PAC-learnability of Probabilistic Deterministic Finite State Automata in terms of Variation Distance ⋆

Nick Palmer [a] Paul W. Goldberg [b,∗]

[a] *Dept. of Computer Science, University of Warwick, Coventry CV4 7AL, U.K.*

[b] *Dept. of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, U.K.*

Abstract

We consider the problem of PAC-learning distributions over strings, represented by probabilistic deterministic finite automata (PDFAs). PDFAs are a probabilistic model for the generation of strings of symbols, that have been used in the context of speech and handwriting recognition, and bioinformatics. Recent work on learning PDFAs from random examples has used the KL-divergence as the error measure; here we use the variation distance. We build on recent work by Clark and Thollard, and show that the use of the variation distance allows simplifications to be made to the algorithms, and also a strengthening of the results; in particular that using the variation distance, we obtain polynomial sample size bounds that are independent of the expected length of strings.

Key words: Computational complexity, machine learning

1 Introduction

A probabilistic deterministic finite automaton (PDFA) is a finite automaton that has, for each state, a probability distribution over the transitions going

⋆ This work was supported by EPSRC Grant GR/R86188/01. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

∗ Corresponding author.
Email addresses: `npalmer@dcs.warwick.ac.uk` (Nick Palmer), `P.W.Goldberg@liverpool.ac.uk` (Paul W. Goldberg).

out from that state. Each transition emits a symbol from a finite alphabet, and the automaton is deterministic in that at most one transition with a given symbol is possible from any state. Thus, a PDFA defines a probability distribution over the set of strings over its alphabet.

PDFAs are just one of a variety of structures used to model stochastic processes in fields such as AI and machine learning. Similar structures seen in related work include

- Probabilistic nondeterministic finite automata (PNFA),
- Hidden Markov models (HMM), and
- Partially observable Markov decision processes (POMDP).

A PNFA is similar to a PDFA, but whereas a PDFA may have at most one transition with a given symbol leaving a state, a PNFA may have more than one transition emitting the same symbol. Thus even with knowledge of the starting state and the symbol generated by a transition from this state, the machine may be in one of several states. This model has more expressive power, and consequently it is harder to obtain positive results for learning.

In a HMM, each state has a probability distribution over symbols, and a symbol is emitted when that state is visited. HMMs and PNFAs have essentially the same expressive power [7]. Abe and Warmuth [2] give a strong computational negative result for learning PNFAs and HMMs, namely that is it hard to maximise the likelihood of an individual string using these models. This holds for a fixed number of states, non-fixed alphabet.

POMDPs are associated with online learning problems, where choices can be made by the learner as data is analysed. There is an underlying probabilistic finite automaton whose states are not directly observable. A POMDP takes actions as input from the learner, where an observation is output and a reward is awarded to the learner (at each step, the reward depends on the transition taken and the learner's action). The objective in these learning problems is to maximise some function of the rewards. A POMDP is an extension of the notion of a Markov decision process to situations where the state is not always known to the algorithm.

Positive results for PAC-learning[1] sub-classes of PDFAs were introduced by Ron et al. [13], where they show how to PAC-learn *acyclic* PDFAs, and apply the algorithm to speech and handwriting recognition. Recently Clark and Thollard [4] presented an algorithm that PAC-learns general PDFAs, using the Kullback-Leibler divergence (see Cover and Thomas [5]) as the error measure (the distance between the true distribution defined by the target PDFA, and the hypothesis returned by the algorithm). The algorithm is polynomial in

[1] PAC (probably approximately correct) learnability is defined in Section 2.

three parameters: the number of states, the "distinguishability" of states, and the expected length of strings generated from any state of the target PDFA. Distinguishability (defined in Section 3) is a measure of the extent to which any pair of states have an associated string that is significantly more likely to be generated from one state than the other. While unrestricted PDFAs can encode noisy parity functions [10] (believed to be hard to PAC-learn), these PDFAs have "exponentially low" distinguishability.

