JMLR: Workshop and Conference Proceedings vol 49:1–16, 2016

Complexity Theoretic Limitations on Learning DNF's

Amit Daniely AMITDANIELY@GOOGLE.COM
Google Inc, Mountain-View, California[*]

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

Abstract

Using the recently developed framework of Daniely et al. (2014), we show that under a natural assumption on the complexity of random K-SAT, learning DNF formulas is hard. Furthermore, the same assumption implies the hardness of various learning problems, including intersections of $\omega(\log(n))$ halfspaces, agnostically learning conjunctions, as well as virtually all (distribution free) learning problems that were previously shown hard (under various complexity assumptions).

Keywords: DNFs, Hardness of learning

1. Introduction

In the PAC learning model (Valiant, 1984), a learner is given an oracle access to randomly generated samples $(X, Y) \in \mathcal{X} \times \{0, 1\}$ where X is sampled from some *unknown* distribution $\mathcal{D}$ on $\mathcal{X}$ and $Y = h^*(X)$ for some *unknown* $h^* : \mathcal{X} \to \{0, 1\}$. It is assumed that h^* comes from a predefined *hypothesis class* $\mathcal{H}$, consisting of $0, 1$ valued functions on $\mathcal{X}$. The learning problem defined by $\mathcal{H}$ is to find $h : \mathcal{X} \to \{0, 1\}$ that minimizes $\mathrm{Err}_{\mathcal{D}}(h) := \mathrm{Pr}_{X \sim \mathcal{D}}(h(X) \neq h^*(X))$. For concreteness, we take $\mathcal{X} = \{\pm 1\}^n$, and say that the learning problem is tractable if there is an algorithm that on input ϵ, runs in time $\mathrm{poly}(n, 1/\epsilon)$ and outputs, w.h.p., a hypothesis h with $\mathrm{Err}(h) \leq \epsilon$.

Assuming $\mathbf{P} \neq \mathbf{NP}$, the status of most basic *computational* problems is fairly well understood. In a sharp contrast, 30 years after Valiant's paper, the status of most basic *learning* problems is still wide open – there is a huge gap between the performance of best known algorithms and hardness results (e.g., Daniely et al. (2014)). The main obstacle is the ability of a learning algorithm to return a hypothesis which does not belong to $\mathcal{H}$ (such an algorithm is called *improper*). This flexibility makes it very hard to apply reductions from $\mathbf{NP}$-hard problems (see Applebaum et al. (2008); Daniely et al. (2014)). Until recently, there was only a single framework, due to Kearns and Valiant (Kearns and Valiant, 1989), to prove lower bounds on learning problems. The framework of Kearns and Valiant (1989) makes it possible to show that certain cryptographic assumptions imply hardness of certain learning problems. As indicated above, the lower bounds established by this method are far from the performance of best known algorithms.

In a recent paper Daniely et al. (2014) (see also Daniely et al. (2013)) developed a new framework to prove hardness of learning based on hardness on average of CSP problems. Yet, Daniely et al. (2014) were not able to use their technique to establish hardness results that are based on a natural assumption on a well studied problem. Rather, they made a quite speculative hardness assumption, that is concerned with general CSP problems, most of which were never studied ex-

[*] Most work was done while the author was a Ph.D. student at the Hebrew University, Jerusalem, Israel

plicitly. This was recognized in Daniely et al. (2014) as the main weakness of their approach, and therefore the main direction for further research. About a year after, Allen et al. (2015) refuted the assumption of Daniely et al. (2014). On the other hand Daniely (2016) was able to overcome the use of the speculative assumption, and proved hardness of approximately and agnostically learning of halfspaces based on a natural assumption on the complexity of refuting random K-XOR instances, in the spirit of Feige's assumption (Feige, 2002). Likewise, in this paper, under a natural assumption on the complexity of random K-SAT we show:

1. Learning DNF's is hard.
2. Learning intersections of $\omega(\log(n))$ halfspaces is hard, even over the boolean cube.
3. Learning sparse polynomial threshold functions is hard, even over the boolean cube.
4. Agnostically[1] learning conjunctions is hard.
5. Agnostically learning halfspaces is hard, even over the boolean cube.
6. Agnostically learning parities is hard.
7. Learning finite automata is hard.

To the best of our knowledge, results 1, 2, 3, 4 are new, in the sense that there were no previous unrestricted hardness of learning results for these problems. We note that 5, 7 can be established under cryptographic assumptions, using the cryptographic technique (Feldman et al., 2006; Kearns and Valiant, 1989), and also assuming that random K-XOR is hard (Daniely, 2016). Also, 6 follows from the hardness of learning parities with noise[2] (Blum et al., 2003), which is often taken as a hardness assumption. As for 2, the previously best lower bounds (Klivans and Sherstov, 2006) only rule out learning intersections of polynomially many halfspaces, again under cryptographic assumptions. To the best of our knowledge, 1-7 implies the hardness of virtually all (distribution free) learning problems that were previously shown hard (under various complexity assumptions).

1.1. The random K-SAT assumption

Unless we face a dramatic breakthrough in complexity theory, it seems unlikely that hardness of learning can be established on standard complexity assumptions such as $\mathbf{P} \neq \mathbf{NP}$ (see Applebaum et al. (2008); Daniely et al. (2014)). Indeed, all currently known lower bounds are based on cryptographic assumptions. Similarly to Feige's paper Feige (2002), we rely here on the hardness of refuting random K-SAT formulas. As cryptographic assumptions, our assumption asserts the hardness on average of a certain problem that have resisted extensive attempts of attack during the last 50 years (e.g. Davis et al. (1962); Beame and Pitassi (1996); Beame et al. (1998); Ben-Sasson and Wigderson (1999); Feige (2002); Feige and Ofek (2004); Coja-Oghlan et al. (2004, 2010)).

Let $J = \{C_1, \ldots, C_m\}$ be a random K-SAT formula on n variables. Precisely, each K-SAT constraint C_i is chosen independently and uniformly from the collection of n-variate K-SAT constraints. A simple probabilistic argument shows that for some constant C (depending only on K), if

[1] See section 2.1 for a definition of agnostic learning.

[2] Note that agnostically learning parities when $\mathcal{D}$ is uniform is not equivalent to the problem that is usually referred as "learning parities with noise", since in agnostic learning, the noise might depend on the instance.

$m \geq Cn$, then J is not satisfiable w.h.p. The problem of *refuting random K-SAT formulas* (a.k.a. the problem of distinguishing satisfiable from random K-SAT formulas) seeks efficient algorithms that provide, for most formulas, a *refutation*. That is, a proof that the formula is not satisfiable.

Concretely, we say that an algorithm is able to refute random K-SAT instances with $m = m(n) \geq Cn$ clauses if on $1 - o_n(1)$ fraction of the K-SAT formulas with m constraints, it outputs "unsatisfiable", while for *every* satisfiable K-SAT formula with m constraints, it outputs "satisfiable"[3]. Since such an algorithm never errs on satisfiable formulas, an output of "unsatisfiable" provides a proof that the formula is not satisfiable.

