Theoretical Computer Science 473 (2013) 46–60

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Learning probabilistic automata: A study in state distinguishability[☆]

Borja Balle [*], Jorge Castro, Ricard Gavaldà

Departament de Llenguatges i Sistemes Informàtics, LARCA Research Group, Universitat Politècnica de Catalunya, Spain

ARTICLE INFO

Keywords:
Distribution learning
PAC learning
Probabilistic automata
Statistical queries

ABSTRACT

Known algorithms for learning PDFA can only be shown to run in time polynomial in the so-called distinguishability μ of the target machine, besides the number of states and the usual accuracy and confidence parameters. We show that the dependence on μ is necessary in the worst case for every algorithm whose structure resembles existing ones. As a technical tool, a new variant of Statistical Queries termed L_∞-queries is defined. We show how to simulate L_∞-queries using classical Statistical Queries and show that known PAC algorithms for learning PDFA are in fact statistical query algorithms. Our results include a lower bound: every algorithm to learn PDFA with queries using a reasonable tolerance must make $\Omega(1/\mu^{1-c})$ queries for every $c > 0$. Finally, an adaptive algorithm that PAC-learns w.r.t. another measure of complexity is described. This yields better efficiency in many cases, while retaining the same inevitable worst-case behavior. Our algorithm requires fewer input parameters than previously existing ones, and has a better sample bound.

1. Introduction

Probabilistic finite automata (PFA) are important as modeling formalisms as well as computation models. They are closely related to Hidden Markov Models (HMMs) in their ability to represent distributions on finite alphabets and also to POMDPs; see e.g. [3–5] for background.

One of the main associated problems is that of approximating the distribution generated by an unknown probabilistic automaton from examples. The problem is relatively simple if the structure of the automaton is somehow known and only transition probabilities have to be estimated, and much harder and poorly-solved in practice if the transition graph is unknown. Probabilistic Deterministic Finite Automata (PDFA) — in which the underlying automaton is deterministic but transitions still have probabilities — have often been considered as a restriction worth studying, even though they cannot generate all distributions generated by PFA [3].

The grammatical inference community has produced a substantial number of methods for learning (distributions generated by) PFA or PDFA, most of them using so-called "state split–merge" or "evidence-driven" strategies; see the references in [6,4,5,7]. Many of these methods are only proved valid empirically, but some have proofs of learning in the limit.

The problem has also been intensely studied in variants of the PAC model adapted to distribution learning. Abe and Warmuth showed in [8] that hardness is not information-theoretic: one can learn (distributions generated by) PFA with samples of size polynomial in alphabet size, number of states in the target machine, and inverses of the accuracy and confidence parameters (ϵ and δ); but also that the problem is computationally intractable for large alphabet sizes, unless RP = NP. Kearns et al. [9] showed that learning PDFA even over 2-letter alphabets is computationally as hard as solving the *noisy parity learning problem*, of interest in coding theory and for which only super-polynomial time algorithms are known.

[☆] Preliminary versions of these results appeared in the conference papers Castro and Gavaldà (2008) [1] and Balle et al. (2010) [2].

[*] Corresponding author.
E-mail addresses: bballe@lsi.upc.edu (B. Balle), castro@lsi.upc.edu (J. Castro), gavalda@lsi.upc.edu (R. Gavaldà).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.10.009

It was later observed that polynomial-time learnability of PDFA is feasible if one allows polynomiality not only in the number of states but also in other measures of the target automaton complexity. Specifically, Ron et al. [10] showed that acyclic PDFA can be learned w.r.t. the Kullback–Leibler divergence in time polynomial in alphabet size, $1/\epsilon$, $1/\delta$, number of target states, and $1/\mu'$, where μ' denotes the *prefix-distinguishability* of the target automaton, to be defined in Section 2. Clark and Thollard extended the result to general PDFA by using a simplified notion of *distinguishability* μ and considering also as a parameter the expected length of the strings generated by the automata [6]. Their algorithm, which we denote by CT, is a state split–merge method, and was in turn extended or refined in subsequent work [11–13,1]; see also [14] for a simpler version of CT called ALISA. Similar algorithms without strong theoretical guarantees can be found in [15–17]. Furthermore, in [18] a PAC algorithm for learning PFA was given, similar in spirit to [7], whose running time is polynomial in the inverse of a condition parameter, intuitively an analog of μ for PFA. More recently, a Bayesian approach to PDFA learning has been suggested by Pfau et al. [19].

Here we consider the dependence on the distinguishability parameter μ of known algorithms. We know that the sample complexity and running time of CT and related algorithms are polynomially bounded on $1/\mu$ (as well as other parameters), but it is conceivable that one could also prove a polynomial bound in another parameter, much smaller but yet unidentified. In this paper we pursue this line of research and provide two results.

Our first result shows that any such hypothetical parameter must necessarily coincide with μ when restricted to a worst-case subclass of PDFA. Inspired by the noisy parity problem, we show that this worst-case subclass contains PDFA defined in terms of parity concepts. These results hold, at least, for a large class of learning algorithms including all algorithms that proceed by applying statistical tests to subsets of the sample to distinguish distributions generated at different states of the target automaton. We formalize this notion by defining a variant of Kearns' Statistical Queries [20], called L_∞-queries. We observe that known algorithms for learning PDFA, such as CT, ALISA and our variant [1], can be rewritten accessing the target distribution only through L_∞-queries (to infer the structure) plus standard Statistical Queries (to approximate transition probabilities). We then show that any algorithm that learns the class of PDFA with a given distinguishability μ from L_∞-queries and Statistical Queries with reasonably bounded tolerance will require at least $\Omega(1/\mu^{1-c})$ queries for every $c > 0$. Our result thus indicates that, if PDFA learning algorithms of complexity substantially smaller than $1/\mu$ do exist, they must use their input sample quite differently from known algorithms. We also show how to simulate L_∞-queries using standard Statistical Queries and deduce that CT and related algorithms can be fully implemented in the statistical query model; this settles in the negative a conjecture we posed in [2].

Our second result in this study in distinguishability is an adaptive algorithm AdaCT for learning general PDFA that uses the prefix-distinguishability μ' defined by Ron et al. for the case of acyclic PDFA. Using techniques similar to those in [6,1], we provide a detailed analysis of this new algorithm. Its several improvements over previously existing variants of CT suggest that a careful implementation of AdaCT might be used in real-world problems, even though it can be rigorously shown to satisfy the PAC learning criteria. In fact, preliminary experimental results presented in [1] suggest that, in many cases, the new algorithm obtains similar results to those provided by known heuristic methods, with sample sizes much smaller than required by the bounds. This is due to several singular features of AdaCT. First, even though μ and μ' coincide inside the worst-case class defined in terms of parities, in general $1/\mu'$ can be much smaller than $1/\mu$, yielding a much more efficient learning procedure in many cases. In addition, the upper bounds for AdaCT are asymptotically better than the corresponding bounds for CT. Furthermore, our algorithm requires fewer input parameters than any previous PAC algorithm; this reduces the amount of guess-work and tedious cross-validations needed for selecting the correct parameters. Another interesting feature of AdaCT is its adaptive nature: receiving a sample and using only n, Σ and δ as input parameters, it is able to reuse the sample more thoroughly and extract more information from it than CT. Finally, unlike CT, our algorithm does not require a minimum sample size; it produces a hypothesis from a sample received up-front, which is guaranteed to be correct when the sample is large enough.

The rest of the paper is organized as follows. Section 2 is devoted to notation and preliminaries. Our learning model in terms of L_∞-queries is discussed in Section 3, and a lower bound on the complexity of algorithms learning PDFA in this model is proved in Section 4. In Section 5 the algorithm AdaCT is described and analyzed. We conclude with a discussion of our results in Section 6.

2. Preliminaries

2.1. Notation and definitions

Let Σ be a finite alphabet, and for $n > 0$ write $[n] = \{1, \ldots, n\}$. Given $h = (h_1, \ldots, h_n) \in \Sigma^n$ and $1 \leq i < j \leq n$ let $h_{i:j} = (h_i, \ldots, h_j)$. For any symbol $\sigma \in \Sigma$, we write $\bar{\sigma}$ to denote a string containing only the symbol σ, and whose length can be deduced from the context. We say that $\hat{p}$ is an α-approximation of p if $|p - \hat{p}| \leq \alpha$.

Several measures of divergence between distributions are considered. Let D_1 and D_2 be probability distributions over a discrete set X. The Kullback–Leibler (KL) divergence is defined as

$$\mathrm{KL}(D_1 \| D_2) = \sum_{x \in X} D_1(x) \log \frac{D_1(x)}{D_2(x)}, \tag{1}$$

where the logarithm is taken to base 2; KL is sometimes called relative entropy. The supremum distance is $\mathrm{L}_\infty(D_1, D_2) = \max_{x\in X} |D_1(x) - D_2(x)|$, and the total variation distance is $\mathrm{L}_1(D_1, D_2) = \sum_{x\in X} |D_1(x) - D_2(x)|$. Finally, we define the supremum distance on prefixes as $\mathrm{prefL}_\infty(D_1, D_2) = \max_{x\in\Sigma^*} |D_1(x\Sigma^*) - D_2(x\Sigma^*)|$, where $D(x\Sigma^*)$ is the probability under D of having x as a prefix.

Let D be a distribution over Σ^*. We use $\mathrm{supp}(D)$ to denote the support of D; that is, the set of strings which have positive probability under D. The expected length of D is $L_D = \sum_{x\in\Sigma^*} |x|D(x)$. If $A \subseteq \Sigma^*$ is prefix-free, we denote by D^A the conditional distribution under D of having a prefix in A. So, for every $y \in \Sigma^*$, we have

$$D^A(y) = \frac{D(Ay)}{D(A\Sigma^*)} = \frac{\sum_{x\in A} D(xy)}{\sum_{x\in A} D(x\Sigma^*)}. \tag{2}$$

Given a multiset S of strings from Σ^* we denote by $S(x)$ the multiplicity of x in S, write $|S| = \sum_{x\in\Sigma^*} S(x)$ and for every $\sigma \in \Sigma$ define $S(\sigma) = \sum_{x\in\Sigma^*} S(\sigma x)$. To resolve the ambiguity of this notation on strings of length 1, we will always use Greek letters to mean elements of Σ, and Latin letters for strings. We also denote by $S(\xi)$ the multiplicity of the empty word, $S(\lambda)$. To each multiset S corresponds an empirical distribution $\widehat{S}$ defined in the usual way: $\hat{S}(x) = S(x)/|S|$. Finally, $\mathrm{prefixes}(S)$ denotes the multiset of prefixes of strings in S.