In this paper we study the same problem, using variation distance instead of Kullback-Leibler divergence. The general message of this paper is that this modification allows some strengthening and simplifications of the resulting algorithms. The main one is that—as conjectured in [4]—a polynomial bound on the sample-size requirement is obtained that does not depend on the length of strings generated by the automaton. We also have no need for a distinguished "final symbol" that must terminate all data strings, or a "ground state" in the automaton constructed by the algorithm. We have also simplified the algorithm by not re-sampling at each iteration; instead we use the same sample in all iterations.

The variation distance between probability distributions D and D' is the L_1 distance; for a discrete domain X, it is $L_1(D, D') = \sum_{x \in X} |D(x) - D'(x)|$. KL divergence is in a strong sense a more "sensitive" measure than variation distance – this was pointed out in Kearns et al. [10], which introduced the general topic of PAC-learning probability distributions. In Cryan et al. [6] a smoothing technique is given for distributions over the boolean domain (where the length of strings is a parameter of the problem)—an algorithm that PAC-learns distributions using the variation distance can be converted to an algorithm that PAC-learns using the KL-divergence. (Abe et al. [1] give a similar result in the context of learning p-concepts.) Over the domain Σ^* (strings of unrestricted length over alphabet Σ) that technique does not apply, which is why we might expect stronger results as a result of switching to the variation distance.

In the context of pattern classification, the variation distance is useful in the following sense. Suppose that we seek to classify labelled data by fitting distributions to each label class, and using the Bayes classifier on the hypothesis distributions. (See [8] for a discussion of the motivation for this general approach, and results for PAC-learnability.) We show in [12] that PAC-learnability using the variation distance implies agnostic PAC classification. The corresponding result for KL-divergence is that the expected negative log-likelihood cost is close to optimum.

Our approach follows [4], in that we divide the algorithm into two parts. The first (Algorithm 1 of Figure 1) finds a DFA that represents the structure of the hypothesis automaton, and the second (Algorithm 2 of Figure 2) finds estimates of the transition probabilities. Algorithm 1 constructs (with high probability) a DFA whose states and transitions are a subset of those of the target. Algorithm 2 learns the transition probabilities by following the paths of random strings through the DFA constructed by Algorithm 1. We take advantage of the fact that commonly-used transitions can be estimated more precisely.

2 Terms and Definitions

A probabilistic deterministic finite state automaton (PDFA) can stochastically generate strings of symbols as follows. The automaton has a finite set of states – one of which is distinguished as the *initial state*. The automaton generates a string by making transitions between states (starting at the initial state), each transition occurring with a constant probability specifically associated with that transition. The symbol labelling that transition is then output. The automaton halts when the *final state* is reached.

Definition 1 *A PDFA A is a sextuple $(Q, \Sigma, q_0, q_f, \tau, \gamma)$, where*

- *Q is a finite set of states,*
- *Σ is a finite set of symbols (the alphabet),*
- *$q_0 \in Q$ is the initial state,*
- *$q_f \notin Q$ is the final state,*
- *$\tau : Q \times \Sigma \to Q \cup \{q_f\}$ is the (partial) transition function,*
- *$\gamma : Q \times \Sigma \to [0, 1]$ is the function representing the probability of a given symbol (and the corresponding transition) occurring from a given state.*

It is required that $\sum_{\sigma\in\Sigma} \gamma(q, \sigma) = 1$ for all $q \in Q$, and when $\tau(q, \sigma)$ is undefined, $\gamma(q, \sigma) = 0$. In addition q_f is reachable from any state of the automaton, that is, for all $q \in Q$ there exists $s \in \Sigma^*$ such that $\tau(q, s) = q_f \wedge \gamma(q, s) > 0$.

