Lower Bounds on Learning Random Structures with Statistical Queries

Dana Angluin[1⋆], David Eisenstat[2], Leonid (Aryeh) Kontorovich[3], and Lev Reyzin[4⋆⋆]

[1] Yale University, New Haven CT USA
dana.angluin@yale.edu
[2] Brown University, Providence RI USA
eisenstatdavid@gmail.com
[3] Ben Gurion University of the Negev, Israel
karyeh@cs.bgu.ac.il
[4] Yahoo! Research, New York NY, USA
lreyzin@yahoo-inc.com

Abstract. We show that random DNF formulas, random log-depth decision trees and random deterministic finite acceptors cannot be weakly learned with a polynomial number of statistical queries with respect to an arbitrary distribution on examples.

1 Introduction

Polynomial time learning algorithms have been given for random log-depth decision trees by Jackson and Servedio [6], random monotone DNF formulas by Jackson et al. [5] and Sellie [12] and random general DNF formulas by Sellie [12], with respect to the uniform distribution. These algorithms are based on statistical estimates of various parameters and can be implemented using statistical queries as defined by Kearns [8].

Blum et al. [2] give upper and lower bounds on the number of statistical queries required to learn concepts from a given class in terms of a distribution-dependent statistical query dimension of the class. A corollary of their characterization is that parity functions with $\log n$ relevant variables cannot be weakly learned with respect to the uniform distribution using a polynomial number of statistical queries. This implies that arbitrary log-depth decision trees and DNF formulas with $\Theta(n)$ terms are not weakly learnable with respect to the uniform distribution using a polynomial number of statistical queries, because they can represent parity functions with $\log n$ relevant variables.

The key difference between these negative results and the positive results cited above is that the choice of a structure (DNF formula or decision tree) to be learned is random. In particular, "bad structures" (capable of representing a parity problem with $\log n$ relevant variables with respect to the uniform distribution) occur with vanishing probability as n increases.

⋆ Research supported by the National Science Foundation under Grant CCF-0916389.
⋆⋆ This material is based upon work supported by the National Science Foundation under a National Science Foundation Graduate Research Fellowship and under Grant # 0937060 to the Computing Research Association for the Computing Innovation Fellowship program.

1.1 Learning DFAs

A deterministic finite acceptor M over the alphabet $\{0,1\}$ can be used to represent the set $L(M) \cap \{0,1\}^n$ of accepted binary strings of length n. Gold [4] gave one of the earliest hardness results for learning DFAs, that finding a smallest DFA consistent with given positive and negative data is NP-hard. The PAC-reduction given by Pitt and Warmuth [10] of learning Boolean formulas to learning DFAs, combined with the negative results of Kearns and Valiant [7] for PAC-learning Boolean formulas, gives cryptographic evidence for the hardness of PAC-learning arbitrary DFAs.

Because a DFA of $2n+1$ states can compute the parity of an arbitrary subset of a string of n bits, the results of Blum et al. [2] imply that there is no algorithm to learn arbitrary DFAs using a polynomial number of statistical queries with respect to the uniform distribution. In light of the positive results described above for random DNF formulas and random decision trees, it is natural to ask how hard it is to learn random DFAs of $O(n^c)$ states with respect to the uniform distribution over strings of length n.

Trakhtenbrot and Barzdin [13] consider the problem of learning arbitrary finite automata using experiments – in each experiment the learning algorithm selects an input string to query and receives the sequence of output symbols generated by the target machine on reading that input. For a DFA, this is equivalent to a sequence of membership queries on all prefixes of the experiment string. Trakhtenbrot and Barzdin also consider the problem of learning almost all automata under a natural distribution on DFAs.

Building on these results, Freund et al. [3] consider a model in which the target is a DFA with an arbitrary transition graph and states independently labeled as accepting or rejecting with probability $1/2$. The learner sees the outputs of a random walk in the transition graph, and must at each step predict the next output. Freund et al. show that there is an algorithm that makes a number of mistakes bounded by a polynomial in n for uniformly almost all automata of n states. In an empirical study of the effectiveness of a learning algorithm based on Trakhtenbrot and Barzdin's contraction algorithm, Lang [9] studied what fraction of all $16n^2-1$ binary strings of length at most $2\log n+3$ is needed to achieve high levels of generalization for randomly generated machines of about n states.

These results do not directly shed light on the question of how difficult it is to learn a set of binary strings of length n accepted by a random DFA of $O(n^c)$ states with respect to the uniform distribution. We are interested in this problem, which is open to the best of our knowledge.

1.2 Lower bounds

In this paper we show that no algorithm using a polynomial number of statistical queries can learn random DNF formulas, random decision trees, or random DFAs with respect to an arbitrary distribution. Thus the random choice of a structure is not sufficient for the positive results cited above for DNF formulas and decision trees. Even if the structure is randomly chosen, it may be possible to find a "bad input distribution" that allows a hard parity problem to be embedded in that random structure. This situation is a natural one to consider in the case of a boosting algorithm attempting to learn a random structure,

because successive rounds of boosting may modify an initially simple input distribution in complex ways that depend on the target concept.