The problem of refuting random K-SAT formulas has been extensively studied during the last 50 years. It is not hard to see that the problem gets easier as m gets larger. The currently best known algorithms Feige and Ofek (2004); Coja-Oghlan et al. (2004, 2010) can only refute random instances with $\Omega\left(n^{\lceil \frac{K}{2} \rceil}\right)$ constraints for $K \geq 4$ and $\Omega\left(n^{1.5}\right)$ constraints for $K = 3$. In light of that, Feige (2002) made the assumption that for $K = 3$, refuting random instances with Cn constraints, for every constant C, is hard (and used that to prove hardness of approximation results). Here, we put forward the following assumption.

Assumption 1 *Refuting random* K-SAT *formulas with $n^{f(K)}$ constraints is hard for some $f(K) = \omega(1)$.*

Terminology 2 *Computational problem is* RSAT-hard *if its tractability refutes assumption 1.*

We outline below some evidence to the assumption, in addition to known algorithms' performance.

Hardness of approximation. Define the *value*, $\mathrm{VAL}(J)$, of a K-SAT formula J as the maximal fraction of constraints that can be simultaneously satisfied. Hastad's celebrated result (Håstad, 2001) asserts that if $\mathbf{P} \neq \mathbf{NP}$, it is hard to distinguish satisfiable K-SAT instances from instances with $1 - 2^{-K} \leq \mathrm{VAL}(J) \leq 1 - 2^{-K} + \epsilon$. Since the value of a random formula is approximately $1 - 2^{-K}$, we can interpret Hastad's result as claiming that it is hard to distinguish satisfiable from "semi-random" K-SAT formulas (i.e., formulas whose value is approximately the value of a random formula). Therefore, assumption 1 can be seen as a strengthening of Hastad's result.

Resolution lower bounds. The length of resolution refutations of random K-SAT formulas have been extensively studied (e.g. Haken (1985); Beame and Pitassi (1996); Beame et al. (1998); Ben-Sasson and Wigderson (1999)). It is known (theorem 2.24 in Ben-Sasson (2001)) that random formulas with $n^{\frac{K}{2}-\epsilon}$ constraints only have exponentially long resolution refutations. This shows that a large family of algorithms (the so-called Davis-Putnam algorithms Davis et al. (1962)) cannot efficiently refute random formulas with $n^{\frac{K}{2}-\epsilon}$ constraints. These bounds can also be taken as an indication that random instances do not have short refutations in general, and therefore hard to refute.

Hierarchies and SOS lower bounds. A family of algorithms whose performance has been analyzed are convex relaxations (Buresh-Oppenheim et al., 2003; Schoenebeck, 2008; Alekhnovich et al., 2005) that belong to certain *hierarchies* of convex relaxations. Among those hierarchies, the strongest is the Lasserre hierarchy (a.k.a. Sum Of Squares). Algorithms from this family achieve state of the art results for the K-SAT problem and many similar problems. In Grigoriev (2001); Schoenebeck (2008) it is shown that relaxations in the Lasserre hierarchy that work in sub-exponential time cannot refute random formulas with $n^{\frac{K}{2}-\epsilon}$ constraints.

[3] See a precise definition in section 2.2

Lower bounds on statistical algorithms. Another family of algorithms whose performance has been analyzed are the so-called statistical algorithms. Similarly to hierarchies lower bounds, the results in Feldman et al. (2015) imply that statistical algorithms cannot refute random K-SAT formulas with $n^{\frac{K}{2}-\epsilon}$ constraints for any $\epsilon > 0$.

1.2. Results

Learning DNF's. A DNF *clause* is a conjunction of literals. A DNF *formula* is a disjunction of DNF clauses. Each DNF formula over n variables naturally induces a function on $\{\pm 1\}^n$. The *size* of a DNF clause is the the number of literals, and the size of a DNF formula is the sum of the sizes of its clauses. For $q : \mathbb{N} \to \mathbb{N}$, denote by $\mathrm{DNF}_{q(n)}$ the class of functions over $\{\pm 1\}^n$ that are realized by DNFs of size $\leq q(n)$. Also, $\mathrm{DNF}^{q(n)}$ is the class of functions that are realized by DNF formulas with $\leq q(n)$ clauses. Since each clause is of size at most n, $\mathrm{DNF}^{q(n)} \subset \mathrm{DNF}_{nq(n)}$.

Learning hypothesis classes consisting of poly sized DNF's formulas has been a major effort in computational learning theory (e.g. Valiant (1984); Klivans and Servedio (2001); Linial et al. (1989); Mansour (1995)). Already in Valiant's paper (Valiant, 1984), it is shown that for every constant q, DNF-formulas with $\leq q$ clauses can be learnt efficiently. As for lower bounds, *properly* learning DNF's is known to be hard (Pitt and Valiant, 1988). Yet, hardness of improperly learning DNF's formulas has remained a major open question. Here we show:

Theorem 3 *If $q(n) = \omega(\log(n))$ then learning $\mathrm{DNF}^{q(n)}$ is RSAT-hard.*

Since $\mathrm{DNF}^{q(n)} \subset \mathrm{DNF}_{nq(n)}$, we immediately conclude that learning DNF's of size, say, $\leq n\log^2(n)$, is RSAT-hard. By a simple scaling argument (e.g. Daniely et al. (2014)), we obtain an even stronger result:

Corollary 4 *For every $\epsilon > 0$, it is RSAT-hard to learn $\mathrm{DNF}_{n^\epsilon}$.*

Remark 5 *By boosting results (Schapire, 1989), hardness of improper learning is automatically very strong quantitatively. Namely, for every $c > 0$, it is hard to find a classifier with error $\leq \frac{1}{2} - \frac{1}{n^c}$. Put differently, making a random guess on each example, is essentially optimal.*

Additional results. Theorem 3 implies the hardness of several problems, in addition to DNFs.

Corollary 6 *Learning intersections of $\omega(\log(n))$ halfspaces over $\{\pm 1\}^n$ is RSAT-hard.*

Corollary 7 *Learning polynomial threshold functions over $\{0,1\}^n$ with support size $\omega(\log(n))$ is RSAT-hard.*

Corollary 8 *Agnostically learning conjunctions is RSAT-hard.*

Corollary 9 *Agnostically learning halfspaces over $\{\pm 1\}^n$ is RSAT-hard.*

Corollary 10 *Agnostically learning parities[4] is RSAT-hard.*

Corollary 11 *For every $\epsilon > 0$, learning automata of size n^ϵ is RSAT-hard.*

[4] A parity is any hypothesis of the form $h(x) = \Pi_{i\in S} x_i$ for some $S \subset [n]$.

Theorems 6 and 7 are direct consequences of theorem 3, as any function realized by a DNF formula with $q(n)$ can be also realized by an intersection of $q(n)$ halfspaces, or a polynomial threshold functions over $\{0,1\}^n$ with support size $q(n)$. Theorem 8 follows from theorem 3, as learning DNFs can be reduced to agnostically learning conjunctions (Lee et al., 1996). Theorem 9 follows from theorem 8, as conjunctions are a subclass of halfspaces. Theorem 10 follows from theorem 3 and Feldman et al. (2006), who showed that learning DNFs can be reduced to agnostically learning parities. Theorem 11 follows from theorem 3 by Pitt and Warmuth (1988), who showed that learning DNFs can be reduced to learning Automata.