It is common for a definition of a PDFA to include the specification of a final symbol at the end of all words; we do not require that restriction here. Where appropriate, we extend the use of τ and γ to strings:

$$\tau(q, \sigma_1\sigma_2...\sigma_k) = \tau(\tau(q, \sigma_1), \sigma_2...\sigma_k)$$
$$\gamma(q, \sigma_1\sigma_2...\sigma_k) = \gamma(q, \sigma_1).\gamma(\tau(q, \sigma_1), \sigma_2...\sigma_k)$$

If A denotes a PDFA, it follows that A defines a probability distribution over strings in Σ^*. (The reachability of the final state ensures that A will halt with probability 1.) Let $D_A(s)$ denote the probability that A generates $s \in \Sigma^*$, so we have

$$D_A(s) \;=\; \gamma(q_0, s) \text{ for } s \text{ such that } \tau(q_0, s) = q_f.$$

We use the pair (q, σ) to denote the transition from state $q \in Q$ labelled with character $\sigma \in \Sigma$. Let $D_A(q)$ denote the probability that a random string generated by A uses state $q \in Q$. Thus $D_A(q)$ is the probability that $s \sim D_A$ (i.e. s sampled from distribution D_A) has a prefix p with $\tau(q_0, p) = q$. In a similar way, $D_A(q, \sigma)$ denotes the probability that a random string generated by A uses transition (q, σ)—the probability that a random string $s \sim D_A$ has a prefix $p\sigma$ with $\tau(q_0, p) = q$.

Suppose D and D' are probability distributions over Σ^*. The variation (L_1) distance between D and D' is $L_1(D, D') = \sum_{s \in \Sigma^*} |D(s) - D'(s)|$. A class $\mathcal{C}$ of probability distributions is PAC-learnable by algorithm $\mathcal{A}$ with respect to the variation distance if the following holds. Given parameters $\epsilon > 0$, $\delta > 0$, and access to samples from any $D \in \mathcal{C}$, using runtime and sample size polynomial in ϵ^{-1} and δ^{-1}, $\mathcal{A}$ should, with probability $1 - \delta$, output a distribution D' with $L_1(D, D') < \epsilon$. If $D \in \mathcal{C}$ is described in terms of additional parameters that represent the complexity of D, then we require $\mathcal{A}$ to be polynomial in these parameters as well as ϵ^{-1} and δ^{-1}.

3 Constructing the PDFA

In this section we describe the first part of the algorithm, which constructs the underlying DFA of a target PDFA A. That is, it constructs the states Q and transitions given by τ, but not the probabilities given by γ. The algorithms has access to a source of strings in Σ^* generated by D_A. We allow "very unlikely" states to be ignored, as described at the end of this section where we explain how our algorithm differs from previous related algorithms. Properties of the constructed DFA are proved in Section 4.

The algorithm is shown in Figure 1. We have the following parameters (in addition to the PAC parameters ϵ and δ):

- $|\Sigma|$: the alphabet size,
- n: an upper bound on the number of states of the target automaton,
- μ: a lower bound on distinguishability, defined below.

In the context of learning using the KL-divergence, a simple class of PDFAs (see Clark and Thollard [4]) can be constructed to show that the parameters above are insufficient for PAC learnability in terms of just those parameters. In [4], parameter L is also used, denoting the expected length of strings.

From the target automaton A we generate a hypothesis automaton H using a variation on the method described by [4] utilising *candidate nodes*, where the L_∞ norm between the suffix distributions of states is used to distinguish between them (as studied also in [9,13]). We define a candidate node in the same way as [4]. Suppose G is a graph whose vertices correspond to a subset of the states of A, and whose edges correspond to transitions. Initially G will have a single vertex corresponding to the initial state; G is then constructed in a greedy incremental fashion.

$G = \langle V, E \rangle$ denotes the directed graph constructed by the algorithm. V is the set of vertices and E the set of edges. Each edge is labelled with a letter $\sigma \in \Sigma$, so an edge is a member of $V \times \Sigma \times V$. Note that due to the deterministic nature of the automaton, there can be at most one vertex v_q such that $(v_p, \sigma, v_q) \in E$ for any $v_p \in V$ and $\sigma \in \Sigma$.

Definition 2 *A candidate node in hypothesis graph G is a pair (u, σ) (also denoted $\hat{q}_{u,\sigma}$), where u is a node in the graph and $\sigma \in \Sigma$ where $\tau_G(u, \sigma)$ is undefined.*

Let D_q denote the distribution over strings generated using state q as the initial state, so that

$$D_q(s) = \gamma(q, s) \text{ for } s \text{ such that } \tau(q, s) = q_f.$$

Given a sample S of strings generated from D_A, we define a multiset associated with each node or candidate node in a hypothesis graph. The multiset for node q is an i.i.d. sample from D_q, derived from S, obtained by taking members of S that use q and deleting their prefixes that reach q for the first time in a string. For a candidate node, we use the following definition.