Specifically, we consider the problem of learning the behavior of a random structure using statistical queries, where the distribution on examples is adversarial, and therefore may depend on the random structure to be learned. For the cases when the random structure is a DNF formula, a log-depth decision tree, or a DFA we show that with at least a constant probability, there is a distribution on the inputs that embeds a nontrivial parity computation in the random structure. In general the "bad" distribution constrains some variables to constant values and some variables to copy other variables or their negations, and is uniform on the rest. These results provide some support for the distributional assumptions made in the positive results of Jackson and Servedio [6], Jackson et al. [5] and Sellie [11, 12].

2 Preliminaries

We consider concept classes over binary strings. Let $\Sigma = \{0, 1\}$. Σ^* is the set of all binary strings and ε is the empty string. The set of all binary strings of length n, length at most n and length less than n are denoted Σ^n, $\Sigma^{\leq n}$ and $\Sigma^{<n}$. A concept class $\mathcal{C}$ is a collection $\{\mathcal{C}_n\}$ indexed by the positive integer n, where each $\mathcal{C}_n$ is a set of concepts over Σ^n (or $\Sigma^{\leq n}$) that is, a set of mappings f from Σ^n (or $\Sigma^{\leq n}$) to Σ. Concept classes are generally specified by some representation, for example, log depth decision trees over n variables. A concept class has polynomially bounded representations if there is a fixed polynomial p such that every concept f in $\mathcal{C}_n$ has a representation of length bounded by $p(n)$.

2.1 Learning with Statistical Queries

Let $X = \Sigma^n$ or $\Sigma^{\leq n}$, let D be a probability distribution over X and f a mapping from X to $\{0, 1\}$. The goal of learning is to be able to predict $f(x)$ well when x is drawn according to D. Statistical queries provide a particular way of gathering information about the function f. They were introduced by Kearns to characterize a wide class of noise tolerant learning algorithms [8].

The statistics oracle $\mathrm{STAT}(f, D)$ answers statistical queries, each of which specifies two arguments: a predicate χ mapping $X \times \{0, 1\}$ to $\{0, 1\}$, and a tolerance $\tau \in [0, 1]$. The answer returned by the oracle is any number v such that

$$|\mathrm{E}_D[\chi(x, f(x))] - v| \leq \tau.$$

That is, the oracle may return any number v within an additive error τ of the expected value of χ on a labeled example $(x, f(x))$ where x is drawn according to D and classified using f. A statistical query abstracts the process of drawing a sample of labeled examples $(x, f(x))$ according to D to estimate the probability that they satisfy the predicate χ. For example, a statistical query might be used to estimate the probability that the conjunction of two literals is equal to the label of an example.

A concept class $\mathcal{C}$ with polynomially bounded representations is (strongly) learnable in polynomial time using statistical queries if there exists a polynomial p and a

learning algorithm A such that for every positive integer n, for every $f \in \mathcal{C}_n$, for every probability distribution D on Σ^n and for every $\epsilon > 0$, A with inputs n and ϵ and access to the statistics oracle $\mathrm{STAT}(f, D)$ satisfies the following properties.

1. For every statistical query (χ, τ) made by A, the predicate χ can be evaluated in time bounded by $p(1/\epsilon, n)$ and $1/\tau < p(1/\epsilon, n)$.
2. A runs in time bounded by $p(1/\epsilon, n)$.
3. A outputs a hypothesis h such that $h(x)$ can be evaluated in time bounded by $p(1/\epsilon, n)$ and when x is drawn from D, the probability that $h(x) \neq f(x)$ is at most ϵ.

For a concept class that does not have polynomially bounded representations, the polynomial p has an additional parameter $\mathrm{size}(f)$ bounding the length of the representation of the target concept f. For a randomized learning algorithm A we require that A return an ϵ-good hypothesis h with at least an inverse polynomial probability. A single statistical query suffices to estimate the accuracy of a candidate hypothesis, allowing the probability of success to be boosted easily using repeated runs.

To define weak learnability, we omit ϵ and ask that the probability that $h(x) \neq f(x)$ be bounded by $1/2 - 1/q(n)$ for some polynomial q. That is, the error rate in this case need only be better than $1/2$ by an inverse polynomial. Polynomial time weak learnability using statistical queries has been shown to imply polynomial time strong learnability using statistical queries by Aslam and Decatur [1].

To define learnability using a polynomial number of statistical queries, we replace the requirement that A run in polynomial time with the requirement that A make at most a polynomial number of statistical queries. We still require that each statistical query have a polynomial time evaluatable predicate and an inverse polynomial tolerance.

To define learnability of random target concepts, we assume that there is a probability distribution on the elements of $\mathcal{C}_n$ that determines the choice of the target concept f. We require that the probability that A fails to satisfy the required conditions be $o(1)$ as a function of n. Thus, there may be a subset of the concepts in $\mathcal{C}_n$ on which A always fails, but the probability of that set must tend to zero as n increases.