Now we fix $\Sigma = \{0, 1\}$ and equip it with addition and multiplication via the identification $\Sigma \simeq \mathbb{Z}/(2\mathbb{Z})$. Given $h \in \Sigma^n$, the parity function $P_h : \Sigma^n \to \Sigma$ is defined as $P_h(u) = \bigoplus_{i\in[n]} h_i u_i$, where $\oplus$ denotes addition modulo 2. Each h defines a distribution D_h over Σ^{n+1} whose support depends of P_h as follows: for $u \in \Sigma^n$ and $c \in \Sigma$ let

$$D_h(uc) = \begin{cases} 2^{-n} & \text{if } P_h(u) = c, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

2.2. Probabilistic automata

A PDFA T is a tuple $\langle Q, \Sigma, \tau, \gamma, \xi, q_0\rangle$ where Q is a finite set of states, Σ is an arbitrary finite alphabet, $\tau : Q \times \Sigma \longrightarrow Q$ is the transition function, $\gamma : Q \times (\Sigma \cup \{\xi\}) \longrightarrow [0, 1]$ defines the probability of emitting each symbol from each state ($\gamma(q, \sigma) = 0$ when $\sigma \in \Sigma$ and $\tau(q, \sigma)$ is not defined), ξ is a special symbol not in Σ reserved to mark the end of a string, and $q_0 \in Q$ is the initial state. It is required that $\sum_{\sigma\in\Sigma\cup\{\xi\}} \gamma(q, \sigma) = 1$ for every state q. The transition function τ is extended to $Q \times \Sigma^*$ in the usual way. Also, the probability of generating a given string $x\xi$ from state q can be calculated recursively as follows: if x is the empty word λ the probability is $\gamma(q, \xi)$, otherwise x is a string $\sigma_0\sigma_1 \dots \sigma_k$ with $k \geq 0$ and $\gamma(q, \sigma_0\sigma_1 \dots \sigma_k\xi) = \gamma(q, \sigma_0)\gamma(\tau(q, \sigma_0), \sigma_1 \dots \sigma_k\xi)$. Assuming every state of T has non-zero probability of generating some string, one can define for each state q a probability distribution D_q on Σ^*: for each x, probability $D_q(x)$ is $\gamma(q, x\xi)$. The one corresponding to the initial state D_{q_0} is called the distribution defined by T. Most commonly, we will identify a PDFA and the distribution it defines.

For a given PDFA T we denote by L_T the maximum of the expected lengths of all distributions D_q. We will drop the subscript T when the target is obvious from the context. The complexity of learning a particular PDFA is measured by the expected length L and a quantity measuring how *different* the states of the automata are; this measure is usually called *distinguishability*. Different notions of distinguishability can be associated with different metrics between probability distributions. The following ones have been found useful for learning purposes because provably correct statistical tests based on those notions can be easily implemented [1,6,10].

Definition 1. We say distributions D_1 and D_2 are *μ-distinguishable* when $\mu \leq \mathrm{L}_\infty(D_1, D_2)$. A PDFA T is μ-distinguishable when for each pair of states q_1 and q_2 their corresponding distributions D_{q_1} and D_{q_2} are μ-distinguishable. The *distinguishability* of a PDFA is defined as the supremum over all μ for which the PDFA is μ-distinguishable.

Definition 2. We say distributions D_1 and D_2 are *μ'-prefix-distinguishable* when

$$\mu' \leq \max\{\mathrm{L}_\infty(D_1, D_2), \mathrm{prefL}_\infty(D_1, D_2)\}.$$

A PDFA T is μ'-prefix-distinguishable when for each pair of states q_1 and q_2 their corresponding distributions D_{q_1} and D_{q_2} are μ'-prefix-distinguishable. The *prefix-distinguishability* of a PDFA is defined as the supremum over all μ' for which the PDFA is μ'-prefix-distinguishable.

We typically use μ' and μ to denote, respectively, the prefix-distinguishability and the distinguishability of a PDFA. Note that in general $\mu' \geq \mu$. It is easy to construct pairs of PDFA for whom μ' and μ are essentially the same, and also pairs of PDFA such that μ' is exponential in μ — see the dicussion in Section 6 for examples and the impact this has on our results.

For any n and μ we will use $\mathcal{PDFA}_n$ to denote the class of all PDFA with at most n states, and $\mathcal{PDFA}_{n,\mu}$ to denote the subclass of $\mathcal{PDFA}_n$ containing all PDFA with distinguishability at least μ. Abusing our notations, we will use $\mathcal{PDFA}_{n,\mu'}$ to denote the subclass of $\mathcal{PDFA}_n$ containing all PDFA with prefix-distinguishability at least μ'.

It is a simple observation that for each $h \in \{0, 1\}^n$, the distribution D_h defined in the previous section can be modelled by a PDFA with at most $2n + 2$ states. Furthermore, for all h the corresponding PDFA has distinguishability and prefix-distinguishability $2/2^n$.

2.3. Statistical queries

Now we introduce a variant of classical statistical queries [20] for the case when the target of the learning process is a distribution instead of a concept. To the best of our knowledge, this model has not been considered in the literature before.

Recall that in the usual statistical query model, an algorithm for learning concepts over some set X can ask queries of the form (χ, α) where $\chi : X \times \{0, 1\} \to \{0, 1\}$ is a predicate and $0 < \alpha < 1$ is some tolerance. If D is a distribution over X and $f : X \to \{0, 1\}$ is a concept, a query $\mathrm{SQ}_f^D(\chi, \alpha)$ to the oracle answers with an α-approximation $\hat{p}_\chi$ of $p_\chi = \mathbb{P}_{x\sim D}[\chi(x, f(x)) = 1]$. Kearns interprets this oracle as a proxy to the usual PAC example oracle EX_f^D abstracting the fact that learners usually use examples only to obtain statistical properties about f under D. The obvious adaptation of statistical queries for learning distributions over X is to do the same, but without labels.

Let D be a distribution over some set X. A *statistical query for D* is a pair (χ, α) where $\chi : X \to \{0, 1\}$ is (an encoding of) a predicate and $0 < \alpha < 1$ is some tolerance. The query $\mathrm{SQ}^D(\chi, \alpha)$ returns an α-approximation of $p_\chi = \mathbb{P}_{x\sim D}[\chi(x) = 1]$. Since χ is the characteristic function of some subset of X, learners can ask the oracle an approximation to the probability of any event; we will usually identify χ and its characteristic set. Furthermore, we require that χ can be evaluated efficiently. Note that this new statistical query can easily be simulated using examples drawn i.i.d. from D.

2.4. Learning models

Now we introduce two models — one with examples and another with queries — for learning distributions w.r.t. an arbitrary measure of divergence.

Let $\mathcal{D}$ be a class of distributions over some fixed set X, and let d denote an arbitrary notion of divergence between two probability distributions. Assume $\mathcal{D}$ is equipped with some measure of complexity assigning a positive number $|D|$ to any $D \in \mathcal{D}$.

We say that an algorithm L *learns with examples* a class of distributions $\mathcal{D}$ w.r.t. d using $S(\cdot)$ examples and time $T(\cdot)$ if, for all $0 < \epsilon, \delta < 1$ and $D \in \mathcal{D}$, with probability at least $1 - \delta$, the algorithm uses $S(1/\epsilon, 1/\delta, |D|)$ examples drawn i.i.d. from D and after $T(1/\epsilon, 1/\delta, |D|)$ steps outputs a hypothesis $\widehat{D}$ such that $d(D, \widehat{D}) \leq \epsilon$. The probability is over the sample used by L and any internal randomization.

We say that a statistical query algorithm L *learns with queries* a class of distributions $\mathcal{D}$ w.r.t. d using $R(\cdot)$-bounded $S(\cdot)$ statistical queries and time $T(\cdot)$ if, for all $0 < \epsilon < 1$ and $D \in \mathcal{D}$, the algorithm makes $S(1/\epsilon, |D|)$ queries to SQ^D, all of them with tolerance at least $R(\epsilon, 1/|D|)$, and after $T(1/\epsilon, |D|)$ steps outputs a hypothesis $\widehat{D}$ such that $d(D, \widehat{D}) \leq \epsilon$.

As usual, we say that the algorithms considered above are efficient if the functions $R(\cdot)$, $S(\cdot)$ and $T(\cdot)$ are polynomial on each of their parameters.

Note that we do not require that algorithms are proper learners; that is, return some hypothesis from $\mathcal{D}$. Furthermore, we do not make any distinction between algorithms that output a hypothesis that is a generator or an evaluator, as is done, for example, in [9]. Mainly, this is because our lower bounds are representation independent, and our learning algorithms output PDFA, which can be used either as evaluators or as generators.

3. Learning PDFA with queries

This section introduces a new kind of statistical query, called L_∞-query, that appears naturally in many state-merging algorithms for learning PDFA. We show how these queries can be simulated using standard statistical queries. Furthermore, we prove a lower bound on the trade-off between the accuracy and number of such queries that an algorithm learning the class of all PDFA must make.

3.1. L_∞-queries

In this section we present a new kind of query, the L_∞-query. Roughly speaking, these queries can be used whenever the learning task is to approximate a probability distribution whose support is contained in a free monoid Σ^*. This query is an abstraction of a pattern of access to distributions appearing in algorithms that learn (distributions generated by) PDFA [16,10,6,12,13,1]. At some point, all algorithms described in these papers use samples from suffix distributions to test for state-distinctness w.r.t. the supremum distance.

The oracle DIFF_∞^D answers queries of the form (A, B, α, β), where $A, B \subseteq \Sigma^*$ are (encodings of) disjoint and prefix-free sets, and $0 < \alpha, \beta < 1$ are, respectively, *tolerance* and *thershold* parameters of the query. Let $\mu = \mathrm{L}_\infty(D^A, D^B)$ denote the supremum distance between distributions D^A and D^B. The oracle must answer a query $\mathrm{DIFF}_\infty^D(A, B, \alpha, \beta)$ according to the following rules:

(i) If either $D(A\Sigma^*) < \beta$ or $D(B\Sigma^*) < \beta$, it answers `?`.
(ii) If both $D(A\Sigma^*) > 3\beta$ and $D(B\Sigma^*) > 3\beta$, it answers with some α-approximation $\hat{\mu}$ of μ.
(iii) Otherwise, the oracle may either answer `?` or give an α-approximation of μ, arbitrarily.

To be precise, the algorithm asking a query will provide A and B in the form of oracles deciding the membership problems for $A\Sigma^*$ and $B\Sigma^*$.

We say that an L_∞-query algorithm L *learns with* L_∞*-queries* a class of distributions $\mathcal{D}$ w.r.t. d using $(Q(\cdot), R(\cdot))$-bounded $S(\cdot)$ L_∞-queries and time $T(\cdot)$ if, for all $0 < \epsilon < 1$ and $D \in \mathcal{D}$, the algorithm makes $S(1/\epsilon, |D|)$ queries to SQ^D, all of them with tolerance at least $Q(\epsilon, 1/|D|)$ and threshold at least $R(\epsilon, 1/|D|)$, and after $T(1/\epsilon, |D|)$ steps outputs a hypothesis $\widehat{D}$ such that $d(D, \widehat{D}) \leq \epsilon$.

Algorithms for learning distributions that can make Statistical Queries *and* L_∞-queries will also be considered. These algorithms are given access to oracles SQ^D and DIFF_∞^D.

Similarly to what happens with standard Statistical Queries, any L_∞-query can be easily simulated with high probability using examples as shown by the following result, which can be proved by a standard argument with Chernoff bounds. Accordingly, we will consider these algorithms to be efficient when the functions $Q(\cdot)$, $R(\cdot)$, $S(\cdot)$ and $T(\cdot)$ are polynomial on each of their parameters.

