$ that is set to true in e_j, but to false in x. Further, for every such ℓ_j the formula φ must contain at least one monomial $M(\ell_j)$ that includes ℓ_j since otherwise no (φ, x)-maximal monomial could contain ℓ_j. If $M(\ell_j)$ is not satisfied by e_j we may throw away some literals of this monomial to achieve this property, but keep ℓ_j.

We repeat this process for each sample of E belonging to $\mathtt{sat}(M_i) \setminus \mathtt{sat}(\varphi_i)$. Finally, by removing M_i from φ and taking all other monomials M potentially shortened we get a k-term DNF φ' that is compatible to E. Thus a learner cannot distinguish between φ and φ'. But φ' is not satisfied by x, therefore x cannot be in $\mathtt{sat}(\psi)$, which again contradicts Lemma 9. Thus, there must be a sample $e \in E$ that satisfies M_i, but not ξ_d.

Claim 3: The number of (ψ, e)-maximal monomials is bounded by $2^k - 1$.

Consider the monomials $M(e)$ that are (ψ, e)-maximal. Since $e \notin \mathtt{sat}(\xi_d)$, the con-

struction of ξ_d yields $\mathtt{sat}(M(e)) \cap (\mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d)) = \emptyset$, because otherwise $x \in \mathtt{sat}(M(e)) \cap (\mathtt{sat}(\chi_d) \setminus \mathtt{sat}(\varphi_d))$ would cause a (ψ, x)-maximal monomial $M(x) = M(e)$ in ξ_d and therefore $e \in \mathtt{sat}(\xi_d)$. Therefore, $\mathtt{sat}(M(e)) \subseteq \mathtt{sat}(\varphi_d)$ and $M(e)$ is (φ_d, e)-maximal, because by Lemma 9, $\mathtt{sat}(\varphi_d) \subseteq \mathtt{sat}(\psi)$. The number of (φ_d, e)-maximal monomials is bounded by $2^k - 1$ due to Lemma 7.

Finally, the fact $\mathtt{sat}(M_i) \subseteq \mathtt{sat}(\psi)$ implies that at least one of the (ψ, e)-maximal monomials M_i' covers M_i completely. □

6 The Complexity and Correctness of `Learn-`k`-Term-DNF`

First we show that it is sufficient to consider only the $(2^k - 1)$ shortest monomials as done in line 10 of the algorithm.

Lemma 11. *With probability at least $1 - \delta/2$, for every $(\varepsilon_1 2^{2k})$-large monomial M_i at the beginning of the second phase of* `Learn-`k`-Term-DNF` *a monomial M_i' with $\mathtt{sat}(M_i') \supseteq \mathtt{sat}(M_i)$ belongs to $\mathcal{M}$.*

Proof. By Proposition 2, such a M_i' is added to $\mathcal{M}$ in line 7. It remains to show that M_i' is not removed later, which could only happen in line 10.

To prove that M_i' is among the $2^k - 1$ shortest monomials in $\mathcal{M}$ note that $|\mathtt{sat}(M_i)| > |\mathtt{sat}(\xi_d)|$ implies that $|\mathtt{sat}(M_i)|$ is bigger than the number of satisfying assignments of any monomial in ξ_d. Thus, only (φ_d, e)-maximal monomials can be shorter than M_i'. By Lemma 7, the number of these monomials is bounded by $2^k - 1$. Hence, M_i' is one of the $2^k - 1$ shortest monomials that will be kept in $\mathcal{M}$. □

Now we are ready to conclude that `Learn-`k`-Term-DNF` satisfies the properties stated in Theorem 1.

Proof (Theorem 1). The output hypothesis of the learner is a disjunction of maximal monomials. The support of every maximal monomial is a subset of the support of the k-CNF formula ψ. Since $\mathtt{sat}(\psi) \subseteq \mathtt{sat}(\varphi)$ it follows that the output hypothesis has no false positives.

Suppose that for every every $(\varepsilon_1 2^{2k})$-large monomial M_i of φ the learner has added a monomial M_i' with $\mathtt{sat}(M_i') \supseteq \mathtt{sat}(M_i)$ to $\mathcal{M}$. In the final stage as possible hypotheses the set $\mathcal{H}_{\mathcal{M}}$ of all up to k-fold disjunctions of monomials of $\mathcal{M}$ are used. Its size can be bounded by

$$|\mathcal{H}_{\mathcal{M}}| \;\leq\; \sum_{i=0}^{k} \binom{2^k - 1}{i} < 2^{k^2} \;\; .$$

Assume that for every $(\varepsilon_1 2^{2k})$-large monomial M_i of φ a covering monomial M_i' is part of the output and let H be a hypothesis that includes all these M_i'. Then less than k monomials from φ with less than $\varepsilon_1\, 2^{2k}\, g(\varphi, p, k)$ satisfying assignments each are not covered. Hence, the error of H is bounded by

$$k\, \varepsilon_1\, 2^{2k}\, g(\varphi, p, k) \;=\; \frac{\varepsilon}{2}\, |\mathtt{sat}(\varphi)| \;\; .$$

Let us estimate the probability that such a good H will not be given as output.