2.2 The Parity Learning Problem

A parity function over n variables with ℓ relevant variables is a mapping from Σ^n to $\{0, 1\}$ that is equal to the sum modulo 2 of a fixed subset of ℓ of its arguments. For the problem of learning a parity function ψ over n variables with $\ell(n)$ relevant variables with respect to the uniform distribution on examples, a learning algorithm is given n, $\ell(n)$ and access to the statistics oracle $\mathrm{STAT}(\psi, U)$, where U is the uniform distribution over Σ^n. When $\ell(n) = \Theta(\log n)$, no learning algorithm, even a randomized one, can achieve weak learning for this problem using polynomially many statistical queries [2]

3 Random DNF formulas

In this section we prove the lower bound for random DNF formulas. The embedding is quite straightforward in this case, and highlights the general framework of the reduction. The framework is similar for random log depth decision trees and random deterministic finite acceptors, but the embeddings are somewhat more complex.

3.1 Model

Let n be positive integer and $V = \{v_1, \ldots, v_n\}$. We adopt the model used by Sellie of random DNF formulas over the variables V. Each term is a conjunction of $c\log(n)$ literals created by selecting a random subset of $c\log(n)$ variables and negating each variable independently with probability $1/2$. The target random DNF formula is a disjunction of independently selected terms. Sellie gives a polynomial time algorithm to learn a random DNF with at most $n^c \log\log(n)$ terms under the uniform distribution on inputs [12].

For ease of description we first consider random monotone DNF formulas, in which the step of negating variables is omitted; the general case is described later. Given a positive integer ℓ, let $n = 2^{3\ell}$; then we have $\ell = \frac{1}{3}\log(n)$ and $2^\ell = n^{1/3}$. Let ϕ denote a random monotone DNF formula of $t = 2^{\ell-1}$ terms, where each term contains ℓ variables.

We first show that with probability $1 - o(1)$, no variable occurs more than once in ϕ, that is, ϕ is a read once formula. We can think of the process of choosing terms as successive random choices of a set of ℓ variables. If each set chosen avoids the variables chosen by the previous sets, then the formula is read once. Consider the last set chosen; it must avoid a collection of at most $s = (t-1)\ell$ variables, which it does with probability at least

$$\binom{(n-s)}{\ell}\binom{n}{\ell}^{-1} \geq 1 - \frac{s\ell}{n-s+1},$$

provided n is sufficiently large. Thus the probability of failure for the last term is $O((\log^2 n)/n^{2/3})$, and the probability that any of the $O(n^{1/3})$ terms fails is $O((\log^2 n)/n^{1/3})$, which bounds the probability that ϕ is not read once.

In what follows we assume that ϕ is read once. As an example, for $\ell = 3$ we have $n = 512$ and $t = 4$ and a possible value of ϕ is the following.

$$\phi = v_{14}v_{133}v_{170} \vee v_{22}v_{101}v_{337} \vee v_{55}v_{266}v_{413} \vee v_{10}v_{332}v_{507}$$

3.2 Embedding Parity

We consider the problem of learning a parity function with ℓ relevant variables from a total of $m = n/t$ variables $Y = \{y_1, y_2, \ldots, y_m\}$ with respect to the uniform distribution on assignments to Y. Because $\ell = \Theta(\log(n))$ and $m = \Theta(n^{2/3})$, such a function cannot be weakly learned using a polynomial number of statistical queries with respect to the uniform distribution [2].

Let L denote the set of literals over Y. A mapping from V to L is an *equi-grouping* if exactly $n/(2m) = t/2$ variables in V are mapped to each literal in L.

With respect to an arbitrary mapping f from V to L, an assignment a to the variables Y induces an assignment a_f to the variables V by $a_f(v_i) = a(f(v_i))$, that is, the value assigned to variable v_i is the value assigned by a to the literal $f(v_i) \in L$. More generally, a distribution D over assignments a to variables in Y induces a distribution D_f over assignments b to variables in V, where $D_f(b)$ is the sum of $D(a)$ such that $a_f = b$.

Fix an arbitrary parity function with ℓ relevant variables from Y. It can be represented by a DNF formula ψ of $t = 2^{\ell-1}$ terms, where each term has exactly one literal for each relevant variable, and the number of positive literals in each term is odd. For example, if the relevant variables are $\{y_{33}, y_{57}, y_{108}\}$, we have

$$\psi = y_{33}y_{57}y_{108} \vee y_{33}y'_{57}y'_{108} \vee y'_{33}y_{57}y'_{108} \vee y'_{33}y'_{57}y_{108}.$$

Note that the parity formula ψ and the random DNF ϕ each contain t terms of ℓ literals each. We describe a straightforward embedding of ψ into ϕ.

Choose a random bijection between the terms of ϕ and the terms of ψ, and for each term, a random bijection between the variables in the term of ϕ and the literals in the corresponding term of ψ. If v_i is a variable in ϕ, let $f(v_i)$ be the corresponding literal in ψ. Because the variables in ϕ are all distinct, f maps exactly $t/2$ distinct variables of ϕ to each literal of ψ.

Extend f arbitrarily to an equi-grouping by randomly dividing the unused variables in V into groups of size $t/2$ and mapping each group to a random distinct one of the unused literals in L. For every assignment a to the variables Y, $\psi(a) = \phi(a_f)$, so this construction embeds the parity function ψ into the random monotone DNF formula ϕ.