1.3. Related work

As indicated above, hardness of learning is traditionally established based on cryptographic assumptions. The first such result follows from Goldreich et al. (1986), and show that if one-way functions exist, than it is hard to learn polynomial sized circuits. To prove lower bounds on simpler hypothesis classes, researchers had to rely on more concrete hardness assumptions. Kearns and Valiant (1989) were the first to prove such results. They showed that assuming the hardness of various cryptographic problems (breaking RSA, factoring Blum integers and detecting quadratic residues), it is hard to learn automata, constant depth threshold circuits, $\log$-depth circuits and boolean formulae. Kharitonov (1993) showed, under a relatively strong assumption on the complexity of factoring random Blum integers, that learning constant depth circuits (for unspecified constant) is hard. Klivans and Sherstov (2006) showed that, under the hardness of the shortest vector problem, learning intersections of polynomially many halfspaces is hard. By Feldman et al. (2006), it also follows that agnostically learning halfspaces is hard. Hardness of agnostically learning halfspaces also follows from the hardness of learning parities with noise (Kalai et al., 2005).

There is a large body of work on various variants of the standard (improper and distribution free) PAC model. Hardness of proper learning, when the learner must return a hypothesis from the learnt class, is much more understood (e.g. Khot and Saket (2008, 2011); Guruswami and Raghavendra (2006); Feldman et al. (2006); Pitt and Valiant (1988)). Hardness of learning with restrictions on the distribution were studied in, e.g., Klivans and Kothari (2014); Kalai et al. (2005); Kharitonov (1993). Hardness of learning when the learner can ask the label of unseen examples were studied in, e.g., Angluin and Kharitonov (1991); Kharitonov (1993).

Lower bounds using the technique we use in this paper initiated in Daniely et al. (2013, 2014). In Daniely et al. (2013) it was shown, under Feige's assumption, that if the number of examples is limited (even though information theoretically sufficient), then learning halfspaces over sparse vectors is hard. The full methodology we use here was presented in Daniely et al. (2014). They made a strong and general assumption, that says, roughly, that for every random CSP problem, if the number of random constraints is too small to provide short resolution proofs, then the SDP relaxation of Raghavendra (2008) has optimal approximation ratio. Under this assumption they concluded hardness results that are similar to the results presented here. Later on, Allen et al. (2015) refuted this assumption. On the other hand, Daniely (2016) proved inapproximabilty results for agnostically learning halfspaces assuming that random K-XOR is hard.

2. Preliminaries

2.1. PAC Learning

A *hypothesis class*, $\mathcal{H}$, is a series of collections of functions $\mathcal{H}_n \subset \{0,1\}^{\mathcal{X}_n},\ n = 1, 2, \ldots$. We often abuse notation and identify $\mathcal{H}$ with $\mathcal{H}_n$. The instance spaces $\mathcal{X}_n$ we consider are $\{\pm 1\}^n$, $\{0,1\}^n$ or $\mathcal{X}_{n,K}$ (see section 2.2). Distributions on $\mathcal{Z}_n := \mathcal{X}_n \times \{0,1\}$ are denoted $\mathcal{D}_n$. The error of $h : \mathcal{X}_n \to \{0,1\}$ is $\mathrm{Err}_{\mathcal{D}_n}(h) = \Pr_{(x,y)\sim\mathcal{D}_n}(h(x) \neq y)$. For a class $\mathcal{H}_n$, we let $\mathrm{Err}_{\mathcal{D}_n}(\mathcal{H}_n) = \min_{h\in\mathcal{H}_n} \mathrm{Err}_{\mathcal{D}_n}(h)$. We say that $\mathcal{D}_n$ is *realizable* by h (resp. $\mathcal{H}_n$) if $\mathrm{Err}_{\mathcal{D}_n}(h) = 0$ (resp. $\mathrm{Err}_{\mathcal{D}_n}(\mathcal{H}_n) = 0$). A *sample* is a sequence $S = \{(x_1, y_1), \ldots (x_m, y_m)\} \in \mathcal{Z}_n^m$. The *empirical error* of $h : \mathcal{X}_n \to \{0,1\}$ on S is $\mathrm{Err}_S(h) = \frac{1}{m}\sum_{i=1}^m 1(h(x_i) \neq y_i)$, and the empirical error of $\mathcal{H}_n$ on S is $\mathrm{Err}_S(\mathcal{H}_n) = \min_{h\in\mathcal{H}_n} \mathrm{Err}_S(h)$. We say that S is *realizable* by h (resp. $\mathcal{H}_n$) if $\mathrm{Err}_S(h) = 0$ (resp. $\mathrm{Err}_S(\mathcal{H}_n) = 0$).

A *learning algorithm*, $\mathcal{L}$, obtains an error, confidence and complexity parameters $0 < \epsilon < 1$, $0 < \delta < 1$, and n, as well as an oracle access to examples from an unknown distribution $\mathcal{D}_n$ on $\mathcal{Z}_n$. It should output a (description of) hypothesis $h : \mathcal{X}_n \to \{0,1\}$. We say that $\mathcal{L}$ *(PAC) learns* $\mathcal{H}$ if, for every realizable $\mathcal{D}_n$, w.p. $\geq 1-\delta$, $\mathcal{L}$ outputs a hypothesis with error $\leq \epsilon$. We say that $\mathcal{L}$ *agnostically learns* $\mathcal{H}$ if, for every $\mathcal{D}_n$, w.p. $\geq 1-\delta$, $\mathcal{L}$ outputs a hypothesis with error $\leq \mathrm{Err}_{\mathcal{D}_n}(\mathcal{H}) + \epsilon$. We say that $\mathcal{L}$ is *efficient* if it runs in time $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$, and outputs a hypothesis that can be evaluated in time $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$. Finally, $\mathcal{L}$ is *proper* if it always outputs a hypothesis in $\mathcal{H}$. Otherwise, we say that $\mathcal{L}$ is *improper*.

2.2. Random Constraints Satisfaction Problems

Let $\mathcal{X}_{n,K}$ be the collection of *(signed) K-tuples*, that is, vectors $x = [(\alpha_1, i_1), \ldots, (\alpha_K, i_K)]$ for $\alpha_1, \ldots, \alpha_K \in \{\pm 1\}$ and distinct $i_1, \ldots, i_K \in [n]$. For $j \in [K]$ we denote $x(j) = (x^1(j), x^2(j)) = (\alpha_j, i_j)$. Each $x \in \mathcal{X}_{n,K}$ defines a function $U_x : \{\pm 1\}^n \to \{\pm 1\}^K$ by $U_x(\psi) = (\alpha_1\psi_{i_1}, \ldots, \alpha_K\psi_{i_K})$.