Proposition 1. *For any distribution D over Σ^*, an L_∞-query $\mathrm{DIFF}_\infty^D(A, B, \alpha, \beta)$ can be simulated with error probability smaller than δ using $\widetilde{O}(1/(\alpha^2\beta^2)\log(1/\delta))$ examples drawn i.i.d. from D.*

Remark 1 (*The Role of β*)**.** An algorithm asking an L_∞-query does not know a priori the probability under D of having a prefix in A. It could happen that the region $A\Sigma^*$ had very low probability, and this might indicate that a good approximation of D in this region is not necessary in order to obtain a good estimate of D. Furthermore, getting this approximation would require a large number of examples. Thus, β allows a query to fail — i.e. return '?' — when at least one of the regions being compared has low probability.

Remark 2 (*Representation of A and B*)**.** From now on we will concentrate on the information-theoretic aspects of L_∞-queries. Our focus will be on the number of queries needed to learn certain classes of distributions, as well as the required tolerances and thresholds. We are not concerned with how A and B are encoded or how membership to them is tested from the code: the representation could be a finite automaton, a Turing machine, a hash table, a logical formula, etc. The only requirement we impose is that membership testing can be done efficiently.

It is a simple observation that the algorithm CT given by Clark and Thollard in [6] for learning PDFA can be rewritten using L_∞-queries for learning the relevant transitions in the target and Statistical Queries for approximating its transition and stopping probabilities. By inspecting their proofs one can determine the required tolerances and thresholds. More precisely, we have the following.

Theorem 2 (*Clark and Thollard [6]*)**.** *There exists an algorithm $\mathtt{CT}_{\mathrm{L}_\infty,\mathtt{SQ}}$ that, given access to oracles SQ^T and DIFF_∞^T for some $T \in \mathcal{PDFA}_{n,\mu}$ with expected length L, on input $\mathtt{CT}_{\mathrm{L}_\infty,\mathtt{SQ}}(n, \Sigma, \mu, L, \epsilon)$ makes $O(n^2|\Sigma|)$ (α, β)-bounded L_∞-queries and $O(n|\Sigma|)$ γ-bounded statistical queries, and returns a PDFA H such that $\mathrm{KL}(T\|H) \leq \epsilon$, where $\alpha = \Omega(\mu)$ and $\beta, \gamma = poly(\epsilon, 1/n, 1/|\Sigma|, 1/L)$.*

Interestingly, the dependence on μ only appears in the part of the algorithm that recovers the relevant transition structure in the target.

3.2. Relation with statistical queries

In this section we show that L_∞-queries can be simulated using Statistical Queries. The simulation has some interesting consequences that are discussed between the end of this section and the beginning of the next one. We begin by stating a simple lemma which is commonly used when approximating conditional probabilities using statistical queries (see e.g. [21]).

Lemma 3. *Let $0 \leq a, b, \gamma \leq 1$. If $|\hat{a} - a|, |\hat{b} - b| \leq b\gamma/3$, then $|\hat{a}/\hat{b} - a/b| \leq \gamma$.*

Note that, in order to obtain an absolute approximation of a quotient, a *relative* approximation of the denominator is needed. Although the lemma will be used when estimating fractions where the denominator is unknown, in general a lower bound on b is enough to obtain the desired approximation.

To bound the running time of the simulation we will use the following simple isoperimetric inequality, see e.g. [22]. Think of Σ^* as a tree and let $\Gamma \subseteq \Sigma^*$ be a finite connected subset. We denote by $\partial\Gamma$ the *boundary* of Γ; that is, the set of all vertices in $\Sigma^* \setminus \Gamma$ that are adjacent to some vertex in Γ.

Lemma 4. *Any finite connected $\Gamma \subseteq \Sigma^*$ satisfies $|\Gamma| \leq |\partial\Gamma|/(|\Sigma| - 1)$.*

Let D be a distribution over Σ^* and $A, B \subseteq \Sigma^*$ disjoint and prefix-free. Furthermore, let $0 < \alpha, \beta < 1$. The following theorem shows how to find an approximation of $\mathrm{L}_\infty(D^A, D^B)$ using statistical queries. Its strategy is similar to that of the algorithm for learning decision trees with membership queries by Kushilevitz and Mansour [23].

Theorem 5. *An L_∞-query $\mathrm{DIFF}_\infty^D(A, B, \alpha, \beta)$ can be simulated using $O(|\Sigma|L/\alpha^2)$ calls to SQ^D with tolerance $\Omega(\alpha\beta)$, where L is the maximum of the expected lengths of D^A and D^B.*

Proof. The simulation begins by checking whether $\mathrm{SQ}^D(A\Sigma^*, \beta) < 2\beta$ or $\mathrm{SQ}^D(B\Sigma^*, \beta) < 2\beta$. If this is the case, then either having a prefix in A or in B has probability at most 3β under D and the simulation returns '?'. Otherwise, the simulation will compute an approximation $\hat{\mu}$ of $\mu = \mathrm{L}_\infty(D^A, D^B)$ such that $|\hat{\mu} - \mu| \le \alpha$.

Note that if $\mu \le \alpha$, then $\hat{\mu} = 0$ is a safe approximation. In contrast, if $\mu > \alpha$, then it must be the case that $\mu = |D^A(x) - D^B(x)|$ for some x such that either $D^A(x)$ or $D^B(x)$ is larger than α. Therefore, it is enough to find a set of strings T that contains every x such that either $D^A(x) > \alpha$ or $D^B(x) > \alpha$ and return the maximum of $|D^A(x) - D^B(x)|$ over T. Now we show how to implement this strategy using statistical queries.

Note that hereafter we can assume $D(A\Sigma^*) \ge \beta$ and $D(B\Sigma^*) \ge \beta$, and for any string x let us write

$$f(x) = \frac{\mathrm{SQ}^D(Ax\Sigma^*, \alpha\beta/9)}{\mathrm{SQ}^D(A\Sigma^*, \alpha\beta/9)} \quad \text{and} \quad g(x) = \frac{\mathrm{SQ}^D(Ax, \alpha\beta/9)}{\mathrm{SQ}^D(A\Sigma^*, \alpha\beta/9)}. \tag{4}$$

By Lemma 3 we know that $f(x)$ and $g(x)$ are, respectively, $\alpha/3$-approximations of $D^A(x\Sigma^*)$ and $D^A(x)$. In the first place, our simulation builds a set T^A as follows. Starting with an empty T^A and beginning with the root λ, recursively explore the tree Σ^*, and at each node x do: stop exploring if $f(x) \le 2\alpha/3$; otherwise, add x to the current T^A if $g(x) > 2\alpha/3$, and recursively explore the nodes $x\sigma$ for each $\sigma \in \Sigma$. Now we prove the following two claims about the output of this procedure: (i) for all $x \in T^A$ we have $D^A(x) > \alpha/3$, and (ii) if $D^A(x) > \alpha$, then $x \in T^A$. We will also bound the size of T^A. For (i) observe that the inclusion condition $g(x) > 2\alpha/3$ implies $D^A(x) > \alpha/3$; this implies $|T^A| < 3/\alpha$. On the other hand, let x be a string with $D^A(x) > \alpha$. For any prefix y of x we have $D^A(y\Sigma^*) > \alpha$ and $f(y) > 2\alpha/3$. Thus, our stopping condition guarantees that for each $\sigma \in \Sigma$ the nodes $y\sigma$ will be explored. In addition, $D^A(x) > \alpha$ implies $g(x) > 2\alpha/3$. Hence, when x is reached it will be added to T^A. This is claim (ii). Another set T^B for D^B with similar guarantees is built in the same way.

The next step is to show how to use $T = T^A \cup T^B$ to answer the query. Let us write

$$h(x) = \left| \frac{\mathrm{SQ}^D(Ax, \alpha\beta/6)}{\mathrm{SQ}^D(A\Sigma^*, \alpha\beta/6)} - \frac{\mathrm{SQ}^D(Bx, \alpha\beta/6)}{\mathrm{SQ}^D(B\Sigma^*, \alpha\beta/6)} \right|, \tag{5}$$

and note that, by Lemma 3, $h(x)$ is an α-approximation of $|D^A(x) - D^B(x)|$. The simulation returns $\hat{\mu} = \max_{x\in T} h(x)$, where $\hat{\mu} = 0$ if T is empty. This can be computed using at most $O(1/\alpha)$ queries with tolerance at least $\Omega(\alpha\beta)$.

Finally, we bound the number of statistical queries used by the simulation, and their tolerances. Let R^A be the set of all vertices of Σ^* explored when constructing T^A; note that R^A is connected. The number of statistical queries used to build T^A is obviously $O(|R^A|)$. Now let $\widehat{S^A}$ denote the set of *stopping* vertices of the process that builds T^A:

$$\widehat{S^A} = \{x\sigma \mid f(x) > 2\alpha/3 \wedge f(x\sigma) \le 2\alpha/3\}. \tag{6}$$

It is clear that $\widehat{S^A} \subseteq R^A$. Furthermore, any vertex in ∂R^A is adjacent to some vertex in $\widehat{S^A}$, and each vertex in $\widehat{S^A}$ is adjacent to at most $|\Sigma|$ vertices in ∂R^A. Thus, by Lemma 4, we have $|R^A| \le |\widehat{S^A}||\Sigma|/(|\Sigma| - 1) = O(|\widehat{S^A}|)$. We claim that $|\widehat{S^A}| = O(|\Sigma|L/\alpha^2)$.

To prove the claim, first note that by definition $\widehat{S^A} \subseteq S^A$, where

$$S^A = \{x\sigma \mid D^A(x\Sigma^*) > \alpha/3 \wedge D^A(x\sigma\Sigma^*) \le \alpha\}. \tag{7}$$

Now consider the following set:

$$Q^A = \{y \mid D^A(y\Sigma^*) > \alpha/3 \wedge \forall\sigma\, D^A(y\sigma\Sigma^*) \le \alpha\}, \tag{8}$$

and observe that it is prefix-free; this implies $|Q^A| < 3/\alpha$. Now we establish the following correspondence between S^A and Q^A: each $x \in S^A$ can be uniquely mapped to some $y \in Q^A$, and via this mapping each $y \in Q^A$ is assigned at most $|\Sigma| + (|\Sigma| - 1)|y|$ elements of S^A. Indeed, given $x\sigma \in S^A$ one can see that either $x \in Q^A$ or $xw \in Q^A$ for some suffix w; any ambiguity is resolved by taking the first such w in lexicographical order. Furthermore, given $y \in Q^A$, any element of S^A that is assigned to y by this mapping is of the form $y\sigma$ or $z\sigma$ for some (possibly empty) proper prefix z of $y = z\sigma' w$ with $\sigma \ne \sigma'$. The claim follows from the observation that Markov's inequality implies $|y| \le 3L/\alpha$ for all $y \in Q^A$. Therefore, to build T we explore at most $O(|\Sigma|L/\alpha^2)$ nodes, effecting at most $O(|\Sigma|L/\alpha^2)$ statistical queries with tolerance $\Omega(\alpha\beta)$. This concludes the proof. □

The following corollary gives a specialization of the previous simulation for the case when a bound on the length of the words generated by D is known.