Continuing the example of ϕ and ψ, we could choose f to map v_{14} and v_{22} to y_{33}, v_{55} and v_{10} to y'_{33}, also v_{133} and v_{266} to y_{57}, v_{101} and v_{332} to y'_{57} and finally, v_{170} and v_{507} to y_{108}, v_{337} and v_{413} to y'_{108}, though different bijections are permitted. The unused 500 variables v_i are divided arbitrarily into groups of two and each group of two is mapped to one of the 250 unused literals y_j or y'_j. An assignment to the variables y_j induces an assignment to the variables v_i in which the variables in each group of two take on the value of the literal they are mapped to.

Note that the uniform distribution U on assignments to Y induces the distribution U_f on assignments to V, in which groups of $t/2$ variables are assigned the same value, the groups corresponding to a literal and its complement receive complementary values, and groups corresponding to different variables are independent.

3.3 Reduction

We now describe a reduction showing that a learning algorithm A that weakly learns a random monotone DNF formula ϕ (over n variables with t terms and ℓ variables per term) with respect to an arbitrary distribution D using a polynomial number of statistical queries to oracle $\mathrm{STAT}(\phi, D)$ could be used to weakly learn an arbitrary parity function ψ (over m variables with ℓ relevant variables) with respect to the uniform distribution using the same number of statistical queries to oracle $\mathrm{STAT}(\psi, U)$.

For the reduction, we randomly choose an equi-grouping g mapping V to L. We then run A with variables V, simulating access to a statistical query oracle $\mathrm{STAT}(\phi, U_g)$, where ϕ is a DNF formula that embeds ψ with respect to g. (That is, $\psi(a) = \phi(a_g)$ for all assignments a to the variables Y.)

A statistical query (χ, τ) made by A is transformed to (χ', τ), where

$$\chi'(a, b) = \chi(a_g, b).$$

That is, χ' transforms the assignment a to Y into the assignment a_g to V, keeps the label b, and applies χ. The query (χ', τ) is asked of the statistical query oracle $\mathrm{STAT}(\psi, U)$

for the parity problem, and the answer is returned as the answer to A's query (χ, τ). Even though we do not know a correct function ϕ embedding ψ, this transformation allows us to answer the statistical queries of A correctly for some such ϕ.

When A halts and returns a hypothesis h, the reduction halts and outputs a hypothesis $h'(a) = h(a_g)$. That is, the hypothesis h' transforms an assignment a to Y to the assignment a_g to V and applies h.

3.4 Correctness

Suppose that instead of choosing a random equi-grouping g, we could generate a random monotone DNF formula ϕ, rejecting if it is not read once, and otherwise choosing an embedding f of ψ into ϕ as described in the embedding subsection. The resulting distribution over equi-groupings f would still be uniform. Because ϕ is read once with probability $1 - o(1)$, this means that if A succeeds in weakly learning random monotone DNF formulas, the reduction gives a randomized algorithm to learn with probability $1 - o(1)$ an arbitrary parity function over m variables with ℓ relevant variables to the same level of accuracy using the same number of statistical queries with the same tolerances as A.

The extension to general (non-monotone) DNF formulas is straightforward; a general DNF formula with n variables, t terms and ℓ variables per term is read once with probability $1 - o(1)$, and embedding a parity function ψ into a general read once formula just requires mapping literals (rather than variables) over V to literals over Y and modifying the definition of the induced assignment a_g appropriately. Thus we conclude the following.

Theorem 1. *No algorithm can weakly learn random monotone (or general) DNF formulas with n variables, $n^{1/3}$ terms and $\log n$ variables per term with respect to an arbitrary distribution using a polynomial number of statistical queries.*

3.5 Extensions

This technique can also be used to show lower bounds for DNF formulas with more or fewer terms. If $(t\ell)^2$ is $o(n)$, then a random DNF with n variables, t terms and ℓ variables per term will be read once with probability $1 - o(1)$. If the number of terms is larger than $2^{\ell-1}$, it can be trimmed by choosing one literal per excess term to fix to the value 0 so that the term is eliminated under the constructed distribution. If the number of terms is smaller than $2^{\ell-1}$, we can choose a number of literals per term to fix to the value 1 so that the term effectively becomes smaller under the constructed distribution. For example, we could use the same logic to embed parity functions with $\Theta(\log\log(n))$ relevant variables (which are still not weakly learnable with a polynomial number of statistical queries) by using $\Theta(\log(n))$ terms.

To handle these cases, instead of just choosing an equi-grouping g from V to L, the reduction first randomly chooses an appropriate number of variables from V to set randomly and independently to 0 or 1, and then chooses an equi-grouping on the rest. The resulting induced distribution on assignments to V is constant on some variables, and behaves as before on the rest.

4 Random Decision Trees

The reduction for decision trees is slightly more complex than that for DNF formulas: with high probability the depth k top of a random decision tree of depth $3k$ will query all distinct variables, which can be used to embed an arbitrary depth k decision tree by choosing random paths of length $2k$ from the boundary nodes to leaf nodes.

4.1 Model

We consider binary decision trees over n variables $V = \{v_1, \ldots, v_n\}$ of uniform depth $k \geq 0$, where $n = 2^k$. The nodes of the tree are indexed by binary strings of length at most k, where the empty string is the root and the two children of x are $x0$ and $x1$. The string indexing a node gives the sequence of query answers to reach that node.