Let $P : \{\pm 1\}^K \to \{0,1\}$ be some predicate. A *P-constraint* with n variables is a function $C : \{\pm 1\}^n \to \{0,1\}$ of the form $C(x) = P \circ U_x$ for some $x \in \mathcal{X}_{n,K}$. An instance to the *CSP problem* $\mathrm{CSP}(P)$ is a *P-formula*, i.e., a collection $J = \{C_1, \ldots, C_m\}$ of P-constraints (each is specified by a K-tuple). The goal is to find an assignment $\psi \in \{\pm 1\}^n$ that maximizes the fraction of satisfied constraints (i.e., constraints with $C_i(\psi) = 1$). We will allow CSP problems where P varies with n (but is still fixed for every n). For example, we can look of the $\lceil \log(n) \rceil$-SAT problem.

We will often consider the problem of distinguishing satisfiable from random P formulas (a.k.a. the problem of refuting random P formulas). Concretely, for $m : \mathbb{N} \to \mathbb{N}$, we say that the problem $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(P)$ is easy, if there exists an efficient randomized algorithm, $\mathcal{A}$, such that:

- If J is a satisfiable instance to $\mathrm{CSP}(P)$ with n variables and $m(n)$ constraints, then
$$\Pr_{\text{coins of } \mathcal{A}}(\mathcal{A}(J) = \text{“satisfiable”}) \geq \frac{3}{4}$$
- If J is a random[5] instance to $\mathrm{CSP}(P)$ with n variables and $m(n)$ constraints then, with probability $1 - o_n(1)$ over the choice of J,
$$\Pr_{\text{coins of } \mathcal{A}}(\mathcal{A}(J) = \text{“random”}) \geq \frac{3}{4}\ .$$

[5] To be precise, in a random formula with n variable and m constraints, the K-tuple defining each constraint is chosen uniformly, and independently from the other constraints.

2.3. The methodology of Daniely et al. (2014)

In this section we briefly survey the technique of Daniely et al. (2014, 2013) to prove hardness of improper learning. Let $\mathcal{D} = \{\mathcal{D}_n^{m(n)}\}_n$ be a polynomial ensemble of distributions, that is, $\mathcal{D}_n^{m(n)}$ is a distribution on $\mathcal{Z}_n^{m(n)}$ and $m(n) \le \mathrm{poly}(n)$. Think of $\mathcal{D}_n^{m(n)}$ as a distribution that generates samples that are far from being realizable. We say that it is hard to distinguish realizable from $\mathcal{D}$-random samples if there is no efficient randomized algorithm $\mathcal{A}$ with the following properties:

- For every realizable sample $S \in \mathcal{Z}_n^{m(n)}$, $\Pr_{\text{internal coins of } \mathcal{A}}\left(\mathcal{A}(S) = \text{“realizable”}\right) \ge \frac{3}{4}$.
- If $S \sim \mathcal{D}_n^{m(n)}$, then with probability $1 - o_n(1)$ over the choice of S, it holds that

$$\Pr_{\text{internal coins of } \mathcal{A}}\left(\mathcal{A}(S) = \text{“unrelizable”}\right) \ge \frac{3}{4} \,.$$

For $p : \mathbb{N} \to (0, \infty)$ and $1 > \beta > 0$, we say that $\mathcal{D}$ is $(p(n), \beta)$-*scattered* if, for large enough n, it holds that for every function $f : \mathcal{X}_n \to \{0,1\}$, $\Pr_{S \sim \mathcal{D}_n^{m(n)}}\left(\mathrm{Err}_S(f) \le \beta\right) \le 2^{-p(n)}$.

Example 1 *Let $\mathcal{D}_n$ be a distribution over $\mathcal{Z}_n$ such that if $(x, y) \sim \mathcal{D}_n$, then y is a Bernoulli r.v. with parameter $\frac{1}{2}$, independent from x. Let $\mathcal{D}_n^{m(n)}$ be the distribution over $\mathcal{Z}_n^{m(n)}$ obtained by taking $m(n)$ independent examples from $\mathcal{D}_n$. For $f : \mathcal{X}_n \to \{0,1\}$, $\Pr_{S \sim \mathcal{D}_n^{m(n)}}\left(\mathrm{Err}_S(f) \le \frac{1}{4}\right)$ is the probability of getting at most $\frac{m(n)}{4}$ heads in $m(n)$ independent tosses of a fair coin. By Hoeffding's bound, this probability is $\le 2^{-\frac{1}{8}m(n)}$. Therefore, $\mathcal{D} = \{\mathcal{D}_n^{m(n)}\}_n$ is $\left(\frac{1}{8}m(n), 1/4\right)$-scattered.*

Hardness of distinguishing realizable from scattered samples turns out to imply hardness of learning.

Theorem 12 *Daniely et al. (2014) Every hypothesis class that satisfies the following condition is not efficiently learnable. There exists $\beta > 0$ such that for every $d > 0$ there is an (n^d, β)-scattered ensemble $\mathcal{D}$ for which it is hard to distinguish between a $\mathcal{D}$-random sample and a realizable sample.*

The basic observation of Daniely et al. (2014, 2013) is that an efficient algorithm, running on a very scattered sample, will return a bad hypothesis w.h.p. The reason is that the output classifier has a short description, given by the polynomially many examples the algorithm uses. Hence, the number of hypotheses the algorithm *might return* is limited. Now, since the sample is scattered, all these hypotheses are likely to perform purely. Based on that observation, an efficient learning algorithm can efficiently distinguish realizable from scattered samples: We can simply run the algorithm on the given sample to obtain a classifier h. Now, if the sample is realizable, h will perform well. Otherwise, if the sample is scattered, h will perform purely. Relying on that, we will be able to distinguish between the two cases. For completeness, we include the proof of theorem 12 in section A.

3. Proof of theorem 3

3.1. An overview

Intuitively, the problem of distinguishing satisfiable from random formulas is similar to the problem of distinguishing realizable from random samples. In both problems, we try to distinguish rare and structured instances from very random and "messy" instances. The course of the proof is to reduce the first problem to the second. Concretely, we reduce the problem $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(\mathrm{SAT}_K)$ to the problem of distinguishing realizable (by $\mathrm{DNF}^{q(n)}$) samples from $(n^{d-2}, \frac{1}{4})$-scattered samples. With such a reduction at hand, assumption 1 and theorem 12, imply theorem 3.

CSP PROBLEMS AS LEARNING PROBLEMS

The main conceptual idea is to interpret CSP problems as learning problems. Let $P : \{\pm 1\}^K \to \{0,1\}$ be some predicate. Every $\psi \in \{\pm 1\}^n$ naturally defines $h_\psi : \mathcal{X}_{n,K} \to \{0,1\}$, by mapping each K-tuple x to the truth value of the corresponding constraint, given the assignment ψ. Namely, $h_\psi(x) = P \circ U_x(\psi)$. Finally, let $\mathcal{H}_P \subset \{0,1\}^{\mathcal{X}_{n,K}}$ be the hypothesis class $\mathcal{H}_P = \{h_\psi \mid \psi \in \{\pm 1\}^n\}$.