Corollary 6. *If $\mathrm{supp}(D) \subseteq \Sigma^{\le n}$, then a call to $\mathrm{DIFF}^D_\infty(A, B, \alpha, \beta)$ can be simulated using $O(|\Sigma|n/\alpha)$ calls to SQ^D with tolerance $\Omega(\alpha\beta)$.*

Proof. Just inspect the previous proof and use the fact that $|x| > n$ implies $D^A(x\Sigma^*) = 0$ and $D^B(x\Sigma^*) = 0$. □

As a consequence of our simulation we observe that algorithm CT is in fact a Statistical Query algorithm, an observation which, to the best of our knowledge, has not been made before.

Corollary 7. *There exists an algorithm $\mathtt{CT}_{\mathtt{SQ}}$ that, given access to an oracle SQ^T for some $T \in \mathcal{PDFA}_{n,\mu}$ with expected length L, on input $\mathtt{CT}_{\mathtt{SQ}}(n, \Sigma, \mu, L, \epsilon)$ makes $O(n^2|\Sigma|^2/(L\mu^2))$ γ-bounded statistical queries, and returns a PDFA H such that $\mathrm{KL}(T\|H) \le \epsilon$, where $\gamma = poly(\epsilon, \mu, 1/n, 1/|\Sigma|, 1/L)$.*

Similar claims can be made of the variants of CT in [12,13,1,14]. Also, algorithms in the ALERGIA family [16,15] can be rewritten to access to access the target via L_∞-queries and Statistical Queries only.

4. A lower bound for learning PDFA

In this section we prove a lower bound on the number of L_∞-queries that any (α, β)-bounded algorithm learning the class of all PDFA must make. In order to do so, we show such a lower bound for a class of distributions defined using parities that can be modeled using PDFA. Our lower bound is unconditional, and applies to algorithms that learn w.r.t. KL as well as w.r.t. L_1. Furthermore, it is representation-independent; that is, it holds whatever kind of hypothesis the algorithm outputs. From the main result we deduce another lower bound in terms of the distinguishability that describes a trade-off between the number of queries and the tolerance and threshold parameters.

Fix $\Sigma = \{0, 1\}$ and let $\mathcal{D}_n$ denote the class containing the distributions D_h for all $h \in \Sigma^n$; all these are distributions over Σ^{n+1} representable by a PDFA with $\Theta(n)$ states.

Before stating our lower bound we remark that a similar, though worse, result already follows from a result by Kearns [20] via the simulation given in the previous section. Indeed, Kearns proved that learning parity *concepts* is a difficult problem in the statistical query model. More precisely, he showed the following.

Theorem 8 (*Kearns [20]*). *Any γ-bounded statistical query algorithm that learns the class of all parity concepts must make at least $\Omega(2^n\gamma^2)$ queries.*

Combining the simulation given by Corollary 6 with a simple reduction argument, the following result about the number of L_∞-queries needed for learning $\mathcal{D}_n$ follows.

Corollary 9. *Any (α, β)-bounded L_∞-query algorithm that learns the class $\mathcal{D}_n$ with $\epsilon \leq 1/9$ must make at least $\Omega(2^n\alpha^3\beta^2/n)$ queries.*

This bound can be improved using a direct proof as is shown at the end of this section. In particular, we are able to prove the following lower bound, which improves the dependence on n and α in the previous one.

Theorem 10. *Any (α, β)-bounded L_∞-query algorithm that learns the class $\mathcal{D}_n$ with $\epsilon \leq 1/9$ must make at least $\Omega(2^n\alpha^2\beta^2)$ queries.*

An immediate consequence of this theorem is that the same lower bound applies to algorithms that learn the class of all PDFA.

Corollary 11. *Any (α, β)-bounded L_∞-query algorithm that learns the class $\mathcal{PDFA}_n$ with $\epsilon \leq 1/9$ must make at least $\Omega(2^n\alpha^2\beta^2)$ queries.*

Theorem 10 is proved by devising an adversary that answers in a malicious way each query asked by the learning algorithm. The argument is similar in nature to that used in [20] to prove that parities can not be learned in the Statistical Query model. Basically, we show that the number of distributions in $\mathcal{D}_n$ that are inconsistent with each answer given by the adversary it is at most $O(1/(\alpha^2\beta^2))$. Since there are 2^n distributions in $\mathcal{D}_n$, any learning algorithm is unable to distinguish the correct target with $o(2^n\alpha^2\beta^2)$ queries. By special properties of distributions in $\mathcal{D}_n$, the adversary can always find a distribution which is consistent with every answer given to the learner but still has a large error w.r.t. the hypothesis provided by the learner. The details of the proof are deferred to the end of this section. Now we discuss some corollaries of Theorem 10.

An interesting consequence of our lower bound in terms of α and β is that, by a simple padding argument, it allows us to derive a lower bound in terms of the distinguishability. The dependence on ϵ in the statement is ignored.

Corollary 12. *Suppose $2\log(1/\mu) + 3 \leq n \leq (1/\mu)^{o(1)}$. Let $a, b, c \geq 0$ be constants and take $\alpha = \mu^a \cdot n^{O(1)}$ and $\beta = \mu^b \cdot n^{O(1)}$. If an (α, β)-bounded L_∞-query algorithm learns the class $\mathcal{PDFA}_{n,\mu}$ using at most $n^{O(1)}/\mu^c$ queries, then necessarily $2a + 2b + c \geq 1$.*

Proof. Recall the class of distributions $\mathcal{D}_k$ from Theorem 10. For all positive integers m and k, define the class of distributions $\mathcal{C}_{m,k}$ as follows: for every distribution D in $\mathcal{D}_k$, there is a distribution in $\mathcal{C}_{m,k}$ that gives probability $D(x)$ to each string of the form $0^m x$, and 0 to strings not of this form. Every distribution in $\mathcal{D}_k$ is generated by a PDFA with at most $2k + 2$ states and distinguishability $2/2^k$. It follows that every distribution in $\mathcal{C}_{m,k}$ is generated by a PDFA with at most $m + 2k + 2$ states and distinguishability 2^{-k}. Assuming w.l.o.g. that $k = \log(1/\mu)$ is an integer, for $m = n - 2k - 2$ we have $\mathcal{C}_{m,k} \subseteq \mathcal{PDFA}_{n,\mu}$.

Now note that, by an immediate reduction, any (α, β)-bounded L_∞-query algorithm that learns $\mathcal{P}_{n,\mu}$ using at most $n^{O(1)}/\mu^c$ queries can learn the class $\mathcal{D}_k$ with the same number of queries. Therefore, Theorem 10 implies that necessarily $n^{O(1)}/\mu^c = \Omega(\alpha^2\beta^2/\mu)$. Substituting for α and β, and using that $1/n = \mu^{o(1)}$, the bound yields

$$\frac{1}{\mu^{c+o(1)}} = \Omega\left(\frac{\mu^{2a+2b}}{\mu^{1+o(1)}}\right). \tag{9}$$

Therefore, since $1/\mu = \omega(1)$, we conclude that $2a + 2b + c \geq 1$. □

The next corollary collects the extreme cases of the previous one. These results illustrate the trade-off that algorithms for learning PDFA must face: either ask a lot of queries with moderate tolerance and threshold, or ask a few queries with very small values of tolerance and threshold.

Corollary 13. *Suppose $2\log(1/\mu) + 3 \leq n \leq (1/\mu)^{o(1)}$. Then the following hold:*

(i) *If $\alpha, \beta = \mu^{o(1)} \cdot n^{O(1)}$, then any (α, β)-bounded L_∞-query algorithm that learns $\mathcal{PDFA}_{n,\mu}$ must make at least $\Omega(1/\mu^{1-\nu})$ queries for any constant $\nu > 0$.*
(ii) *If an (α, β)-bounded L_∞-query algorithm learns $\mathcal{PDFA}_{n,\mu}$ with $n^{O(1)}$ queries, then necessarily $\alpha^2\beta^2 = O(\mu \cdot n^{O(1)})$.*

Proof. Both statements follow by inspection from the previous proof. □

4.1. Technical lemmata

Before getting to the proof of Theorem 10 we must go over a chain of lemmata. We begin with two simple bounds which are stated without proof.

Lemma 14. *Let X be a random variable and $\tau, \gamma \in \mathbb{R}$. The following hold:*

(i) *if $\tau > |\gamma|$, then $\mathbb{P}[|X| > \tau] \leq \mathbb{P}[|X - \gamma| > \tau - |\gamma|]$,*
(ii) *$\mathbb{P}[|X| < \tau] \leq P[|X - \gamma| > \gamma - \tau]$.*

The following technical lemma bounds the probability that the maximum of a finite set of random variables deviates from some quantity in terms of the individual variances. Let $\{X_i\}_{i\in[m]}$ be a finite collection of random variables with $\gamma_i = \mathbb{E}[X_i]$ and $\gamma_1 \geq \cdots \geq \gamma_m$. Let $Z = \max_{i\in[m]} |X_i|$ and $\gamma = \max_{i\in[m]} |\gamma_i|$.

Lemma 15. *If $\gamma = \gamma_1$, then for all $t > 0$*

$$\mathbb{P}[|Z - \gamma| > t] \leq \frac{\mathbb{V}[X_1]}{t^2} + \sum_{i\in[m]} \frac{\mathbb{V}[X_i]}{t^2}. \tag{10}$$

Proof. In the first place, note that by (i) in Lemma 14 and the definition of γ we have, for any $i \in [m]$, $\mathbb{P}[|X_i| > \gamma + t] \leq \mathbb{P}[|X_i - \gamma_i| > t]$. Thus, by the union bound and Chebyshev's inequality,

$$\mathbb{P}[Z > \gamma + t] \leq \sum_{i\in[m]} \frac{\mathbb{V}[X_i]}{t^2}. \tag{11}$$

On the other hand, for any $i \in [m]$ we have $\mathbb{P}[Z < \gamma - t] \leq \mathbb{P}[|X_i| < \gamma - t] \leq \mathbb{P}[|X_i - \gamma_i| > \gamma_i - \gamma + t]$, where the last inequality follows from (ii) in Lemma 14. Choosing $i = 1$ we obtain

$$\mathbb{P}[Z < \gamma - t] \leq \frac{\mathbb{V}[X_1]}{t^2}. \tag{12}$$

The result follows from (11) and (12). □

For any $A \subseteq \Sigma^*$, $c \in \Sigma$ and $h \in \Sigma^n$, we write $A_n = A \cap \Sigma^n$ and $A_h^c = \{x \in A_n \mid P_h(x) = c\}$. Let $A, B \subseteq \Sigma^{1:n} = \Sigma^{\leq n} \cap \Sigma^+$ be disjoint and prefix-free. Our next lemma shows that, for any $D_h \in \mathcal{D}_n$, one can write $\mathrm{L}_\infty(D_h^A, D_h^B)$ as the maximum of $2n$ easily computable quantities. Let $p_A = D_h(A\Sigma^*)$ and $p_B = D_h(B\Sigma^*)$, and note that we have $p_A = \sum_{k=1}^n |A_k|/2^k$ and $p_B = \sum_{k=1}^n |B_k|/2^k$; that is, these probabilities are independent of h. Now define the following quantities:

$$X_k^c = \frac{|A_{h_{1:k}}^c|}{p_A} - \frac{|B_{h_{1:k}}^c|}{p_B}. \tag{13}$$

These quantities can be used to easily compute the supremum distance between D^A and D^B.