A *decision tree* of depth k is specified by two maps, $\alpha : \Sigma^{<k} \to V$ and $\beta : \Sigma^k \to \Sigma$, where α determines the variable queried at each internal node and β determines the binary label of each leaf. We require that the variables queried along any path in the tree be distinct. The value of a decision tree on an assignment a mapping V to $\{0, 1\}$ is determined by querying the values of the variables indicated by α to reach a leaf of the tree, whose β value is the desired output.

For a random decision tree of depth $k \geq 1$, the values of α are chosen uniformly at random and the values of β are chosen such that for all $x \in \Sigma^{k-1}$, one of $\beta(x0)$ and $\beta(x1)$ is 0 and the other is 1, where both outcomes have equal probability and the choices for different x's are independent. This is one of the models of random decision trees considered by Jackson and Servedio [6]; results for their other models should be similar.

4.2 Embedding

We show that an arbitrary decision tree of depth k can be embedded with probability $1 - o(1)$ in a random decision tree of depth $3k$ (or, with a bit more work, depth $(2+\epsilon)k$). Fix an arbitrary decision tree $T = (\alpha, \beta)$ of depth k over the variables $V = \{v_1, \ldots, v_n\}$. Let $T' = (\alpha', \beta')$ be a random decision tree of depth $k' = 3k$ with $n' = 2^{k'} = n^3$ variables $W = \{w_1, \ldots, w_{n'}\}$.

For each $x \in \Sigma^k$, let $y(x) \in \Sigma^{k'-1-k}$ be chosen uniformly at random. That is, for each internal node x at depth k in T', we choose a random extension $y(x)$ of its path that reaches the parent of a pair of leaves in T'. Let H' be the set of prefixes of strings $xy(x)$; these are the nodes along any of the n chosen paths.

Define T' to be *favorable* if the map α' is one-to-one on the domain H', that is, all the variables queried along the chosen paths in T' are distinct. T' is favorable with probability $1 - o(1)$ by a union bound, because the set H' has $O(n \log n)$ elements. We assume that T' is favorable in what follows. The embedding is illustrated in Figures 1 and 2.

We now define a map g from W to $V \cup \{0, 1\}$. If $g(w_i) = v_j$ we say w_i copies v_j, and if $g(w_i) = 0$ or $g(w_i) = 1$ we say that w_i is fixed to the corresponding constant value. For $x \in \Sigma^{<k}$, we define $g(\alpha'(x)) = \alpha(x)$, that is, the variable $\alpha'(x)$ copies the variable $\alpha(x)$. For $x \in \Sigma^k$ and z a proper prefix of $y(x)$, we define $g(\alpha'(xz)) = b$

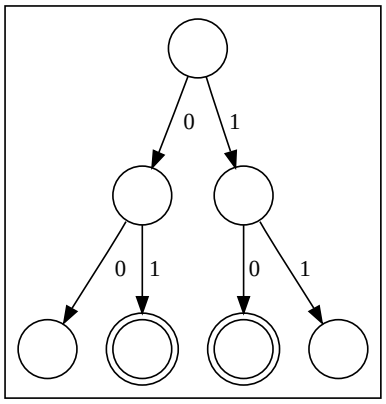

Fig. 1. The arbitrary decision tree T. Leaves with output 1 are indicated by double circles.

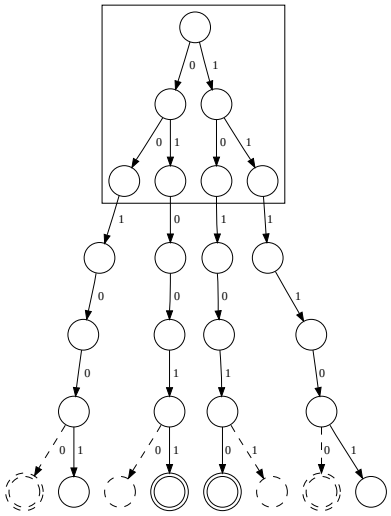

Fig. 2. The subgraph of the embedding tree T' containing a copy of T, the chosen random paths, and the correctly chosen outputs. T' is favorable if all the internal nodes of this subgraph have distinct variables.

where $b \in \Sigma$ is the unique value such that $xzb \in H'$, that is, the variable $\alpha'(xz)$ is fixed to the value necessary to follow the path chosen below x. For $x \in \Sigma^k$, we define $g(\alpha'(xy(x))) = b$ where $b \in \Sigma$ is the unique value such that $\beta'(xy(x)b) = \beta(x)$. That is, the variable $\alpha'(xy(x))$ is fixed to the value necessary to arrive at the leaf of T' with the same output as T at x. This is possible because β' takes on 0 and 1 in some order for the two children of $xy(x)$. At this point, each variable v_i in V has been assigned a distinct copy in W for each occurrence of v_i in T, so each variable in V has at most $n-1$ copies in W. We now choose unused variables from W without replacement to bring each variable in V up to exactly $n-1$ copies. Each remaining unused variable in W is fixed to 0 or 1 randomly and independently.