The problem $\mathrm{CSP}(P)$ can be now formulated as follows. Given $x_1, \ldots, x_m \in \mathcal{X}_{n,K}$, find $h_\psi \in \mathcal{H}_P$ with minimal error on the sample $(x_1, 1), \ldots, (x_m, 1)$. Now, the problem $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(P)$ is the problem of distinguishing a realizable sample from a random sample $(x_1, 1), \ldots, (x_m, 1) \in \mathcal{X}_{n,K} \times \{0,1\}$ where the different x_i's where chosen independently and uniformly from $\mathcal{X}_{n,K}$.

The above idea alone, applied on the problem $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(\mathrm{SAT}_K)$ (or other problems of the form $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(P)$), is still not enough to establish theorem 3, due to the two following points:

- In the case that sample $(x_1, 1), \ldots, (x_m, 1)$ is random, it is, in a sense, "very random". Yet, it is not scattered at all! Since all the labels are 1, the constant function 1 realizes the sample.
- We must argue about the class $\mathrm{DNF}^{q(n)}$ rather than the class $\mathcal{H}_P$.

Next, we explain how we address these two points.

MAKING THE SAMPLE SCATTERED

To address the first point, we reduce $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(\mathrm{SAT}_K)$ to a problem of the following form. For a predicate $P : \{\pm 1\}^K \to \{0,1\}$ we denote by $\mathrm{CSP}(P, \neg P)$ the problem whose instances are collections, J, of constraints, each of which is either P or $\neg P$ constraint, and the goal is to maximize the number of satisfied constraints. Denote by $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(P, \neg P)$ the problem of distinguishing[6] satisfiable from random formulas with n variables and $m(n)$ constraints. Here, in a random formula, each constraint is chosen w.p. $\frac{1}{2}$ to be a uniform P constraint and w.p. $\frac{1}{2}$ a uniform $\neg P$ constraint.

The advantage of the problem $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(P, \neg P)$ is that in the "learning formulation" from the previous section, it is the problem of distinguishing a realizable sample from a sample $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X}_{n,K} \times \{0,1\}$ where the pairs (x_i, y_i) where chosen at random, independently and uniformly. As explained in example 1, this sample is $(\frac{1}{8}m(n), \frac{1}{4})$-scattered.

We will consider the predicate $T_{K,M} : \{0,1\}^{KM} \to \{0,1\}$ defined by

$$T_{K,M}(z) = (z_1 \vee \ldots \vee z_K) \wedge (z_{K+1} \vee \ldots \vee z_{2K}) \wedge \ldots \wedge \left(z_{(M-1)K+1} \vee \ldots \vee z_{MK}\right) .$$

We reduce the problem $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(\mathrm{SAT}_K)$ to $\mathrm{CSP}^{\mathrm{rand}}_{n^{d-1}}(T_{K,q(n)}, \neg T_{K,q(n)})$. This is done in two steps. First, we reduce $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(\mathrm{SAT}_K)$ to $\mathrm{CSP}^{\mathrm{rand}}_{n^{d-1}}(T_{K,q(n)})$. This is done as follows. Given an instance $J = \{C_1, \ldots, C_{n^d}\}$ to $\mathrm{CSP}(\mathrm{SAT}_K)$, by a simple greedy procedure, we try to find n^{d-1} disjoint

[6] As in $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(P)$, in order to succeed, and algorithm must return "satisfiable" w.p. $\geq \frac{3}{4}$ on every satisfiable formula and "random" w.p. $\geq \frac{3}{4}$ on $1 - o_n(1)$ fraction of the random formulas.

subsets $J'_1, \ldots, J'_{n^{d-1}} \subset J$, such that for every t, J'_t consists of $q(n)$ constraints and each variable appears in at most one of the constraints in J'_t. Now, from every J'_t we construct $T_{K,q(n)}$-constraint that is the conjunction of all constraints in J'_t. As we show, if J is random, this procedure will succeed w.h.p. and will produce a random $T_{K,q(n)}$-formula. If J is satisfiable, this procedure will either fail or produce a satisfiable $T_{K,q(n)}$-formula.

The second step is to reduce $\text{CSP}^{\text{rand}}_{n^{d-1}}(T_{K,q(n)})$ to $\text{CSP}^{\text{rand}}_{n^{d-1}}(T_{K,q(n)}, \neg T_{K,q(n)})$. This is done by replacing each constraint, w.p. $\frac{1}{2}$, with a random $\neg P$ constraint. Clearly, if the original instance is a random instance to $\text{CSP}^{\text{rand}}_{n^{d-1}}(T_{K,q(n)})$, the produced instance is a random instance to $\text{CSP}^{\text{rand}}_{n^{d-1}}(T_{K,q(n)}, \neg T_{K,q(n)})$. Furthermore, if the original instance is satisfied by the assignment $\psi \in \{\pm 1\}^n$, the same ψ, w.h.p., will satisfy all the new constraints. The reason is that the predicate $\neg T_{K,q(n)}$ is positive on almost all inputs – namely, on $1 - \left(1 - 2^{-K}\right)^{q(n)}$ fraction of the inputs. Therefore the probability that a random $\neg T_{K,q(n)}$-constraint is satisfied by ψ is $1 - \left(1 - 2^{-K}\right)^{q(n)}$, and hence, the probability that all new constraints are satisfied by ψ is $\geq 1 - n^{d-1}\left(1 - 2^{-K}\right)^{q(n)}$. Now, since $q(n) = \omega(\log(n))$, the last probability is $1 - o_n(1)$.

REDUCING $\mathcal{H}_P$ TO $\text{DNF}^{q(n)}$

To address the second point, we will realize $\mathcal{H}_{\neg T_{K,q(n)}}$ by the class $\text{DNF}^{q(n)}$. More generally, we will show that for every predicate $P : \{\pm 1\}^K \to \{0, 1\}$ expressible by a DNF formula with T clauses, $\mathcal{H}_P$ can be realized by DNF formulas with T clauses (note that for $\neg T_{K,q(n)}$, $T = q(n)$).

We first note that hypotheses in $\mathcal{H}_P$ are defined over signed K-tuples, while DNF's are defined over the boolean cube. To overcome that, we will construct an (efficiently computable) mapping $g : \mathcal{X}_{n,K} \to \{\pm 1\}^{2Kn}$, and show that each $h \in \mathcal{H}_P$ is of the form $h = h' \circ g$ for some DNF formula h' with T clauses and $2Kn$ variables. Besides "fixing the domain", g will have additional role – we will choose an expressive g, which will help us to realize hypotheses in $\mathcal{H}_P$. In a sense, g will be a first layer of computation, that is the same for all $h \in \mathcal{H}_P$ (and therefore we do not "pay" for it).

We will group the coordinates of vectors in $\{\pm 1\}^{2Kn}$ into $2K$ groups, corresponding to P's literals, and index them by $[K] \times \{\pm 1\} \times [n]$. For $x = [(\alpha_1, i_1), \ldots, (\alpha_K, i_K)] \in \mathcal{X}_{n,K}$, $g(x)$ will be the vector whose all coordinates are 1, except that for $j \in [K]$, the $(j, -\alpha_j, i_j)$ coordinate is -1.