Definition 3 *Given a sample S, candidate node $\hat{q}_{u,\sigma}$ has multiset $S_{u,\sigma}$ associated with it, where for each $s \in S$, we add s'' to $S_{u,\sigma}$ whenever $s = s'\sigma s''$ and $\tau_G(q_0, s') = u$.*

The L_∞-norm is a measure of distance between a pair of distributions, defined as follows.

Definition 4 $L_\infty(D, D') = \max_{s \in \Sigma^*} |D(s) - D'(s)|$.

Definition 5 *The parameter of distinguishability, μ, is a lower bound on the L_∞-norm between D_{q_1} and D_{q_2} for any pair of nodes (q_1, q_2), where q_1 and q_2 are regarded as having sufficiently different suffix distributions in order to be considered separate states.*

We define as follows the $\hat{L}_\infty$-norm (an empirical version of the L_∞-norm) with respect to multisets of strings S_{q_1} and S_{q_2}, where S_{q_1} and S_{q_2} have been respectively sampled from D_{q_1} and D_{q_2}.

Definition 6 *For nodes q_1 and q_2, with associated multisets S_{q_1} and S_{q_2},*

$$\hat{L}_\infty\left(D_{q_1}, D_{q_2}\right) = \max_{s\in\Sigma^*}\left(\left|\frac{|s \in S_{q_1}|}{|S_{q_1}|} - \frac{|s \in S_{q_2}|}{|S_{q_2}|}\right|\right)$$

where D_q is the empirical distribution over the strings in the multiset S_q associated with q, and where $|s \in S_q|$ is the number of occurrences of string s in multiset S.

As in [13,4], we say that a pair of nodes (q_1, q_2) are *μ-distinguishable* if $L_\infty(D_{q_1}, D_{q_2}) = \max_{s\in\Sigma^*} |D_{q_1}(s) - D_{q_2}(s)| \geq \mu$.

The algorithm uses two quantities, m_0 and N. m_0 is the number of suffixes required in the multiset of a candidate node for the node to be added as a state (or as a transition) to the hypothesis. It will be shown that m_0 is a sufficiently large number to allow us to establish that the distribution over suffixes in the multiset that begin at state q is likely to approximate the true distribution D_q over suffixes at that state. N is the number of (i.i.d.) strings in the sample generated by the algorithm. Polynomial expressions for m_0 and N are given in Algorithm 1.

We show that the probability of Algorithm 1 failing to adequately learn the structure of the automaton is upper bounded by δ'. In Section 5 we show that the transition probabilities are learned (with sufficient accuracy for our purposes) by Algorithm 2 with a failure probability of at most δ''. Overall, the probability of the algorithms failing to learn the target PDFA within a variation distance of ϵ is at most δ, for $\delta = \delta' + \delta''$.

Algorithm 1 differs from [4] as follows. We do not introduce a "ground node" – a node to catch any undefined transitions in the hypothesis graph so as to give a probability greater than zero to the generation of any string. Instead, any state q for which $D_A(q) < \frac{\epsilon}{2n|\Sigma|}$ can be discarded – no corresponding node is formed in our hypothesis graph. There is only a small probability that our hypothesis automaton rejects a random string generated by D_A (when there is no corresponding path through the graph), which means that the contribution to the overall variation distance is very small. This is in contrast to the KL distance, which would become infinite.

Note that in contrast to the previous version of this algorithm in [11], and the algorithm of [4], we make a single sample at the beginning of the algorithm and we use the whole sample at each iteration. The trade-off is that by re-using the same sample at each iteration, we need a much lower failure probability (or higher reliability). It turns out that the total sample-size is about the same, but the algorithm is simpler and corresponds with the natural way one would treat real-world data.