The mapping g induces a mapping from an assignment a to V to an assignment a_g to W by $a_g(w_j) = a(g(w_j))$ for all $j = 1, \ldots, n'$. For every assignment a to V, the output of T on a is the same as the output of T' on a_g. Thus, for any distribution D on assignments a, statistical queries to $\mathrm{STAT}(T', D_g)$ can be answered by making statistical queries to $\mathrm{STAT}(T, D)$.

4.3 Reduction

Suppose A is an algorithm that weakly learns random log depth decision trees using statistical queries with respect to an arbitrary distribution. Faced with the problem of learning an arbitrary depth k decision tree T over the $n = 2^k$ variables V with respect to

a distribution D using statistical queries to $\mathrm{STAT}(T, D)$, we let $k' = 3k$ and $n' = n^3$ and $W = \{w_1, \ldots, w_{n'}\}$. Define the set of *suitable reduction mappings* to be all g mapping g from W to $V \cup \{0, 1\}$ subject to the condition that $|g^{-1}(v_i)| = n - 1$ for $i = 1, \ldots, n$, that is, exactly $n-1$ variables in W are mapped to each variable in V and the remaining variables in W are mapped to 0 or 1. Choose a random suitable reduction mapping g.

We run A with the n' variables W and depth parameter k', simulating statistical queries to $\mathrm{STAT}(T', D_g)$ for some decision tree T' of depth k' over the variables W. When A makes a statistical query (χ, τ), we define

$$\chi'(a) = \chi(a_g),$$

make a statistical query to $\mathrm{STAT}(T, D)$ with (χ', τ) and return the answer to A. When A halts with output h, we output h', where $h'(a) = h(a_g)$.

4.4 Correctness

The distribution over suitable reduction mappings g would be uniform if we could first generate a random depth k' decision tree T' and a random choice of paths $y(x)$ from its nodes at depth k (rejecting if T' is not favorable) and then proceed to choose g to embed T in T' as described in the embedding subsection. Because the probability that we reject T' as not favorable is $o(1)$, with probability $1 - o(1)$, A weakly learns a tree T' embedding T, which means that the reduction must weakly learn T. Because log depth decision trees with n variables can express all parity functions over n variables with $\log n$ relevant variables, our reduction proves the following.

Theorem 2. *No algorithm can weakly learn random decision trees with n variables and depth $\log n$ with respect to an arbitrary distribution using a polynomial number of statistical queries.*

5 Random Deterministic Finite Acceptors

The embedding for random DFAs is somewhat more complex. By the results of Trakhtenbrot and Barzdin, we know that with high probability, if one state of a random DFA is reachable from another, it is reachable by a path of length $O(\log n)$. In order to represent parity, we embed two trees of $O(\log n)$ depth in the machine, but we must also find paths that return from the leaves of the trees to both their roots, in order to test the parities of a sequence of variables.

5.1 Model of Random DFAs

Let n be a positive integer and let Q be a finite set of n states with start state $q_0 \in Q$. We consider a standard model of random deterministic finite acceptors, in which the entries of the transition function $\delta : Q \times \Sigma \to Q$ and the set of accepting states $F \subseteq Q$ are chosen uniformly at random.

5.2 Representing Parity

We name variables for the parity problem using strings from a prefix-free set V constructed as follows. Let $\sigma : \{1, 2, \ldots\} \to \Sigma^*$ be the bijection defined by

$$\begin{aligned}\sigma(1) &= \epsilon \\ \sigma(2m) &= \sigma(m)0 \\ \sigma(2m+1) &= \sigma(m)1.\end{aligned}$$

Thus σ maps m to its binary representation after the first 1. Let $\ell = \lceil \log^2 n \rceil$ and let $V = \{\sigma(m) : m \in \{\ell, \ldots, 2\ell - 1\}\}$ be the set of *variables*. Note that $|V| = \ell = \Theta(\log^2 n)$.

Our goal is to make a random machine compute a parity function whose relevant variables are any subset $U \subseteq V$. To give some intuition, we first describe how to construct a finite state machine that accepts a sequence of variables when the sequence has an odd number of occurrences of variables from a given set U. As an example, assume that $\ell = 6$, which gives the set of variables

$$V = \{10, 11, 000, 001, 010, 011\}.$$

Assume that the set of relevant variables is the following.

$$U = \{10, 000, 001\}.$$

The representation we choose for an assignment $a : V \to \{0, 1\}$ is to list (in some order) the variables $v \in V$ such that $a(v) = 1$. For example, if a assigns 1 to the variables 10, 11, 000, and 010, one representation of a would be 1011000010.