Lemma 16. *With the above notation, the following holds:*

$$\mathrm{L}_\infty(D_h^A, D_h^B) = \max_{k\in[n], c\in\Sigma} |X_k^c| / 2^n. \tag{14}$$

Proof. First we write, for $1 \leq k \leq n$, $Y_k = \max_{y\in\Sigma^{n+1-k}} |D_h^A(y) - D_h^B(y)|$, and note that $\mathrm{L}_\infty(D_h^A, D_h^B) = \max_{1\leq k\leq n} Y_k$. Now we show that, for $k \in [n]$, $Y_k = \max\{|X_k^0|, |X_k^1|\}/2^n$. First observe that for $y \in \Sigma^{n-k}$ and $z \in \Sigma$ we have $D_h^A(yz) = \sum_{x\in A\cap\Sigma^k} D_h(xyz)/p_A$. Furthermore, $D_h(xyz) = 2^{-n}$ if and only if $P_{h_{1:k}}(x) \oplus P_{h_{k+1:n}}(y) = z$. Thus, for fixed y and z, we have

$$\sum_{x\in A\cap\Sigma^k} D_h(xyz) = \begin{cases} |A_{h_{1:k}}^0|/2^n & \text{if } P_{h_{k+1:n}}(y) = z, \\ |A_{h_{1:k}}^1|/2^n & \text{if } P_{h_{k+1:n}}(y) \neq z. \end{cases} \tag{15}$$

This concludes the proof. □

Now we will consider the quantities X_k^c as random variables when $h \in \Sigma^n$ is taken uniformly at random. We are interested in the expectation and variance of these quantities.

Lemma 17. *When h is taken uniformly at random, the following hold:*

(i) $\mathbb{E}[X_k^c] = |A_k|/(2p_A) - |B_k|/(2p_B) + O(1)$, *and*
(ii) $\mathbb{V}[X_k^c] = O(|A_k|/p_A^2 + |B_k|/p_B^2 + (|A_k| + |B_k|)/(p_A p_B))$.

Proof. To begin, note that, taking $g \in \Sigma^k$ uniformly at random, the random variables $|A_{h_{1:k}}^c|$ and $|A_g^c|$ follow the same distribution. Thus, we can use g instead of $h_{1:k}$ in the definition of X_k^c.

We start by computing $\mathbb{E}[|A_g^c|]$, $\mathbb{E}[|A_g^c|^2]$ and $\mathbb{E}[|A_g^c||B_g^c|]$. Using indicator variables one has

$$\mathbb{E}[|A_g^c|] = \sum_{x \in A_k} \mathbb{P}[P_g(x) = c] = \frac{|A_k'|}{2} + \mathbb{1}_{\bar{0} \in A_k \wedge c=0} = \frac{|A_k|}{2} + O(1), \tag{16}$$

where $A_k' = A_k \setminus \{\bar{0}\}$. The second term models the fact that $P_g(\bar{0}) = 0$ for all g. This implies (i). To compute $\mathbb{E}[|A_g^c|^2]$ we will use that for different $x, y \in \Sigma^k \setminus \{\bar{0}\}$ a standard linear algebra argument shows $\mathbb{P}[P_g(x) = c \wedge P_g(y) = c] = 1/4$. Furthermore, note that if either $x = \bar{0}$ or $y = \bar{0}$, but not both, then $\mathbb{P}[P_g(x) = c \wedge P_g(y) = c] = 1/2$ if $c = 0$, and it is zero if $c = 1$. Hence, we have

$$\sum_{x \in A_k} \sum_{y \in A_k \setminus \{x\}} \mathbb{P}[P_g(x) = c \wedge P_g(y) = c] = \frac{|A_k'|^2}{4} - \frac{|A_k'|}{4} + \mathbb{1}_{\bar{0} \in A_k \wedge c=0}(|A_k| - 1). \tag{17}$$

Now, combining Eqs. (16) and (17) we obtain

$$\begin{aligned} \mathbb{E}[|A_g^c|^2] &= \sum_{x \in A_k} \sum_{y \in A_k \setminus \{x\}} \mathbb{P}[P_g(x) = c \wedge P_g(y) = c] + \sum_{x \in A_k} \mathbb{P}[P_g(x) = c] & (18) \\ &= \frac{|A_k|^2}{4} + O(|A_k|). & (19) \end{aligned}$$

Furthermore, since $\mathbb{1}_{\bar{0} \in A_k} \mathbb{1}_{\bar{0} \in B_k} = 0$ because A and B are disjoint, similar arguments to those used before yield

$$\begin{aligned} \mathbb{E}[|A_g^c||B_g^c|] &= \sum_{x \in A_k} \sum_{y \in B_k} \mathbb{P}[P_g(x) = c \wedge P_g(y) = c] & (20) \\ &= \frac{|A_k'||B_k'|}{4} + \mathbb{1}_{c=0} \left(\mathbb{1}_{\bar{0} \in B_k} \frac{|A_k|}{2} + \mathbb{1}_{\bar{0} \in A_k} \frac{|B_k|}{2} \right) & (21) \\ &= \frac{|A_k||B_k|}{4} + O(|A_k| + |B_k|). & (22) \end{aligned}$$

Finally, Eqs. (16), (19) and (22) can be used to compute $\mathbb{V}[X_g^c]$. The bound in (ii) follows from the observation that all terms having $|A_k|^2$, $|B_k|^2$ or $|A_k||B_k|$ are cancelled. □

4.2. *Proof of the lower bound*

Finally, we combine the preceding lemmata to prove Theorem 10.

Proof of Theorem 10. Suppose L is an (α, β)-bounded L_∞-query algorithm that learns the class $\mathcal{D}_n$. Now we describe and analyze an adversary answering any L_∞-query asked by L.

Let $\mathrm{DIFF}_\infty^D(A, B, \alpha, \beta)$ be a query asked by L where, w.l.o.g. we assume it uses the smallest tolerance and threshold values allowed by the bounding restriction. Suppose that $h \in \Sigma^n$ is taken uniformly at random and let $p_A = \sum_{k=1}^n |A_k|/2^k$, $p_B = \sum_{k=1}^n |B_k|/2^k$, $\gamma_k^c = \mathbb{E}[X_k^c]/2^n$, where we use the quantities defined in Eq. (13). The adversary uses the expressions given by Lemma 17 to compute these quantities in terms of A and B. It then answers as follows:

(i) if either $p_A \leq 2\beta$ or $p_B \leq 2\beta$, then answer `'?'`,
(ii) otherwise, answer $\hat{\mu} = \max_{k \in [n], c \in \Sigma} |\gamma_k^c|$.

We claim that in each case the number of distributions in $\mathcal{D}_n$ inconsistent with these answers is at most $O(1/(\alpha^2\beta^2))$. If the adversary answers `'?'`, then every distribution in $\mathcal{D}_n$ is consistent with this answer because p_A and p_B are independent of h. In the other case, we use a probabilistic argument to prove the bound.

Suppose the adversary answers $\hat{\mu} = \max_{k \in [n], c \in \Sigma} |\gamma_k^c|$, and note that, interchanging A and B if necessary, we can assume $\hat{\mu} = \max_{k \in [n], c \in \Sigma} \gamma_k^c$. In this case, the number of inconsistent distributions is $2^n\, \mathbb{P}[|\mathrm{L}_\infty(D_h^A, D_h^B) - \hat{\mu}| \geq \alpha]$. We can combine Lemmas 15 and 16 to get

$$\mathbb{P}[|\mathrm{L}_\infty(D_h^A, D_h^B) - \hat{\mu}| \geq \alpha] \leq \frac{1}{2^{2n}\alpha^2} \left(\mathbb{V}[X_M] + \sum_{k,c} \mathbb{V}[X_k^c] \right), \tag{23}$$

where $X_M \in \{X_1^0, \dots, X_n^1\}$ is such that $\hat{\mu} = \mathbb{E}[X_M]/2^n$. Since A and B are disjoint and thus $\sum_{k=1}^n (|A_k| + |B_k|) \leq 2^n$, using the bound (ii) from Lemma 17 we obtain

$$\mathbb{V}[X_M] + \sum_{k,c} \mathbb{V}[X_k^c] = O\left(\frac{2^n}{p_A^2} + \frac{2^n}{p_B^2} + \frac{2^n}{p_A p_B}\right) = O\left(\frac{2^n}{\beta^2}\right), \tag{24}$$

where the last inequality uses that $p_A, p_B > 2\beta$. Therefore, we can conclude that the number of distributions inconsistent with the answer $\hat{\mu}$ is at most $O(1/(\alpha^2\beta^2))$.

Let I denote the maximum number of distributions inconsistent with each query issued by $\mathtt{L}$ and $\mathtt{Q}$ be the number of queries it makes. Now we show that if $I \cdot \mathtt{Q} \leq 2^n - 2$, the algorithm necessarily produces a hypothesis with a large error.

Note that the relative entropy between any two distributions in $\mathcal{D}_n$ is infinite because they have different supports. Since $I \cdot \mathtt{Q} \leq 2^n - 2$ implies that there are at least two distributions in $\mathcal{D}_n$ consistent with every answer given by the adversary, if $\mathtt{L}$ outputs a hypothesis in $\mathcal{D}_n$, the adversary can still choose a consistent distribution with infinite error w.r.t. the hypothesis produced by $\mathtt{L}$. Recalling that for each pair of distributions in $\mathcal{D}_n$ we have $\mathrm{L}_1(D_f, D_g) = 1$, we also get a lower bound for learning $\mathcal{D}_n$ w.r.t. the variation distance. Furthermore, the following argument shows that $\mathtt{L}$ still has a large error even if its output is not restricted to a distribution in $\mathcal{D}_n$.

Assume $\mathtt{L}$ outputs some distribution $\hat{D}$, not necessarily in $\mathcal{D}_n$, such that $\mathrm{KL}(D_h \| \hat{D}) \leq \epsilon$ for some $D_h \in \mathcal{D}_n$. Then it follows from Pinsker's inequality [24] that $\mathrm{KL}(D_g \| \hat{D}) \geq (1/2\ln 2)(1 - \sqrt{2\ln 2\epsilon})^2$ for any other distribution D_g different from D_h. Since $\epsilon \leq 1/9$, we then have $\mathrm{KL}(D_g \| \hat{D}) > 2/9$. Therefore, if $I \cdot \mathtt{Q} \leq 2^n - 2$ the hypothesis produced by the learner will have a large error.

We can conclude that if $\mathtt{L}$ learns $\mathcal{D}_n$ with $\mathtt{Q}$ queries, then necessarily $\mathtt{Q} > (2^n - 2)/I = \Omega(2^n\alpha^2\beta^2)$. □

5. An adaptive learning algorithm

We show below a learning algorithm for PDFA that has as input parameters the alphabet size $|\Sigma|$, an upper bound n on the number of states of the target and the confidence parameter δ. In contrast with the CT algorithm, it does not need as input the distinguishability (or prefix-distinguishability) μ of the target, or the required precision ϵ, or the expected length L of strings emitted from any state.