Algorithm 1

```
Hypothesis Graph $G$ = $\langle V, E\rangle$ = $\langle \{q_0\}, \emptyset\rangle$;
$m_0 = (16/\mu)^2(\log(16/\delta'\mu) + \log(n|\Sigma|) + n|\Sigma|)$;
$N = \max\left(\frac{8n^2|\Sigma|^2}{\epsilon^2}\ln\left(\frac{2^{n|\Sigma|}n|\Sigma|}{\delta'}\right), \frac{4m_0n|\Sigma|}{\epsilon}\right)$;
generate a sample $S$ of $N$ strings iid from $D_A$;
repeat
  for (each $v \in V$, $\sigma \in \Sigma$, where $\tau_G(v,\sigma)$ is undefined)
    create a candidate node $\bar{q}_{v,\sigma}$ with associated
    multiset $S_{v,\sigma} = \emptyset$;
  for (each string $s \in S$, where $s = r\sigma' t$ and $\bar{q}_{\tau_G(q_0,r),\sigma'}$ is
  a candidate node)
    $S_{\tau(q_0,r),\sigma'} \leftarrow S_{\tau(q_0,r),\sigma'} \cup \{t\}$;
  identify candidate node $\bar{q}_{u,\sigma''}$ with the largest multiset, $S_{u,\sigma''}$;
  if ($|S_{u,\sigma''}| \geq m_0$) % candidate node has large enough multiset
    if $\left(\exists v \in V : \hat{L}_\infty\left(D_{\bar{q}_{u,\sigma''}}, D_v\right) \leq \frac{\mu}{2}\right)$
    % candidate "looks like" existing node
      add edge $(u, \sigma'', v)$ to $E$;
    else
      add node $\bar{q}_{u,\sigma''}$ to $V$, with multiset $S_{u,\sigma''}$;
      add edge $(u, \sigma'', \bar{q}_{u,\sigma''})$ to $E$;
until($|S_{u,\sigma''}| < m_0$); % no candidate node has large enough multiset
return $G$.
```

Fig. 1. Constructing the underlying graph

4 Analysis of PDFA Construction Algorithm

The initial state $\bar{q}_0$ of H corresponds to the initial state q_0 of A. Each time a new state $\bar{q}_{v,\sigma}$ is added to H, its corresponding state in A is (with high probability) $\tau(q_v, \sigma)$. (Note that q_v in H already has a corresponding state in A.) Of course, we will argue that the correspondence is 1-1, and that we reproduce a subgraph of A. We claim that at every iteration of the algorithm, with high probability a bijection Φ exists between the states of H and candidate states, and a subset of the states of A, such that $\tau_A(u,\sigma) = v \Leftrightarrow \tau_H(\Phi(u),\sigma) = \Phi(v)$.

We start by showing that with high probability, candidate states are correctly identified as being either unseen so far, or the same as a pre-existing state in the hypothesis. This part exploits the distinguishability assumption.

Proposition 7 *Let D be a distribution over a countable domain. Let δ and μ be positive probabilities. Suppose we draw a sample S of $(16/\mu)^2\log(16/\delta\mu)$ observations of D. Let $\hat{D}$ be the empirical distribution, i.e. the uniform distri-*

bution over multiset S. Then with probability $1-\delta$, $L_\infty(D,\hat{D}) < \frac{1}{4}\mu$.

Proof: Let $X = \{x_1, x_2, \ldots\}$ be the domain. Associate x_i with the interval $I_i = [\sum_{j<i}\Pr(x_j), \sum_{j\leq i}\Pr(x_j)]$. Let U_1 denote the uniform distribution over the unit interval; a point drawn from U_1 selects x_i with probability $\Pr(x_i)$.

Suppose $k \in \mathbf{N}$, $k \leq 16/\mu$. We identify a sufficiently large size for a sample S from U_1 such that with probability at least $1-(\delta\mu/16)$, the proportion of points in S that lie in $[0, k(\mu/16)]$, is within $\mu/16$ of $k(\mu/16)$. By Hoeffding's inequality it is sufficient that $m = |S|$ satisfies

$$\frac{\delta\mu}{16} \geq 2\exp\Big(-2m\Big(\frac{\mu}{16}\Big)^2\Big).$$

That is satisfied by

$$m \geq \Big(\frac{16}{\mu}\Big)^2 \log\Big(\frac{16}{\delta\mu}\Big).$$

Furthermore, by a union bound we can deduce that with probability at least $1-\delta$, for all $k \in \{0, 1, \ldots, 16/\mu\}$, the proportion of points in $[0, k(\mu/16)]$ is within $\mu/16$ of expected value. This implies that for all intervals, including the I_i intervals, the proportion of points in those intervals is within $\mu/4$ of expected value. $\square$