Case 1: H is not satisfied by enough samples to be accepted as a hypothesis.
The probability that this happens can be bounded by

$$p_1 \;\leq\; \sum_{i=0}^{N_2\left(1-\frac{3\varepsilon}{4}\right)} \binom{N_2}{i} \left(1-\frac{\varepsilon}{2}\right)^i \left(\frac{\varepsilon}{2}\right)^{N_2-i} \quad .$$

Using Proposition 2.4 of Angluin and Valiant [2] (for $\beta = \varepsilon/(4-2\varepsilon)$), we can bound p_1 by the following inequation:

$$\begin{aligned} p_1 \;&\leq\; \exp\left(-\frac{\varepsilon^2 N_2\left(1-\frac{\varepsilon}{2}\right)}{2(4-2\varepsilon)^2}\right) \;=\; \exp\left(-\frac{\varepsilon^2 \frac{48}{\varepsilon^2} \ln \frac{2^{k^2+2}}{\delta}\left(1-\frac{\varepsilon}{2}\right)}{32\left(1-\frac{\varepsilon}{2}\right)^2}\right) \\ &=\; \left(\frac{\delta}{2^{k^2+2}}\right)^{\frac{3}{2\left(1-\frac{\varepsilon}{2}\right)}} \quad . \end{aligned}$$

$0 < \varepsilon < 1$ implies $3/(2-\varepsilon) > 1$ and $0 < \delta < 1$ gives $p_1 < \delta/4$.

Case 2: The learner chooses a *bad* hypothesis $H' \in \mathcal{H}_{\mathcal{M}}$ that does not satisfy the error bound ε before reaching a good H. It may happen that the learner terminates before reaching H.
The probability that H' is satisfied by enough samples to be accepted as a hypothesis is

$$p_2 \;\leq\; \sum_{i=N_2\left(1-\frac{3\varepsilon}{4}\right)}^{N_2} \binom{N_2}{i} (1-\varepsilon)^i \, \varepsilon^{N_2-i} \quad .$$

Similarly, (for $\beta = \varepsilon/(4-4\varepsilon)$), one can bound p_2 by the following inequation:

$$\begin{aligned} p_2 \;&\leq\; \exp\left(-\frac{\varepsilon^2 N_2(1-\varepsilon)}{3(4-4\varepsilon)^2}\right) \;=\; \exp\left(-\frac{\varepsilon^2 \frac{48}{\varepsilon^2} \ln \frac{2^{k^2+2}}{\delta}(1-\varepsilon)}{48(1-\varepsilon)^2}\right) \\ &=\; \left(\frac{\delta}{2^{k^2+2}}\right)^{\frac{1}{1-\varepsilon}} \quad . \end{aligned}$$

$0 < \varepsilon < 1$ implies $1/(1-\varepsilon) > 1$ and $0 < \delta < 1$ gives $p_2 < \delta \; 2^{-(k^2+2)}$. Thus, if $H \in \mathcal{H}_{\mathcal{M}}$, then the probability that the learner outputs a hypothesis that violates the error bound ε is

$$p_{\text{error}} \;\leq\; p_1 + |\mathcal{H}_{\mathcal{M}}| \; p_2 < \frac{\delta}{4} + 2^{k^2} \; \delta \; 2^{-(k^2+2)} = \frac{\delta}{2} \quad .$$

According to Lemma 11, the probability for $H \in \mathcal{H}_{\mathcal{M}}$ is at least $(1-\frac{\delta}{2})$. So, with probability at least $(1-\delta)$, the hypothesis output by `Learn-`k`-Term-DNF` satisfies the error bound ε.

It is left to show that for constant k the algorithm runs in time polynomial in $1/\varepsilon$, $1/\delta$, n, and q. The number of samples $\sigma(\varepsilon, \delta, n, k, q)$ fulfills this bound. The k-CNF formula can be learned in polynomial time with respect to all parameters

except k. The algorithm computes at most 2^{k-1} maximal monomials for each sample. By Proposition 1 this can be done in polynomial time. Hence the time needed for computing candidate monomials is polynomial, too. The size of the hypothesis space for the sequential test is $|\mathcal{H}_{\mathcal{M}}| \leq 2^{k^2}$. The test itself can be performed in polynomial time. □

We note that our learning algorithm can be made applicable even if the parameter q is unknown (see [12]).

7 Infeasibility for Unrestricted DNF Formulas

Verbeurgt [30] has developed a method for learning poly(n)-term DNF over the uniform distribution from a polynomial number of positive and negative samples with a quasi-polynomial running time. In contrast, we can show:

Theorem 2. *For every hypothesis space $\mathcal{H}$ learning n-term DNF formulas without false positives requires an exponential number of positive samples. This even holds when fixing the set of possible distributions to a single arbitrary q-bounded distribution for error parameter $\varepsilon < 1/q$.*

Proof. For every nontrivial error parameters (ε, δ) and the uniform distribution the number of positive samples needed to learn (monotone) clauses without false positives is $\Omega(2^{n/4})$, independent of the hypothesis space [15]. The class n-term DNF contains all clauses, thus the number of samples needed to learn n-term DNF without false positives cannot be smaller. For q-bounded distributions, in the best case the error ε may decrease by a factor q at most. □

8 Conclusions

We have presented a polynomial-time algorithm for properly learning k-term DNF formulas from positive samples alone for every fixed k. On the contrary, if k may grow linearly with the number of variables DNF formulas cannot be learned efficiently from positive samples without false positives due to information theoretical reasons. For log-term DNF formulas our learner needs a quasi-polynomial number of samples. Can this bound be improved?