Our algorithm is adaptive in nature, in the sense that it works in the more sensible framework — compared to that of CT — where it receives a sample as input and must extract as much information from it as possible. Note that given some input parameters, CT computes the number of examples it needs and asks for a sample of that size; this is not always possible in many practical cases where only a fixed size sample is available.

Furthermore, the sample bounds of our algorithm depend on the *prefix-distinguishability* of the target PDFA, instead of the distinguishability. This extends the approach followed in [10] for learning acyclic PDFA to the general case. For the purposes of this section, μ denotes the prefix-distinguishability of a given target T.

5.1. From an important graph to a correct hypothesis

According to Clark and Thollard [6], a PAC learner for the class of PDFA can be easily obtained from a polynomial-time algorithm that is able to partially recover the structure of the transitions between some states from the target. This section summarizes their approach and recalls some of their results; we mostly follow their notation.

An algorithm for learning the relevant part of the structure of a target PDFA uses a data structure called a *hypothesis graph*: a directed graph G with a distinguished node called the initial state, where each arc is labelled using a symbol from Σ with the same restrictions that apply to PDFA, and where each node is equipped with a multiset of strings from Σ^*. The size $|G|$ of a hypothesis graph is defined as the sum of the sizes of all the multisets assigned to its nodes.

It is shown in [6] how to obtain a PDFA with small error w.r.t. to a target PDFA T from a hypothesis graph satisfying a particular set of conditions. The following definition uses some quantities defined in Table 1 in terms of an accuracy parameter ϵ and some parameters of T, namely n, $|\Sigma|$, L and its prefix-distinguishability μ.

The following lemma encapsulates a useful claim implicit inside Clark and Thollard's long proof.

Definition 3. A hypothesis graph G is said to be ϵ-*important* w.r.t. a given PDFA T if the following are satisfied.

(i) G has a subgraph G' isomorphic to a subgraph of T. This means that there is a bijection Φ from a subset of states of T to all nodes of G' such that $\Phi(q_0) = v_0$ (where q_0, v_0 are the initial states of T and G, respectively), and if $\tau_{G'}(v, \sigma) = w$ then $\tau(\Phi^{-1}(v), \sigma) = \Phi^{-1}(w)$.

(ii) The states in T whose probability is greater than $\epsilon_2/(L+1)$ — which we call *frequent states* — have a representative in G'; that is, Φ is defined on frequent states.

(iii) If q is a frequent state in T and $\sigma \in \Sigma$ is such that $\gamma(q, \sigma) > \epsilon_5$ — we say (q, σ) is a *frequent transition* — then $\tau_{G'}(\Phi(q), \sigma)$ exists and it equals $\Phi(\tau(q, \sigma))$.

(iv) A multiset S_v is attached to every node v in the graph. If v represents a frequent target state q (i.e. $\Phi(q) = v$ where q is frequent), then for every $\sigma \in \Sigma \cup \{\xi\}$, it holds that $|S_v(\sigma)/|S_v| - \gamma(q, \sigma)| < \epsilon_1$. A multiset for which this property holds is said to be ϵ_1-*good*.

Table 1
Quantities used in the analysis of AdaCT.

	Description	Definition
ϵ_1	Error on transition probabilities	$\frac{\epsilon^2}{16(\lvert\Sigma\rvert+1)(L+1)^2}$
ϵ_2	Probability of frequent states	$\frac{\epsilon}{4n(n+1)L(L+1)\log(4(L+1)(\lvert\Sigma\rvert+1)/\epsilon)}$
ϵ_5	Probability of frequent transitions	$\frac{\epsilon}{4\lvert\Sigma\rvert(n+1)L(L+1)\log(4(L+1)(\lvert\Sigma\rvert+1)/\epsilon)}$
ϵ_0	Auxiliary quantity	$\frac{\epsilon_2\epsilon_5}{n\lvert\Sigma\rvert(L+1)}$
δ_0	Error probability of `Test_Distinct`	$\frac{\delta}{n(n\lvert\Sigma\rvert+\lvert\Sigma\rvert+1)}$
N_0	Examples for correct subgraph	$\frac{16\,e}{\epsilon_0\mu^2}(\ln\frac{16}{\epsilon_0\mu^2}+\ln\frac{32L}{\delta_0^2})$
N_1	Examples for finding frequent states	$\frac{8(L+1)}{\epsilon_2}\ln\frac{1}{\delta_0}$
N_2	Examples for finding frequent transitions	$\frac{8(L+1)}{\epsilon_2\epsilon_5}\ln\frac{1}{\delta_0}$
N_3	Examples for good multisets	$\frac{1}{\epsilon_0\epsilon_1^2}\ln\frac{2(\lvert\Sigma\rvert+1)}{\delta_0}$

Lemma 18 (*From [6]*). *There exists an algorithm* `ImportantToCorrect` *that given an ϵ-important hypothesis graph G for some PDFA T outputs a PDFA H with $\mathrm{KL}(T\|H) \leq \epsilon$. Furthermore,* `ImportantToCorrect` *runs in time poly$(|\Sigma|, |G|)$.*

The algorithm `ImportantToCorrect` is described in the paragraphs "Completing the Graph" and "Estimating Probabilities" of [6]. Basically, it is enough to complete the graph when necessary by introducing a new node — the *ground node* — representing all low frequency states not in G and adding new transitions to the ground node. Finally, a smoothing scheme is performed in order to estimate transition probabilities.

The proof of Lemma 18 is essentially the contents of Sections 4.3, 4.4 and 5 in [6]. It does not involve distinguishability at all, so we can apply it in our setting even if we have changed our measure of distinguishability.

5.2. *Description of the algorithm*

Our learning algorithm takes as inputs $|\Sigma|$, δ, n and a sample S containing N examples drawn i.i.d. from some target PDFA T with at most n states. AdaCT builds a hypothesis graph G incrementally; it performs at most $n|\Sigma|$ learning stages, each one making a pass over all training examples and guaranteed to add one transition to G. We measure the size $|S|$ of the sample by the sum (with repetitions) of the length of all its strings.

For ease of exposure we split the stages in the learning process into two periods. The first one is called the *important period*, and covers an initial portion of stages. During the important period, the graph G that AdaCT is building has theoretical guarantees of being a good representation of the target. In contrast, transitions added in stages of the second period may be wrong. We will show that the output G of AdaCT is important, where the subgraph G' from Definition 3 will be exactly the portion of G built during the important period.

At the beginning of each stage AdaCT has a graph G that summarizes our current knowledge of the target T. Nodes and edges in G represent, respectively, states and transitions of the target T. We call *safe nodes* the nodes of G, as during the important period they are inductively guaranteed (with probability at least $1 - \delta_0$) to stand for distinct states of T. Safe nodes are denoted by strings in Σ^*.

Attached to each safe node v there is a multiset S_v. If v was added to G during the important period, S_v keeps information about the distribution on the target state represented by v. The algorithm starts with a graph G consisting of a single node v_0, representing the initial state of the target, whose attached multiset equals the sample received as input.

When a new stage starts the learner creates a candidate node $u = v_u\sigma$ for each (safe) node v_u of G and each $\sigma \in \Sigma$ such that $\tau_G(v_u, \sigma)$ is undefined. Candidate nodes gather information about transitions in T leading from states that have already a safe representative in G but do not have yet an edge counterpart in G. Attached to each candidate node u there is also a multiset S_u, initially empty.

For each training example $x\xi = \sigma_0 \cdots \sigma_i \cdots \sigma_k\xi$ in the sample, AdaCT traverses the graph matching each observation σ_i to a state until either all observations in x have been exhausted and the example is discarded, or a transition to a candidate node is reached. This occurs when all transitions up to σ_{i-1} are defined and lead to a safe node w, but there is no edge out of w labeled by σ_i. Then the suffix $\sigma_{i+1} \ldots \sigma_k$ is added to the multiset of candidate $u = w\sigma_i$. After that the following example, if any, is considered.

When all training examples have been processed our algorithm chooses a candidate $u = v_u\sigma$ whose attached multiset S_u has maximum cardinality. Then it checks whether there is strong evidence that candidate u corresponds to a target state that has not yet a representative in G. This is done by calling repeatedly the function `Test_Distinct`(u, v) described in Fig. 1, varying v over the set of safe nodes, until some call returns `'Not Clear'` or all safe nodes have been tested. Note that if `Test_Distinct`(u, v) returns `'Distinct'`, we assume that u and v correspond to distinct states in the target.

Once all tests have been done, if all safe nodes have been found to be (supposedly) distinct from u, candidate u is promoted to a new safe node — i.e. G gets a new node labelled with u — u is not anymore a candidate node, and an edge from v_u to u

labeled by σ is added to G. Multiset S_u is attached to the new safe node. Otherwise, the algorithm chooses a safe node v that has not been distinguished from u, and identifies nodes $v_u\sigma$ and v by adding to G an edge from v_u to v labeled by σ; $v_u\sigma$ is not anymore a candidate node.

Finally, if either there are no remaining candidates or G contains already $n|\Sigma|$ arcs, AdaCT returns G and stops. Otherwise, the current stage ends by erasing all candidates, and a new stage begins.

```
Input: a candidate node $u$ and a safe node $v$
Output: 'Distinct' or 'Not Clear'
$m_u \leftarrow |S_u|$; $s_u \leftarrow |\text{prefixes}(S_u)|$;
$m_v \leftarrow |S_v|$; $s_v \leftarrow |\text{prefixes}(S_v)|$;
$t_{u,v} \leftarrow \sqrt{\frac{2}{\min(m_u,m_v)} \ln \frac{8(s_u+s_v)}{\delta_0}}$;
$d \leftarrow \max\{\mathrm{L}_\infty(\widehat{S}_u, \widehat{S}_v), \mathrm{prefL}_\infty(\widehat{S}_u, \widehat{S}_v)\}$;
if $d > t_{u,v}$ then
    return 'Distinct';
else
    return 'Not Clear';
end
```

Fig. 1. The state-distinctness test function Test_Distinct.

5.3. Analysis

In this section we show that algorithm AdaCT returns an ϵ-important hypothesis graph for T with probability at least $1-\delta$, and therefore can be turned into a PAC-learner for PDFA using the post-processing algorithm ImportantToCorrect from Lemma 18.

We begin by proving two lemmata about the behavior of Test_Distinct. Here D_u (resp. D_v) denotes the target distribution on state $q = \tau(q_0, u)$ (resp. $q = \tau(q_0, v)$).