Now, given $\psi \in \{\pm 1\}^n$, we show that $h_\psi : \mathcal{X}_{n,K} \to \{0, 1\}$ equals to $h \circ g$ for a DNF formula h with T clauses. Indeed, suppose that $P(x) = C_1(x) \vee \ldots \vee C_T(x)$ is a DNF representation of P. It is enough to show that for every $C_r(z) = (-1)^{\beta_1} z_{j_1} \wedge \ldots \wedge (-1)^{\beta_l} z_{j_l}$ there is a conjunction of literals $h_r : \{\pm 1\}^{2Kn} \to \{0, 1\}$ such that for all $x = [(\alpha_1, i_1), \ldots, (\alpha_K, i_K)] \in \mathcal{X}_{n,K}$, $h_r(g(x)) = C_r(U_x(\psi))$. To see that such h_r exists, note that $C_r(U_x(\psi)) = 1$ if and only if, for every $1 \leq \tau \leq l$, all the values in $g(x)$ in the coordinates of the form $(j_\tau, \psi_i(-1)^{\beta_\tau}, i)$ are 1.

3.2. From $\text{CSP}^{\text{rand}}_{n^d}(\text{SAT}_K)$ to $\text{CSP}^{\text{rand}}_{n^{d-1}}(T_{K,\frac{n}{\log(n)}})$

Lemma 13 *The problem $\text{CSP}^{\text{rand}}_{n^d}(\text{SAT}_K)$ can be reduced to $\text{CSP}^{\text{rand}}_{n^{d-1}}(T_{K,M})$ for any $M \leq \frac{n}{\log(n)}$.*

It will be convenient to use the following strengthening of Chernoff's bound, recently proved (with a very simple proof) by Linial and Luria (2014)

Theorem 14 *Linial and Luria (2014) Let $X_1 \dots, X_n$ be indicator r.v. such that for all $S \subset [n]$, $\Pr\left(\forall i \in S,\ X_i = 1\right) \le \alpha^{|S|}$. Then, for every $\beta > \alpha$, $\Pr\left(\frac{1}{n}\sum_{i=1}^{n} X_i \ge \beta\right) \le \exp(-D(\beta||\alpha)n) \le \exp(-2(\beta-\alpha)^2 n)$.*

Proof For simplicity, we assume that $M = \frac{n}{\log(n)}$. Suppose toward a contradiction that $\mathrm{CSP}^{\mathrm{rand}}_{n^{d-1}}(T_{K,\frac{n}{\log(n)}})$ can be efficiently solved using an algorithm $\mathcal{A}$. Consider the following algorithm, $\mathcal{A}'$, to $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(\mathrm{SAT}_K)$. On the input $J = \{C_1, \dots, C_{n^d}\}$,

1. Partition the constraints in J into n^{d-1} blocks, $\{C_{t+1}, \dots, C_{t+n}\}$, $t = 1, 2, \dots, n^{d-1}$.
2. For $t = 1, \dots, n^{d-1}$
   - (a) Let $J'_t = \emptyset$.
   - (b) For $r = 1, \dots, n$
     - i. If $|J'_t| < \frac{n}{\log(n)}$ and, for all $C \in J'_t$, the set variables appearing in C_{t+r} is disjoint from the set of variables appearing in C, add C_{t+r} to J'_t.
   - (c) If $|J'_t| < \frac{n}{\log(n)}$, return "satisfiable".
   - (d) Let C'_t be the $T_{K,\lceil\frac{n}{\log(n)}\rceil}$-constraint which is the conjunction of all the constraints in J'_t.
3. Run $\mathcal{A}$ on the instance $J' = \{C'_1, \dots, C'_{n^{d-1}}\}$ and return the same answer as $\mathcal{A}$.

Next, we reach a contradiction as we prove that $\mathcal{A}'$ solves the problem $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(\mathrm{SAT}_K)$. First, suppose that the input, J, is satisfiable. Then, either $\mathcal{A}'$ will return "satisfiable" in step 2c or, will run $\mathcal{A}$ on J'. It is not hard to see that J' is satisfiable as well, and therefore, $\mathcal{A}$ (and therefore $\mathcal{A}'$) will return "satisfiable" w.p. $\ge \frac{3}{4}$.

Suppose now that J is random. First, we claim that $\mathcal{A}'$ will reach 3 w.p. $\ge 1 - o_n(1)$. Indeed, we will show that for large enough n and any fixed t, the probability of exiting at step 2c is $\le \exp\left(-\left(\frac{1}{2^{2K+5}K}\right)^2 n\right)$, from which it follows that the probability of exiting at step 2c for some t is $o_n(1)$. To show that, let X_r, $r = 1, \dots, n$ be the indicator r.v. that is 1 if and only if one of the variables appearing in C_{t+r} also appears in one of $C_{t+1}, \dots, C_{t+r-1}$. Denote also $\bar{X}_r = 1 - X_r$

Let $n' = \lfloor \frac{n}{2K} \rfloor$. It is enough to show that $\sum_{r=1}^{n'} \bar{X}_r \ge \frac{n}{\log(n)}$ w.p. $\ge 1 - \exp\left(-\left(\frac{1}{2^{2K+5}K}\right)^2 n\right)$. Indeed, for every fixed $r \in [n']$, since the number of variables appearing in $C_{t+1}, \dots, C_{t+r-1}$ is $\le \frac{n}{2}$, the probability that $X_r = 1$ is $\le 1 - 2^{-K}$, even if we condition on $X_1, \dots, X_{r-1}$. Hence, the probability that any fixed u variables out of $X_1, \dots, X_{n'}$ are all 1 is $\le \left(1 - 2^{-K}\right)^u$. By theorem 14,

$$\Pr\left(\frac{1}{n'}\sum_{i=1}^{n'} X_i \ge 1 - 2^{-K} + 2^{-K+1}\right) \le \exp\left(-2\left(2^{-K+1}\right)^2 n'\right) \le \exp\left(-\left(\frac{1}{2^{2K+5}K}\right)^2 n\right) .$$

It follows that w.p. $\ge 1 - \exp\left(-\left(\frac{1}{2^{2K+5}K}\right)^2 n\right)$, $\sum_{r=1}^{n'} \bar{X}_r \ge \frac{n'}{2^{K+1}}$, and the claim follows as for sufficiently large n, $\frac{n'}{2^{K+1}} \ge \frac{n}{\log(n)}$. Finally, it is not hard to see that, conditioning on the event that the algorithm reaches step 3, J' is random as well, and therefore w.p. $\ge 1 - o_n(1)$ over the choice of J, $\mathcal{A}$ (and therefore $\mathcal{A}'$) will return "random" w.p. $\ge \frac{3}{4}$ over its internal randomness.