Given this representation, we construct a finite automaton to accept the strings representing assignments with odd parity on the variables in U as follows. Let V' be the set of proper prefixes of strings from V. For each string $v' \in V'$ there are states $[q_0, v']$ and $[q_1, v']$. If for some $b \in \Sigma$ both $v' \in V'$ and $v'b \in V'$ then let

$$\delta([q_0, v'], b) = [q_0, v'b] \text{ and } \delta([q_1, v'], b) = [q_1, v'b].$$

Suppose for some $b \in \Sigma$, $v' \in V'$ and $vb \in V$. If $v'b \in U$, that is, $v'b$ is a relevant variable, we exchange even and odd as follows:

$$\delta([q_0, v'], b) = [q_1, \varepsilon] \text{ and } \delta([q_1, v'], b) = [q_0, \varepsilon].$$

If $v'b \notin U$, we do not exchange even and odd:

$$\delta([q_0, v'], b) = [q_0, \varepsilon] \text{ and } \delta([q_1, v'], b) = [q_1, \varepsilon].$$

The start state is $[q_0, \varepsilon]$ and the only accepting state is $[q_1, \varepsilon]$. After reading in a sequence of distinct variables, the machine is in state $[q_0, \varepsilon]$ if an even number of the variables read are from U, and in state $[q_1, \varepsilon]$ if an odd number of them are from U.

The machine constructed by this process for the example values of U and V is illustrated in Figure 3. Note that on the input string 1011000010 the machine reaches

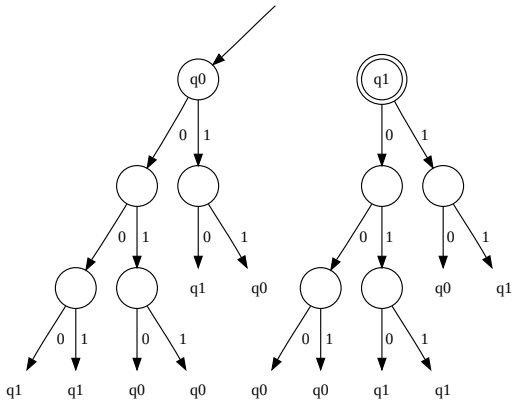

Fig. 3. One idea for a finite state machine to compute the parity of variables $U = \{10, 000, 001\}$ from $V = \{10, 11, 000, 001, 010, 011\}$. The in-degrees of q_0 and q_1 are in general too large.

state q_0 and rejects, which is correct because in this assignment exactly two of the variables in U (namely 10 and 000) are set to 1.

An attempt to embed this kind of machine for computing parity into a random finite state acceptor encounters the problem that the in-degrees of the states $[q_0, \varepsilon]$ and $[q_1, \varepsilon]$ are in general too large. We avoid this problem by changing the input representation to allow each variable to be followed by a string that prepares the machine to accept the next variable. Even though each variable may have a different string, the contents and assignments of these strings do not reveal any information about the relevant variables.

5.3 Embedding and Reduction

We show how to embed a parity computation on the variables V into a random finite state acceptor M with n states with at least constant probability of success. The process is illustrated in Figures 4, 5 and 6.

We first choose any state q_1 different from the start state q_0. We generate a random finite acceptor M with n states. We think of the structure of M as being revealed to us in stages. With probability $1/4$, we have $q_0 \notin F$ and $q_1 \in F$.

Let $V' = \{\sigma(m) : m \in \{1, \ldots, \ell - 1\}\}$. These are all the proper prefixes of the variables V. Because $|V'| = O(\log^2 n)$, with probability $1 - o(1)$, there are two non-overlapping trees rooted at q_0 and q_1, that is, the set of states $R = \{\delta(q, v') : q \in \{q_0, q_1\},\ v' \in V'\}$ has cardinality $2|V'|$. In this case, the values of $\delta(q, v)$ for $q \in \{q_0, q_1\}$ and $v \in V$ are all independent random choices.

For each variable $v \in V$ we would ideally like to find a string x_v such that x_v takes $\delta(q_0, v)$ and $\delta(q_1, v)$ back to the states q_0 and q_1 (in some order), that is,

$$\{\delta(q_0, vx_v), \delta(q_1, vx_v)\} = \{q_0, q_1\}.$$

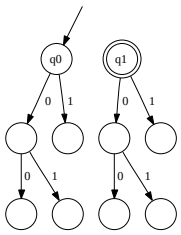

Fig. 4. The even state is q_0 and the odd state is q_1. With probability $1 - o(1)$, there are two non-overlapping trees with $\ell-1$ nodes rooted at q_0 and q_1. We don't yet commit to the outgoing arcs of the leaves.

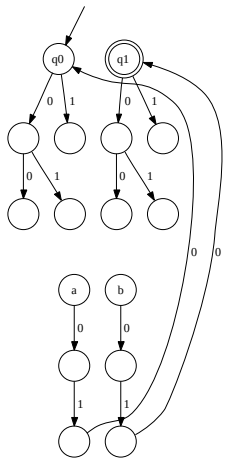

Fig. 5. With constant probability, a pair $(a, b) \in Q^2$ chosen uniformly at random can reach (q_0, q_1) via the same string while avoiding the trees.

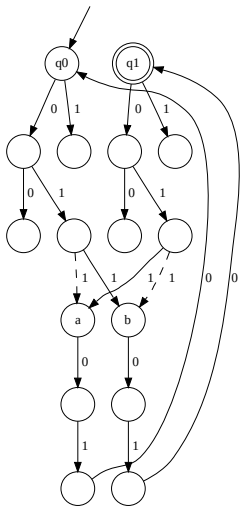

Fig. 6. Now we choose the outgoing arcs corresponding to variable 011. With constant probability, there is a path back to (q_0, q_1). The solid arcs signify a relevant variable; the dashed ones, an irrelevant variable. These cases are equally likely and independent of the string to prepare for another variable.