Lemma 19. *Assume S_u and S_v are respectively samples from distributions D_u and D_v. If $D_u = D_v$, then function* Test_Distinct *(u, v) returns* 'Not Clear' *with probability $1-\delta_0$.*

Proof. Let $D = D_u = D_v$. Function Test_Distinct returns 'Distinct' when there exists a string x such that $|\hat{S}_u(x\Sigma^*) - \hat{S}_v(x\Sigma^*)| > t_{u,v}$ or $|\hat{S}_u(x) - \hat{S}_v(x)| > t_{u,v}$. First, we bound the probability of the event $\mathrm{prefL}_\infty(\hat{S}_u, \hat{S}_v) > t_{u,v}$. To avoid summing over infinitely many x, consider $S_u \cup S_v$ ordered, say lexicographically. Then the event above is equivalent to saying "there is an index i in this ordering such that some prefix of the ith string in $S_u \cup S_v$ in this ordering, call it x, satisfies the condition above". (This is another way of saying that only x's appearing in prefixes $(S_u \cup S_v)$ can make the inequality true, since all others have $\hat{S}_u(x\Sigma^*) = \hat{S}_v(x\Sigma^*) = 0$.) Therefore, its probability is bounded above by the maximum of $(s_u + s_v)\mathbb{P}[\,|\hat{S}_u(x\Sigma^*) - \hat{S}_v(x\Sigma^*)| > t_{u,v}]$ over all strings x. By the triangle inequality, this is at most

$$(s_u + s_v)\left(\mathbb{P}[\,|\hat{S}_u(x\Sigma^*) - D(x\Sigma^*)| > t_{u,v}/2] + \mathbb{P}[|\hat{S}_v(x) - D(x\Sigma^*)| > t_{u,v}/2]\right). \tag{25}$$

Since $\mathbb{E}[\hat{S}_u(x\Sigma^*)] = D(x\Sigma^*)$ and $\mathbb{E}[\hat{S}_v(x\Sigma^*)] = D(x\Sigma^*)$, by Hoeffding's inequality this is at most

$$(s_u + s_v)(2\,\exp(-2(t_{u,v}^2/4)\,m_u) + 2\,\exp(-2(t_{u,v}^2/4)\,m_u)) \tag{26}$$

$$\leq 4(s_u + s_v)\exp(-(t_{u,v}^2/2)\,\min(m_u, m_v)), \tag{27}$$

which is $\delta_0/2$ by definition of $t_{u,v}$.

A similar reasoning also shows that the probability of the event $\mathrm{L}_\infty(\hat{S}_u, \hat{S}_v) > t_{u,v}$ is at most $\delta_0/2$ and we are done. □

Lemma 20. *Assume S_u and S_v are samples from D_u and D_v, respectively. If the distributions D_u and D_v are μ-prefix-distinguishable and $\min(m_u, m_v) \geq (8/\mu^2)\ln(16(m_u + m_v)L/\delta_0^2)$ then* Test_Distinct(u, v) *returns* 'Distinct' *with probability $1-\delta_0$.*

Proof. We first bound the size of prefixes$(S_u \cup S_v)$. Clearly, its expected size is at most $L|S_u \cup S_v|$. Then, by Markov's inequality, $\mathbb{P}[|\text{prefixes}(S_u \cup S_v)| \geq 2L|S_u \cup S_v|/\delta_0]$ is less than $\delta_0/2$. Therefore, we have with probability at least $1-\delta_0/2$ that $s_u + s_v \leq 2(m_u + m_v)L/\delta_0$.

Now assume there is a string x witnessing that $\mathrm{prefL}_\infty(D_u, D_v) > \mu$ (otherwise some x is a witness for $\mathrm{L}_\infty(D_u, D_v) > \mu$ and we argue in a similar way), i.e. a string such that $|D_u(x\Sigma^*) - D_v(x\Sigma^*)| > \mu$. If $\min(m_u, m_v) \geq (8/\mu^2)\ln(16(m_u + m_v)L/\delta_0^2)$, by the argument above with high probability we have $t_{u,v} \leq \mu/2$ and the probability of returning 'Distinct' is at least the probability of the event $|\hat{S}_u(x\Sigma^*) - \hat{S}_v(x\Sigma^*)| > \mu/2$. The hypothesis on x and the triangle inequality shows that the probability of the complementary event $|\hat{S}_u(x\Sigma^*) - \hat{S}_v(x\Sigma^*)| \leq \mu/2$ is at most $\mathbb{P}[\,|\hat{S}_u(x\Sigma^*) - D_u(x\Sigma^*)| > \mu/4] + \mathbb{P}[\,|\hat{S}_v(x\Sigma^*) - D_v(x\Sigma^*)| > \mu/4]$. By the Hoeffding bound, this sum is less than $\delta_0/2$, and we are done. □

We say that a node (candidate or safe) u is *important* when $m_u = |S_u|$ is at least $\epsilon_0 N/2$. If the selected candidate in a learning stage is important we say that the stage is *important*. The important period covers all initial stages until the first non-important one, which belongs to the second period. Note that the second period may be empty. We will denote by G' the portion of G built during the important period. Now we show that, with high probability, G' will contain correct representatives of all frequent states and transitions in T.

Lemma 21. *Assume $N > N_0$ and that stages $1, \ldots, k$ of* `AdaCT` *are important. Then, with probability $1 - kn\delta_0$, at the beginning of stage $k+1$ all nodes of graph G are important and G is isomorphic to a subgraph of T.*

Proof. By induction on k. The lemma is trivial for $k = 0$. Now assume that graph G is isomorphic to a subgraph of T and that nodes of G are important at the beginning of stage k. Clearly, G must have at most n nodes. Let u be the chosen candidate in this stage. If u is promoted to safe the error probability is incremented by at most δ_0. This is because Lemma 19 shows that the error probability of a promotion is δ_0: if node u represents the same target state as a safe node v then it has very little probability of being promoted. On the other hand, if u is merged with some safe node v the error probability is incremented by at most $n\delta_0$: as these nodes are important, the size requirements of Lemma 20 are satisfied, and according to this lemma, if nodes u and v were representatives of different target states, then the call `Test_Distinct`(u, v) would return the wrong value `'Not Clear'` with probability at most δ_0. As there are at most n safe nodes, the error probability of a merge is at most $n\delta_0$. □

Lemma 22. *Let q be a frequent target state from T that has no representative in G at the beginning of stage k. If $N > N_0$ then, with probability $1 - \delta_0$, stage k is important.*

Proof. By the definition of ϵ_0 it holds that $N > N_1$. Note that if G has no node representing q every training example that traverses q generates a new example for some candidate multiset. As the probability of q is greater than $\epsilon_2/(L+1)$ and $N > N_1$, Chernoff bounds show that with probability greater than $1 - \delta_0$ there are at least $N\epsilon_2/2(L+1)$ examples reaching a candidate node. There are at most $n|\Sigma|$ candidate nodes, so by the definition of ϵ_0 the attached multiset of the chosen candidate u at stage k must have cardinality $m_u \geq \epsilon_0 N/2$. □

The proof of the next lemma concerning frequent transitions follows the previous one by replacing ϵ_2 with $\epsilon_2\epsilon_5$; that is, using N_2 instead of N_1.

Lemma 23. *Let (q, σ) be a frequent target transition from T that has no representative in G at the beginning of stage k. If $N > N_0$ then, with probability $1 - \delta_0$, stage k is important.*

So far we have dealt with the requirement that `AdaCT` must output a hypothesis graph containing a subgraph that correctly captures the frequent structure of T. Now we turn to the requirement on the multisets attached to the nodes in the hypothesis graph.

Lemma 24. *Assume subgraph G is isomorphic to a subgraph of T and all attached multisets of nodes in G have cardinality at least $\epsilon_0 N/2$. If $N > N_3$ then, with probability $1 - n\delta_0$, the attached multisets of nodes in G are ϵ_1-good.*

Proof. It follows from the observation that all strings in the attached multisets of G are correctly placed because G is isomorphic to a subgraph of T, the hypothesis that all attached multisets of nodes in G have cardinality at least $\epsilon_0 N/2$, and the Hoeffding bound. □

Theorem 25. *If $N > \max\{N_0, N_3\}$, with probability $1 - \delta$, algorithm* `AdaCT`(n, Σ, δ, S) *returns a hypothesis graph G ϵ-important w.r.t. T. Furthermore,* `AdaCT` *runs in time $O(n^2 |\Sigma| |S|)$.*

Proof. Let G' denote the subgraph of G built during the important period. According to Lemma 21, with probability at least $1 - n^2|\Sigma|\delta_0$, G' is isomorphic to a subgraph of T. Furthermore, using Lemmas 22 and 23, a union bound argument shows that, in addition, with probability at least $1 - n(n+1)|\Sigma|\delta_0$ each frequent state and transition in T has a counterpart in G', and all nodes in G' have multisets of cardinality at least $\epsilon_0 N/2$. Finally, a last union bound applied to Lemma 24 shows that these multisets are ϵ_1-good with probability at least $1 - \delta$. We conclude that G is ϵ-important w.r.t. T with probability at least $1 - \delta$. The running time of `AdaCT` can be easily bounded: there are most $n|\Sigma|$ learning stages, in each of which at most n similarity tests are performed, and each tests runs in time linear with the size of the associated mutilsets which is at most $|S|$. □

As explained already, the graph G returned by `AdaCT` can be post-processed using `ImportantToCorrect` in order to obtain a PDFA H with $\mathrm{KL}(T\|H) \leq \epsilon$. This whole learning process can be performed in time polynomial in n, $|\Sigma|$ and the size of the input sample, measured by the sum of the lengths of all strings; in particular, we stress that the running time depends linearly on the size $|S|$ of the sample, no matter what it is. The hypothesis $N > \max\{N_0, N_3\}$ holds whenever the sample contains roughly

$$\widetilde{O}\left(\frac{n^4 L^5 |\Sigma|^2}{\epsilon^2} \cdot \max\left\{\frac{1}{\mu^2}, \frac{L^4|\Sigma|^2}{\epsilon^4}\right\}\right) \tag{28}$$

examples. That is, the algorithm will PAC learn for samples about this size or larger.

Interestingly, our proof methods, in addition of providing an adaptive algorithm, give slight improvements on the asymptotic sample complexity needed for PAC-learning PDFA. In particular, our sample bound has a dependence $\widetilde{O}(n^4|\Sigma|^4)$ on n and $|\Sigma|$, while the bound proved for CT in [6] has a dependence of type $\widetilde{O}(n^5|\Sigma|^5)$. Furthermore, some experimental results with a previous version of AdaCT were presented in [1]. These results showed that AdaCT can achieve low error rates with sample sizes much smaller than required by the bounds, and within an order of magnitude of those reported for heuristics without rigorous guarantees such as [16,15]. Furthermore, AdaCT has a running time comparable to that of the heuristic algorithms from [16,15], whose dependence on the size of the sample is also linear.

6. Discussion

Let us discuss the results shown so far and indicate possible lines for future research. We will use two examples to illustrate some points in our discussion. In particular, these examples stress some particularities of current measures of complexity for the problem of learning PDFA.

Our first example shows that there can be an exponential gap between the distinguishability and the prefix-distinguishability of a PDFA. Recall the class $\mathcal{D}_n$ of PDFA that define probability distributions with support defined by parities of n bits. Let $T \in \mathcal{D}_{n/2}$ be an acyclic PDFA of $n+2$ states over $\{0, 1\}$ with a single stopping state q_{n+1}. Note that for $0 \leq i \leq n$ we have $\gamma(q_i, 0) = \gamma(q_i, 1) = 1/2$. Now let T' be the following perturbation of T: for $0 \leq i \leq n$ let $\gamma'(q'_i, 0) = 1/2-i/(2n)$ and $\gamma'(q'_i, 1) = 1/2+i/(2n)$. One can see that T' has distinguishability $\mu = (1/2)^{\Theta(n)}$ and prefix-distinguishability $\mu' = \Theta(1/n)$.