■

3.3. From $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(T_{K,M})$ to $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(T_{K,M},\neg T_{K,M})$

Lemma 15 *For any fixed K and $M \ge 2^{K+2}\cdot\log(m(n))$, the problem $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(T_{K,M})$ can be efficiently reduced to the problem $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(T_{K,M},\neg T_{K,M})$*

Proof Given an instance $J=\{C_1,\ldots,C_m\}$ to $\mathrm{CSP}(T_{K,M})$, the reduction will generate an instance to $\mathrm{CSP}(T_{K,M},\neg T_{K,M})$ as follows. For each C_i, w.p. $\frac{1}{2}$, we substitute C_i by a random $\neg T_{K,M}$ constraint. Clearly, if J is a random formula, then the produced formula is a valid random formula to $\mathrm{CSP}^{\mathrm{rand}}_{m(n)}(T_{K,M},\neg T_{K,M})$. It remains to show that if J is satisfiable, then so is J'. Indeed, let $\psi\in\{\pm1\}^n$ be a satisfying assignment to J. It is enough to show that w.p. $\ge\frac{1}{m(n)}$, ψ satisfies all the new $\neg T_{K,M}$- constraints. However, since $\left|(\neg T_{K,M})^{-1}(0)\right| = \left(2^K-1\right)^M = \left(1-2^{-K}\right)^M\cdot 2^{MK}$, the probability that a single random constraint is not satisfied is $\left(1-2^{-K}\right)^M$. It follows that the probability that one of the random $\neg T_{K,M}$ constraints in J' is not satisfiable by ψ is $\le m(n)\left(1-2^{-K}\right)^M$. Finally, we have $m(n)\left(1-2^{-K}\right)^M\le\frac{1}{m(n)}$ since,

$$\begin{aligned}
\log\left(m(n)\left(1-2^{-K}\right)^M\right) &= \log(m(n)) - M\log\left(\frac{1}{1-2^{-K}}\right)\\
&= \log(m(n)) - M\log\left(1+\frac{2^{-K}}{1-2^{-K}}\right)\\
&\le \log(m(n)) - M\frac{2^{-K}}{1-2^{-K}}\\
&\le \log(m(n)) - M2^{-(K+1)}\\
&\le \log(m(n)) - 2\log(m(n)) = \log\left(\frac{1}{m(n)}\right)
\end{aligned}$$

■

3.4. From $\mathrm{CSP}^{\mathrm{rand}}_{n^d}(T_{K,M},\neg T_{K,M})$ to DNF's

Lemma 16 *Suppose that $P:\{\pm1\}^K\to\{0,1\}$ can be realized by a DNF formula with T clauses. Then $\mathcal{H}_P$ can be efficiently realized*[7] *by the class of DNF formulas with T clauses and $2Kn$ variables.*

Proof The realization is given by a function $g:\mathcal{X}_{n,K}\to\{\pm1\}^{2Kn}$, defined as follows. We will index the coordinates of vectors in $\{\pm1\}^{2Kn}$ by $[K]\times\{\pm1\}\times[n]$ and let $g_{j,b,i}(x)=\begin{cases}-1 & x(j)=(-b,i)\\ 1 & \text{otherwise}\end{cases}$.

To see that g indeed defines a realization of $\mathcal{H}_P$ by the class of DNF formulas with T clauses, we must show that for any assignment $\psi\in\{\pm1\}^n$, $h_\psi=h\circ g$ for some DNF formula h with T clauses. Indeed, write $P(z_1,\ldots,z_K)=\vee_{t=1}^{T}\wedge_{r=1}^{R_t} b_{t,r}z_{j_{t,r}}$ for $b_{t,r}\in\{\pm1\}$ and $i_{t,r}\in[K]$. Now

[7] That is, there is an efficiently computable $g:\mathcal{X}_{n,K}\to\{\pm1\}^{2Kn}$ for which each $h\in\mathcal{H}_P$ is of the form $h=h'\circ g$ for some DNF formula g with T clauses and $2Kn$ variables.

consider the formula $h : \{\pm 1\}^{2Kn} \to \{0,1\}$ defined by $h(x) = \vee_{t=1}^{T} \wedge_{r=1}^{R_t} \wedge_{i=1}^{n} x_{j_{t,r},\psi_i b_{t,r},i}$. For $x \in \mathcal{X}_{n,K}$ we have,

$$\begin{aligned} h(g(x)) = 1 \quad &\iff \exists t \in [T]\, \forall r \in [R_t], i \in [n],\ g_{j_{t,r},\psi_i b_{t,r},i}(x) = 1 \\ &\iff \exists t \in [T]\, \forall r \in [R_t], i \in [n],\ x(j_{t,r}) \neq (-\psi_i b_{t,r}, i) \\ &\iff \exists t \in [T]\, \forall r \in [R_t],\ \ x_1(j_{t,r}) \neq -\psi_{x_2(j_{t,r})} b_{t,r} \\ &\iff \exists t \in [T]\, \forall r \in [R_t],\ x_1(j_{t,r}) \psi_{x_2(j_{t,r})} = b_{t,r} \\ &\iff h_\psi(x) = x(\psi) = P(x_1(1)\psi_{x_2(1)}, \ldots, x_1(K)\psi_{x_2(K)}) = 1\ . \end{aligned}$$

□

3.5. Wrapping up – concluding theorem 3

We are now ready to conclude the proof. Let $q : \mathbb{N} \to \mathbb{N}$ be any function such that $q(n) = \omega(\log(n))$. W.l.o.g., we assume that $q(n) = O\left(\log^2(n)\right)$. By theorem 12 it is enough to show that for every d, it is hard to distinguish samples that are realizable by $\mathrm{DNF}^{q(n)}$ and $\left(n^d, 1/4\right)$-scattered samples.

By assumption 1, there is K such that $\mathrm{CSP}^{\mathrm{rand}}_{n^{d+2}}(\mathrm{SAT}_K)$ is hard. Denote $q'(n) = q(2Kn)$. By lemma 13, the problem $\mathrm{CSP}^{\mathrm{rand}}_{n^{d+1}}(T_{K,q'(n)})$ is hard. By lemma 15, the problem $\mathrm{CSP}^{\mathrm{rand}}_{n^{d+1}}(T_{K,q'(n)}, \neg T_{K,q'(n)})$ is hard. Now, since $\neg T_{K,q'(n)}$ can be realized by a DNF formula with $q'(n)$ clauses, by lemma 16, the problem $\mathrm{CSP}^{\mathrm{rand}}_{n^{d+1}}(T_{K,q'(n)}, \neg T_{K,q'(n)})$ can be reduced to a problem of distinguishing samples that are realizable by a DNF formula with $2Kn$ variables and $q'(n)$ clauses, from $\left(\frac{1}{8}n^{d+1}, 1/4\right)$-scattered samples. Changing variables (i.e., replacing $2Kn$ with n'), we conclude that it is hard to distinguish samples that are realizable by $\mathrm{DNF}^{q(n)}$ from $\left(\frac{1}{8(2K)^{d-1}}n^{d+1}, 1/4\right)$-scattered samples, which are in particular $\left(n^d, 1/4\right)$-scattered. The theorem follows.

4. Open questions

Basic learning problems that we are unable to resolve even under the random K-SAT (or K-XOR) assumption include decision trees and intersections of a constantly many halfspaces. (It is worth noting that no known algorithm can learn even intersections of 2 halfspaces).