Depending on the order, the variable v is or is not relevant. Though we cannot achieve this *proper functioning* for all variables, we can do so for a constant fraction of the variables.

We first show that with constant probability, for two random states $(a, b) \in Q^2$, there is a short string x such that $\{\delta(a, x), \delta(b, x)\} = \{q_0, q_1\}$ and neither path touches the states in R. The latter stipulation is important because it enables a bijection that swaps the values of $\delta(q_0, v)$ and $\delta(q_1, v)$, which allows us to conclude that the return strings x_v don't give away the relevant variables.

The proof of existence of the short string is as follows. Let $(a, b) \in Q^2$ be chosen uniformly at random, assume n is even, and let $X = \{\sigma(m) : m \in \{n^2/4, \ldots, n^2/2 - 1\}\}$. We show that if $x \in X$ is chosen uniformly at random, then with probability $(2 - o(1))/n^2$, we have $\{\delta(a, x), \delta(b, x)\} = \{q_0, q_1\}$, that is, x is *good*. Also, if $y \in X$ is chosen independently, then the probability that both x and y are good is $(4-o(1))/n^4$.

By inclusion-exclusion, the probability that exactly one string in X is good is at least

$$\begin{aligned}|X|(2-o(1))/n^2 - 2\binom{|X|}{2}(4-o(1))/n^4 &= (1-o(1))/2 - 2\binom{n^2/4}{2}(4-o(1))/n^4 \\ &= (1-o(1))/2 - (1-o(1))/4 \\ &= (1-o(1))/4.\end{aligned}$$

The key observation is that the success event of x and the success event of y are very close to being independent Bernoulli trials with success probability $2/n^2$. The strings under consideration have length about $2\log n$. If just before the final letter of each string we have reached from a and b two different states whose outward transitions have not been committed in any way, the success probabilities *will* be independent and exactly $2/n^2$. Of course, something may go wrong before then: we may touch a state in R or have the paths loop or touch one another. With only $O(\log^2 n)$ states to avoid however, this event is probability $o(1)$.

Finally, we observe that with constant probability, there will be $\log^2 n/8$ variables that function properly. We can embed any parity function over these variables, which is enough to make $n^{\Theta(\log n)}$ parity functions, ensuring that a superpolynomial number of statistical queries are needed in expectation [2].

For the reduction, an assignment a to $\log^2 n/8$ variables is transformed to a string containing the list of strings vx_v for those variables v on which a takes the value 1. Note that these strings are all of length $O(\log^3 n)$. If desired, the strings could be padded by repeating the variables they contain an odd number of times.

Theorem 3. *No algorithm can weakly learn random deterministic finite acceptors with n states with respect to an arbitrary distribution on strings of length at most $\Theta(\log^3 n)$ using a polynomial number of statistical queries.*

6 Discussion

As described in Section 1, there are polynomial time algorithms to learn certain classes of random decision trees and random DNF formulas with respect to the uniform distribution, and these algorithms can be implemented with statistical queries. However, it is open whether random deterministic finite acceptors of n^c states can be learned in polynomial time with respect to the uniform distribution on strings of length n.

7 Acknowledgements

During parts of this work Aryeh Kontorovich was in the Department of Mathematics of the Weizmann Institute and Lev Reyzin was in the Department of Computer Science of Yale University.

References

1. Javed A. Aslam and Scott E. Decatur. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. *Information and Computation*, 141(2):85–118, 1998.
2. Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *STOC '94: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 253–262, New York, NY, USA, 1994. ACM.
3. Yoav Freund, Michael J. Kearns, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. *Information and Computation*, 138(1):23–48, 1997.
4. E Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.
5. Jeffrey Jackson, Homin Lee, Rocco Servedio, and Andrew Wan. Learning random monotone DNF. In *Lecture Notes in Computer Science 5171: Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, pages 483–497, Berlin / Heidelberg, 2008. Springer.
6. Jeffrey C. Jackson and Rocco A. Servedio. Learning random log-depth decision trees under uniform distribution. *SIAM J. Comput.*, 34(5):1107–1128, 2005.
7. M. Kearns and L. G. Valiant. Crytographic limitations on learning boolean formulae and finite automata. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 433–444, New York, NY, USA, 1989. ACM.
8. Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.
9. Kevin J. Lang. Random DFA's can be approximately learned from sparse uniform examples. In *COLT '92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 45–52, New York, NY, USA, 1992. ACM.
10. Leonard Pitt and Manfred K. Warmuth. Prediction-preserving reducibility. *J. Comput. Syst. Sci.*, 41(3):430–467, 1990.
11. Linda Sellie. Learning random monotone DNF under the uniform distribution. In *COLT*, pages 181–192, 2008.
12. Linda Sellie. Exact learning of random DNF over the uniform distribution. In *STOC*, pages 45–54, 2009.
13. B. A. Trakhtenbrot and Ya. M. Barzdin. *Finite Automata: Behavior and Synthesis*. North Holland, Amsterdam, 1973.