This example motivates the usage of prefix-distinguishability for PDFA learning: while statistical tests for prefix-distinguishability are as simple as the ones used for the standard distinguishability in [6], in some (and hopefully many) examples states can be prefix-distinguished with many fewer examples than those needed to distinguish them w.r.t. μ. However, we recall that for some classes of PDFA distinguishability and prefix-distinguishability are essentially the same. Since our lower bound was proved in terms of μ and the upper bound in terms of μ', we see that this is the strongest possible combination of results given these two measures of distinguishability.

Now we show a class of PDFA with exponentially small prefix-distinguishability that can be efficiently learned with Statistical Queries. For every $h \in \{-1, 0, +1\}^n$ define the *conjunction* $C_h : \{0, 1\} \to \{0, 1\}$ as $C_h(x) = \bigwedge_{i\in[n]} x_i^{h_i}$, with the conventions $x_i^0 = 1, 0^{-1} = 1$ and $1^{-1} = 0$. We construct the class of distributions $\mathcal{C}_n$ following the same schema used for defining $\mathcal{D}_n$, but using conjunctions instead of parities. Note that each $D_h \in \mathcal{C}_n$ can be represented with a PDFA of $\Theta(n)$ states and prefix-distinguishability $\Theta(1/2^n)$. However, it is well known that the class of all conjunctions over n variables can be efficiently learned using Statistical Queries, and a trivial reduction shows that $\mathcal{C}_n$ can also be learned using poly(n) Statistical Queries with tolerance poly($1/n$). That is, our lower bound of $\Omega(1/\mu^{1-c})$ for every $c > 0$ applies when trying to learn the class $\mathcal{PDFA}_{n,\mu}$ of all PDFA with n states and distinguishability μ, but not when trying to learn an arbitrary subclass of $\mathcal{PDFA}_{n,\mu}$.

Section 3 introduced our new variant of Statistical Queries, called L_∞-queries, and proved that they can be simulated using standard Statistical Queries. From there we observed that all known variants of CT for PAC learning PDFA can be rewritten using Statistical Queries. Query formalisms sometimes provide tools for reasoning about algorithms that have no equivalence for "unrestricted" algorithms. In our case, in Section 4 we were able to prove non-trivial lower bounds on the number of L_∞-queries that any algorithm learning the class $\mathcal{PDFA}$ must make in terms of parameters in the queries as well as in terms of the distinguishability. These lower bounds show that PDFA learning is hard in the worst case, at least for all algorithms that use merging techniques based on L_∞ tests. It is worth remarking that similar arguments could ostensibly be used to prove lower bounds for a hypothetical learning model defined in terms of prefL_∞-queries. Furthermore, worst case lower bounds also apply to any PDFA learning algorithm that can be rewritten to use Statistical Queries; these include many known algorithms that do not fall inside the L_∞-query formalism, but learn by collecting other statistical information from the sample like [18,17].

We observe that our simulation of L_∞-queries using Statistical Queries is not completely straightforward. At this point, it is unclear whether a converse to Theorem 5 holds. Namely, we ask whether Statistical Queries can be efficiently simulated using L_∞-queries. We conjecture that this is not possible. We suspect that studying the learnability of the class $\mathcal{C}_n$ defined in the examples above may shed some light onto this problem.

Our main conclusion is that, although PDFA learning is provably difficult in the worst case, efficient learning is feasible in many, hopefully interesting, cases. Furthermore, the current notions of complexity — distinguishability and prefix-distinguishability — are too coarse for measuring the *true* complexity of PDFA learning, as shown in the examples above.

Ideally, one would like a measure of complexity, say η, that provably characterizes the complexity of learning any subclass of $\mathcal{PDFA}$. Namely, if $\mathcal{D} \subseteq \mathcal{PDFA}$ is any large enough subclass, we would be interested in a result saying that any algorithm that learns $\mathcal{D}$ uses at least $\Omega(\eta_{\mathcal{D}})$ examples (or queries), where $\eta_{\mathcal{D}}$ measures the complexity of the class $\mathcal{D}$. A possible line of work in this direction would be to adapt one of the several notions of *dimension* used in concept learning to distribution learning. Furthermore, finding learning algorithms that provably match this hypothetical lower bound in terms of η would be another interesting, practically relevant problem.

The adaptive algorithm AdaCT described in Section 5 represents a significant improvement in the state of the art on PDFA learning algorithms with strong theoretical guarantees. First, the dependence of the bound in Theorem 25 on the parameters of the problem is better than in previous algorithms. Second, the use of prefix-distinguishability for learning general PDFA yields an algorithm that can learn large subclasses of $\mathcal{PDFA}$ faster than algorithms based on distinguishability. Third, the adaptive nature of AdaCT and its small number of input parameters make it a truly practical algorithm. Future work may be directed towards removing the necessity of having n as an input parameter. Note that the main obstruction in this direction in AdaCT is to provide a sensible stopping condition for an algorithm that does not know n.

Acknowledgments

The authors would like to thank the reviewers for many useful comments. This work is partially supported by the Spanish Ministry of Science and Technology contracts TIN-2008-06582-C03-01 (SESAAME) and TIN-2007-66523 (FORMALISM), by the Generalitat de Catalunya 2009-SGR-1428 (LARCA), and by the EU PASCAL2 Network of Excellence (FP7-ICT-216886). B. Balle is supported by an FPU fellowship (AP2008-02064) from the Spanish Ministry of Education.

References

[1] J. Castro, R. Gavaldà, Towards feasible pac-learning of probabilistic deterministic finite automata, in: ICGI ’08: Proceedings of the 9th International Colloquium on Grammatical Inference, Springer-Verlag, Berlin, Heidelberg, ISBN: 978-3-540-88008-0, 2008, pp. 163–174. http://dx.doi.org/10.1007/978-3-540-88009-7_13.

[2] B. Balle, J. Castro, R. Gavaldà, A lower bound for learning distributions generated by probabilistic automata, in: M. Hutter, F. Stephan, V. Vovk, T. Zeugmann (Eds.), Algorithmic Learning Theory, in: Lecture Notes in Computer Science, vol. 6331, Springer, Berlin/Heidelberg, 2010, pp. 179–193. http://dx.doi.org/10.1007/978-3-642-16108-7_17. URL.

[3] P. Dupont, F. Denis, Y. Esposito, Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms, Pattern Recognit. 38 (9) (2005) 1349–1371.

[4] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R. C. Carrasco, Probabilistic Finite-State Machines - Part I, IEEE Trans. Pattern Anal. Mach. Intell. 27 (7) (2005) 1013–1025.

[5] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R.C. Carrasco, Probabilistic finite-state machines - Part II, IEEE Trans. Pattern Anal. Mach. Intell. 27 (7) (2005) 1026–1039.

[6] A. Clark, F. Thollard, PAC-learnability of probabilistic deterministic finite state automata, J. Mach. Learn. Res..

[7] F. Denis, Y. Esposito, A. Habrard, Learning Rational Stochastic Languages, in: G. Lugosi, H.-U. Simon (Eds.), COLT, in: Lecture Notes in Computer Science, vol. 4005, Springer, ISBN: 3-540-35294-5, 2006, pp. 274–288.

[8] N. Abe, M.K. Warmuth, On the computational complexity of approximating distributions by probabilistic automata, Mach. Learn. (ISSN: 0885-6125) 9 (2–3) (1992) 205–260. http://dx.doi.org/10.1007/BF00992677.

[9] M.J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire, L. Sellie, On the learnability of discrete distributions, in: STOC, 1994, pp. 273–282.

[10] D. Ron, Y. Singer, N. Tishby, On the learnability and usage of acyclic probabilistic finite automata, J. Comput. Syst. Sci. 56 (2) (1998) 133–152.

[11] O. Guttman, S.V.N. Vishwanathan, R.C. Williamson, Learnability of Probabilistic Automata via Oracles, in: S. Jain, H.-U. Simon, E. Tomita (Eds.), ALT, in: Lecture Notes in Computer Science, vol. 3734, Springer, ISBN: 3-540-29242-X, 2005, pp. 171–182.

[12] N. Palmer, P.W. Goldberg, PAC-learnability of probabilistic deterministic finite state automata in terms of variation distance, Theoret. Comput. Sci. (ISSN: 0304-3975) 387 (1) (2007) 18–31. http://dx.doi.org/10.1016/j.tcs.2007.07.023.

[13] R. Gavaldà, P.W. Keller, J. Pineau, D. Precup, PAC-learning of Markov models with hidden state, in: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), ECML, in: Lecture Notes in Computer Science, vol. 4212, Springer, ISBN: 3-540-45375-X, 2006, pp. 150–161.

[14] F. Thollard, A. Clark, Learning stochastic deterministic regular languages, in: Grammatical Inference: Algorithms and Applications, in: Lecture Notes in Computer Science, vol. 3264, Springer, Berlin/Heidelberg, 2004, pp. 248–259.

[15] R.C. Carrasco, J. Oncina, Learning stochastic regular grammars by means of a state merging method, in: R.C. Carrasco, J. Oncina (Eds.), ICGI, in: Lecture Notes in Computer Science, vol. 862, Springer, ISBN: 3-540-58473-0, 1994, pp. 139–152.

[16] R.C. Carrasco, J. Oncina, Learning deterministic regular grammars from stochastic samples in polynomial time, RAIRO Theor. Inform. Appl. 33 (1) (1999) 1–20.

[17] F. Thollard, P. Dupont, C.d.l. Higuera, Probabilistic DFA inference using Kullback–Leibler divergence and minimality, in: Proceedings of the Seventeenth International Conference on Machine Learning, in: ICML’00, Morgan Kaufmann Publishers Inc., 2000, pp. 975–982.

[18] D. Hsu, S.M. Kakade, T. Zhang, A spectral algorithm for learning Hidden Markov models, CoRR abs/0811.4413.

[19] D. Pfau, N. Bartlett, F. Wood, Probabilistic deterministic infinite automata, in: J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R. Zemel, A. Culotta (Eds.), Advances in Neural Information Processing Systems 23, 2010, pp. 1930–1938.

[20] M. Kearns, Efficient noise-tolerant learning from statistical queries, J. ACM (ISSN: 0004-5411) 45 (6) (1998) 983–1006 http://doi.acm.org/10.1145/293347.293351.

[21] J.A. Aslam, S.E. Decatur, Specification and simulation of statistical query algorithms for efficiency and noise tolerance, J. Comput. Syst. Sci. (ISSN: 0022-0000) 56 (1998) 191–208. http://dx.doi.org/10.1006/jcss.1997.1558. URL http://portal.acm.org/citation.cfm?id=291953.291957.

[22] S. Hoory, N. Linial, A. Wigderson, Expander graphs and their applications, Bull. Amer. Math. Soc. (N.S.) (ISSN: 0273-0979) 43 (4) (2006) 439.

[23] E. Kushilevitz, Y. Mansour, Learning Decision Trees Using the Fourier Spectrum, SIAM J. Comput. 22 (6) (1993) 1331–1348 http://dx.doi.org/10.1137/0222080. URL http://link.aip.org/link/?SMJ/22/1331/1.

[24] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, New York, NY, USA, ISBN: 0521841089, 2006.