The following result shows that given any partially-constructed DFA, a candidate state is correctly identified with very high probability, using a sample of size $m_0 = (16/\mu)^2(\log(16/\delta'\mu) + \log(n|\Sigma|) + n|\Sigma|)$.

Proposition 8 *Let G be a DFA with transition function τ_G whose vertices and edges are a subgraph of the underlying DFA for PDFA A. Suppose D_A is repeatedly sampled, and we add s_2 to $S_{q,\sigma}$ whenever we obtain a string of the form $s_1\sigma s_2$, where $\tau_G(s_1)$ is state q of G.*

If $|S_{q,\sigma}| \geq m_0$, then with probability at least $1-\delta'(n|\Sigma|2^{n|\Sigma|})^{-1}$, $\hat{L}_\infty(S_{q,\sigma}, D_{q,\sigma}) < \mu/4$.

Proof: Given any G, strings s_2 obtained in this way are all sampled independently from $D_{q,\sigma}$.

Proposition 7 shows that a sufficiently large sample size is given by

$$\Big(\frac{16}{\mu}\Big)^2 \log\Big(\frac{16}{\delta'\mu(n|\Sigma|2^{n|\Sigma|})^{-1}}\Big) = \Big(\frac{16}{\mu}\Big)^2\Big(\log\Big(\frac{16}{\delta'\mu}\Big) + \log(n|\Sigma|) + n|\Sigma|\Big).$$

$\square$

The following result applies a union bound to verify that whatever stage we reach at an iteration, and whatever candidate state we examine, the algorithm is unlikely to make a mistake.

Proposition 9 *With probability δ', for all candidate nodes $\bar{q}_{u,\sigma''}$ found by the algorithm, $\bar{q}_{u,\sigma''}$ is added to G such that G continues to be a subgraph of the PDFA for A.*

Proof: Proposition 7 and the metric property of L_∞ show that if distributions D_v associated with states v are empirically estimated to within L_∞ distance $\mu/4$, then our threshold of $\mu/2$ that is used to distinguish a pair of states, ensures that no mistake is made.

There are at most $2^{n|\Sigma|}$ possible subgraphs G and at most $n|\Sigma|$ candidate nodes for any subgraph. If the probability of failure is at most $\delta'(n|\Sigma|2^{n|\Sigma|})^{-1}$ for any single combination of G and candidate node, then by a union bound and Proposition 8, the probability of failure is at most δ'. □

We have ensured that m_0 is large enough that with high probability the algorithm does not

- identify two distinct nodes with each other, or
- fail to recognize a candidate node as having been seen already.

Next we have to check that it does not "give up too soon", as a result of not seeing m_0 samples from a state that really should be included in G.

Proposition 10 *Let A' be a PDFA whose states and transitions are a subset of those of A. Assume A' contains the initial state q_0. Suppose q is a state of A' but (q,σ) is not a transition of A'. Let S be a sample from D_A, $|S| \geq (8n^2|\Sigma|^2/\epsilon^2)\ln(2^{n|\Sigma|}n|\Sigma|/\delta')$. Let $S_{q,\sigma}(A')$ be the number of elements of S of the form $s_1\sigma s_2$ where $\tau(q_0,s_1)=q$ and for all prefixes s_1' of s_1, $\tau(q_0,s_1')\in A'$. Then*

$$\Pr\left(\left|\left(\frac{S_{q,\sigma}(A')}{|S|}\right) - E\left[\frac{S_{q,\sigma}(A')}{|S|}\right]\right| \geq \frac{\epsilon}{8n|\Sigma|}\right) \leq \frac{\delta'}{2^{n|\Sigma|}n|\Sigma|}.$$