References

1. Aizenstein, H., Pitt, L.: On the learnability of disjunctive normal form formulas. Machine Learning 19(3), 183–208 (1995)
2. Angluin, D., Valiant, L.G.: Fast probabilistic algorithms for Hamiltonian circuits and matchings. J. Comput. Syst. Sci. 18(2), 155–193 (1979)
3. Angluin, D., Kharitonov, M.: When won't membership queries help? J. Comput. System Sci. 50(2), 336–355 (1995)
4. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the Vapnik–Chervonenkis dimension. J. ACM 36(4), 929–965 (1989)

5. Bshouty, N.H., Mossel, E., O'Donnell, R., Servedio, R.A.: Learning DNF from random walks. J. Comput. System Sci. 71(3), 250–265 (2005)
6. Bshouty, N.H., Tamon, C.: On the Fourier spectrum of monotone functions. J. ACM 43(4), 747–770 (1996)
7. De, A., Diakonikolas, I., Servedio, R.A.: Learning from satisfying assignments. In: Proc. 26th SODA. pp. 478–497. SIAM (2014)
8. Denis, F.: PAC learning from positive statistical queries. In: Proc. 9th ALT. pp. 112–126. Springer (1998)
9. Denis, F., Gilleron, R., Letouzey, F.: Learning from positive and unlabeled examples. Theoretical Computer Science 348(1), 70–83 (2005)
10. Ernst, M., Liśkiewicz, M., Reischuk, R.: Algorithmic learning for steganography: proper learning of k-term DNF formulas from positive samples. In: Proc. 26th ISAAC. pp. 151–162. Springer (2015)
11. Flammini, M.: On the learnability of monotone $k\mu$-DNF formulae under product distributions. Inform. Process. Lett. 52(3), 167–173 (1994)
12. Flammini, M., Marchetti-Spaccamela, A., Kučera, L.: Learning DNF formulae under classes of probability distributions. In: Proc. 5th COLT. pp. 85–92. ACM (1992)
13. Jackson, J.C.: An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. J. Comput. System Sci. 55(3), 414–440 (1997)
14. Jerrum, M.R., Valiant, L.G., Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. Theor. Comput. Sc. 43, 169–188 (1986)
15. Kearns, M., Li, M., Valiant, L.: Learning Boolean formulas. J. ACM 41(6), 1298–1328 (1994)
16. Klivans, A.R., Servedio, R.: Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. J. Comput. System Sci. 68(2), 303–318 (2004)
17. Kučera, L., Marchetti-Spaccamela, A., Protasi, M.: On learning monotone DNF formulae under uniform distributions. Inform. and Comput. 110(1), 84–95 (1994)
18. Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, Fourier transform, and learnability. J. ACM 40(3), 607–620 (1993)
19. Liśkiewicz, M., Reischuk, R., Wölfel, U.: Grey-box steganography. Theor. Comput. Sc. 505, 27–41 (2013)
20. Mansour, Y.: An $\mathcal{O}(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. J. Comput. System Sci. 50(3), 543–550 (1995)
21. Natarajan, B.K.: Probably approximate learning of sets and functions. SIAM J. Comput. 20(2), 328–351 (1991)
22. Natarajan, B.K.: On learning boolean functions. In: Proc. 19th STOC. pp. 296–304. ACM (1987)
23. Pitt, L., Valiant, L.G.: Computational limitations on learning from examples. J. ACM 35(4), 965–984 (1988)
24. Sakai, Y., Maruoka, A.: Learning monotone log-term DNF formulas under the uniform distribution. Theory of Comput. Systems 33(1), 17–33 (2000)
25. Sakai, Y., Maruoka, A.: Learning k-term monotone Boolean formulae. In: Proc. 3rd ALT, LNCS, vol. 743, pp. 195–207. Springer (1993)

26. Sakai, Y., Takimoto, E., Maruoka, A.: Proper learning algorithm for functions of k terms under smooth distributions. Inform. and Comput. 152(2), 188–204 (1999)
27. Servedio, R.A.: On learning monotone DNF under product distributions. Inform. and Comput. 193(1), 57–74 (2004)
28. Shvaytser, H.: A necessary condition for learning from positive examples. Machine Learning 5(1), 101–113 (1990)
29. Valiant, L.G.: A theory of the learnable. CACM 27(11), 1134–1142 (1984)
30. Verbeurgt, K.: Learning DNF under the uniform distribution in quasi-polynomial time. In: Proc. 3rd COLT. pp. 314–326. Morgan Kaufmann Publishers Inc., San Francisco (1990)

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il