Acknowledgments

Amit Daniely was a recipient of the Google Europe Fellowship in Learning Theory, and this research was supported in part by this Google Fellowship. Shai Shalev-Shwartz is supported by the Israeli Science Foundation grant number 590-10. We thank Uri Feige, Guy Kindler and Nati Linial for valuable discussions.

References

Mikhail Alekhnovich, Sanjeev Arora, and Iannis Tourlakis. Towards strong nonapproximability results in the lovász-schrijver hierarchy. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 294–303. ACM, 2005.

Sarah Allen, Ryan O'Donnell, and David Witmer. How to refute a random csp? In *FOCS*, 2015.

Dana Angluin and Michael Kharitonov. When won't membership queries help? In *STOC*, pages 444–454, May 1991.

B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 211–220. IEEE, 2008.

Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 274–282. IEEE, 1996.

Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. On the complexity of unsatisfiability proofs for random k-cnf formulas. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 561–571. ACM, 1998.

Eli Ben-Sasson. Expansion in proof complexity. In *Hebrew University*. Citeseer, 2001.

Eli Ben-Sasson and Avi Wigderson. Short proofs are narrowresolution made simple. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 517–526. ACM, 1999.

Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

Joshua Buresh-Oppenheim, Nicola Galesi, Shlomo Hoory, Avner Magen, and Toniann Pitassi. Rank bounds and integrality gaps for cutting planes procedures. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 318–327. IEEE, 2003.

Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong refutation heuristics for random k-sat. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 310–321. Springer, 2004.

Amin Coja-Oghlan, Colin Cooper, and Alan Frieze. An efficient sparse regularity concept. *SIAM Journal on Discrete Mathematics*, 23(4):2000–2034, 2010.

Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *STOC*, 2016.

Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. More data speeds up training time in learning halfspaces over sparse vectors. In *NIPS*, 2013.

Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *STOC*, 2014.

Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM, 2002.

Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3cnf formulas. In *Automata, languages and programming*, pages 519–530. Springer, 2004.

V. Feldman, P. Gopalan, S. Khot, and A.K. Ponnuswami. New results for learning noisy parities and halfspaces. In *In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

Vitaly Feldman, Will Perkins, and Santosh Vempala. On the complexity of random satisfiability problems with planted solutions. In *STOC*, 2015.

Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, October 1986.

Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1):613–622, 2001.

V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of the 47th Foundations of Computer Science (FOCS)*, 2006.

Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

A. Kalai, A.R. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. In *Proceedings of the 46th Foundations of Computer Science (FOCS)*, 2005.

Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *STOC*, pages 433–444, May 1989.

Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 372–381. ACM, 1993.

Subhash Khot and Rishi Saket. Hardness of minimizing and learning dnf expressions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 231–240. IEEE, 2008.

Subhash Khot and Rishi Saket. On the hardness of learning intersections of two halfspaces. *Journal of Computer and System Sciences*, 77(1):129–141, 2011.

Adam Klivans and Pravesh Kothari. Embedding hard learning problems into gaussian space. In *RANDOM*, 2014.

Adam R Klivans and Rocco Servedio. Learning dnf in time $2^{O(n^{1/3})}$. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 258–265. ACM, 2001.

Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *FOCS*, 2006.

Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. Efficient agnostic learning of neural networks with bounded fan-in. *IEEE Transactions on Information Theory*, 42(6):2118–2132, 1996.

N. Linial and Z. Luria. Chernoff's Inequality - A very elementary proof. *Arxiv preprint arXiv:1403.7739 v2*, 2014.

Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. In *FOCS*, pages 574–579, October 1989.

Yishay Mansour. An $o(n \log \log n)$ learning algorithm for dnf under the uniform distribution. *Journal of Computer and System Sciences*, 50(3):543–550, 1995.

L. Pitt and L.G. Valiant. Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35(4):965–984, October 1988.

Leonard Pitt and Manfred K. Warmuth. Prediction preserving reducibility. Technical Report UCSC-CRL-88-26, University of California Santa Cruz, Computer Research Laboratory, November 1988.

Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 245–254. ACM, 2008.

R.E. Schapire. The strength of weak learnability. In *FOCS*, pages 28–33, October 1989.

Grant Schoenebeck. Linear level lasserre lower bounds for certain k-csps. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 593–602. IEEE, 2008.

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

Appendix A. Proof of Theorem 12 (Daniely et al., 2014)

Let $\mathcal{H}$ be the hypothesis class in question and suppose toward a contradiction that algorithm $\mathcal{L}$ learns $\mathcal{H}$ efficiently. Let $M(n, 1/\epsilon, 1/\delta)$ be the maximal number of random bits used by $\mathcal{L}$ when it run on the input n, ϵ, δ. This includes both the bits describing the examples produced by the oracle and "standard" random bits. Since $\mathcal{L}$ is efficient, $M(n, 1/\epsilon, 1/\delta) < \text{poly}(n, 1/\epsilon, 1/\delta)$. Define

$$q(n) = M(n, 1/\beta, 4) + n \ .$$

By assumption, there is a $(q(n), \beta)$-scattered ensemble $\mathcal{D}$ for which it is hard to distinguish a $\mathcal{D}$-random sample from a realizable sample. Consider the algorithm $\mathcal{A}$ defined below. On input $S \in \mathcal{Z}_n^{m(n)}$,

1. Run $\mathcal{L}$ with parameters n, β and $\frac{1}{4}$, such that the examples' oracle generates examples by choosing a random example from S.
2. Let h be the hypothesis that $\mathcal{L}$ returns. If $\text{Err}_S(h) \leq \beta$, output "realizable". Otherwise, output "unrealizable".

Next, we derive a contradiction by showing that $\mathcal{A}$ distinguishes a realizable sample from a $\mathcal{D}$-random sample. Indeed, if the input S is realizable, then $\mathcal{L}$ is guaranteed to return, with probability $\geq 1 - \frac{1}{4}$, a hypothesis $h : \mathcal{X}_n \to \{0, 1\}$ with $\mathrm{Err}_S(h) \leq \beta$. Therefore, w.p. $\geq \frac{3}{4}$ $\mathcal{A}$ will output "realizable".

What if the input sample S is drawn from $\mathcal{D}_n^{m(n)}$? Let $\mathcal{G} \subset \{0, 1\}^{\mathcal{X}_n}$ be the collection of functions that $\mathcal{L}$ might return when run with parameters $n, \epsilon(n)$ and $\frac{1}{4}$. We note that $|\mathcal{G}| \leq 2^{q(n)-n}$, since each hypothesis in $\mathcal{G}$ can be described by $q(n) - n$ bits. Namely, the random bits that $\mathcal{L}$ uses and the description of the examples sampled by the oracle. Now, since $\mathcal{D}$ is $(q(n), \beta)$-scattered, the probability that $\mathrm{Err}_S(h) \leq \beta$ for some $h \in \mathcal{G}$ is at most $|\mathcal{G}| 2^{-q(n)} \leq 2^{-n}$. It follows that the probability that $\mathcal{A}$ responds "realizable" is $\leq 2^{-n}$. This leads to the desired contradiction and concludes our proof.