Proof: From Hoeffding's Inequality it can be seen that

$$\Pr\left(\left|\left(\frac{S_{q,\sigma}(A')}{|S|}\right) - E\left[\frac{S_{q,\sigma}(A')}{|S|}\right]\right| \geq \frac{\epsilon}{8n|\Sigma|}\right) \leq 2\exp\left(-2|S|\left(\frac{\epsilon}{4n|\Sigma|}\right)^2\right). \tag{1}$$

We need $|S|$ to satisfy $\exp(-|S|\epsilon^2(8n^2|\Sigma|^2)^{-1}) \leq \delta'(2^{n|\Sigma|}n|\Sigma|)^{-1}$. Equivalently,

$$\frac{8n^2|\Sigma|^2}{\epsilon^2}\ln\left(\frac{2^{n|\Sigma|}n|\Sigma|}{\delta'}\right) \leq |S|.$$

So the sample size identified in the statement is indeed sufficiently large. □

The following result shows that the algorithm constructs a subset of the states and transitions that with high probability accepts a random string from D_A.

Theorem 11 *There exists T' a subset of the transitions of A, and Q' a subset of the states of A, such that $\sum_{(q,\sigma)\in T'} D_A(q,\sigma) + \sum_{q\in Q'} D_A(q) \le \frac{\epsilon}{2}$, and with probability at least $1-\delta'$, every transition $(q,\sigma) \notin T'$ in target automaton A has a corresponding transition in hypothesis automaton H, and every state $q \notin Q'$ in target automaton A has a corresponding state in hypothesis automaton H.*

Proof: Proposition 9 shows that the probability of all candidate nodes having "good" multisets (if the multisets contain at least m_0 suffixes) is at least $1 - \delta'/2$, from which we can deduce that all candidate nodes can be correctly distinguished from any nodes [2] in the hypothesis automaton.

Proposition 10 shows that with a probability of at least $1 - \delta'(2^{n|\Sigma|}n|\Sigma|)^{-1}$, the proportion of strings in a sample S (generated i.i.d. over D_A, and for $|S| \ge (8n^2|\Sigma|^2/\epsilon^2)\ln(2^{n|\Sigma|}n|\Sigma|/\delta'))$ reaching candidate node $\bar{q}$ is within $\epsilon(8n|\Sigma|)^{-1}$ of the expected proportion $D_A(\bar{q})$. This holds for each of the candidate nodes (of which there are at most $n|\Sigma|$), and for each possible state of the hypothesis graph in terms of the combination of edges and nodes found (of which there are at most $2^{n|\Sigma|}$), with a probability of at least $1 - \delta'/2$.

If a candidate node (or a *potential candidate node* [3]) $\bar{q}$, for which $D_A(\bar{q}) \ge \epsilon(2n|\Sigma|)^{-1}$, is not included in H, then from the facts above it follows that at least $\epsilon N(4n|\Sigma|)^{-1}$ strings in the sample are not accepted by the hypothesis graph. For each string not accepted by H, a suffix is added to the multiset of a candidate node, and there are at most $n|\Sigma|$ such candidate nodes. From this it can be seen that some candidate node has a multiset containing at least $\frac{1}{4}\epsilon N$ suffixes. From the definition of N, $N \ge (4m_0n|\Sigma|/\epsilon)$. Therefore, some multiset contains at least $m_0n|\Sigma|$ suffixes, which must be at least as great as m_0. This means that as long as there exists some significant transition or state that has not been added to the hypothesis, some multiset must contain at least m_0 suffixes, so the associated candidate node will be added to H, and the algorithm will not halt.

Therefore it has been shown that all candidate nodes which are significant enough to be required in the hypothesis automaton (at least a fraction $\epsilon(2n|\Sigma|)^{-1}$ of the strings generated reach the node) are present with a probability of at

[2] Note that due to the deterministic nature of the automaton, distinguishability of transitions is not an issue.

[3] A potential candidate node is any state or transition in the target automaton which has not yet been added to H, and is not currently represented by a candidate node.

least $1 - \frac{1}{2}\delta'$, and that since all multisets contain at least m_0 suffixes, the candidate nodes and hypothesis graph nodes are all correctly distinguished from each other (or combined as appropriate) with a probability of at least $1 - \frac{1}{2}\delta'/2$.

T' is those transitions that have probability less than $\epsilon/2n|\Sigma|$ of being used by a random string, and there can be at most $n|\Sigma|$ such transitions. Hence a random string uses an element of T' with probability at most $\frac{1}{2}\epsilon$. We conclude that with a probability of at least $1-\delta'$, every transition $(q,\sigma) \notin T'$ in target automaton A for which $D_A(q,\sigma) \geq \epsilon(2n|\Sigma|)^{-1}$ and every state $q \notin Q'$ in target automaton A for which $D_A(q) \geq \epsilon(2n|\Sigma|)^{-1}$, has a corresponding transition or state in hypothesis automaton H. □

5 Finding Transition Probabilities

The algorithm is shown in Figure 2. We can assume that we have at this stage found DFA H, whose graph is a subgraph of the graph of target PDFA A. Algorithm 2 finds estimates of the probabilities $\gamma(q,\sigma)$ for each state q in H, $\sigma \in \Sigma$.

If we generate a sample S from D_A, we can trace each $s \in S$ through H, and each visit to a state $q_H \in H$ provides an observation of the distribution over the transitions that leave the corresponding state q_A in A. For string $s = \sigma_1\sigma_2\ldots\sigma_\ell$, let q_i be the state reached by the prefix $\sigma_1\ldots\sigma_{i-1}$. The probability of s is $D_A(s) = \prod_{i=0}^{\ell-1}\gamma(q_i,\sigma_{i+1})$. Letting $n_{q,\sigma}(s)$ denote the number of times that string s uses transition (q,σ), then

$$D_A(s) = \prod_{q,\sigma}\gamma(q,\sigma)^{n_{q,\sigma}(s)}. \tag{2}$$

Let $\hat{\gamma}(q,\sigma)$ denote the estimated probability that is given to transition (q,σ) in H. Provided H accepts s, the estimated probability of string s is given by

$$D_H(s) = \prod_{q,\sigma}\hat{\gamma}(q,\sigma)^{n_{q,\sigma}(s)}. \tag{3}$$

We aim to ensure that with high probability for $s \sim D_A$, if H accepts s then the ratio $D_H(s)/D_A(s)$ is close to 1. This is motivated by the following simple result.

Proposition 12 *Suppose that with probability $1-\frac{1}{4}\epsilon$ for $s \sim D_A$, we have $D_H(s)/D_A(s) \in [1-\frac{1}{4}\epsilon, 1+\frac{1}{4}\epsilon]$. Then $L_1(D_A,D_H) \leq \epsilon$.*

Proof:
$$L_1(D_A, D_H) = \sum_{s\in\Sigma^*} |D_A(s) - D_H(s)|$$
Let $X = \{s \in \Sigma^* \ : \ D_H(s)/D_A(s) \in [1 - \frac{1}{4}\epsilon, 1 + \frac{1}{4}\epsilon]\}$. Then

$$L_1(D_A, D_H) = \sum_{s\in X} |D_A(s) - D_H(s)| + \sum_{s\in\Sigma^*\setminus X} |D_A(s) - D_H(s)| \qquad (4)$$

The first term of the right-hand side of (4) is

$$\sum_{s\in X} D_A(s)\Big|(1 - D_H(s)/D_A(s))\Big| \leq \sum_{s\in X} D_A(s).\Big(\frac{\epsilon}{4}\Big) \leq \frac{\epsilon}{4}.$$

$D_A(X) \geq 1-\frac{1}{4}\epsilon$ and $D_H(X) \geq D_A(X)-\frac{1}{4}\epsilon$, equivalently $D_A(\Sigma^*\setminus X) \leq \frac{1}{4}\epsilon$ and $D_H(\Sigma^*\setminus X) \leq D_A(\Sigma^*\setminus X)+\frac{1}{4}\epsilon \leq \frac{1}{2}\epsilon$, hence the second term in the right-hand side of (4) is at most $\frac{3}{4}\epsilon$. $\square$

We have so far allowed the possibility that H may fail to accept up to a fraction $\frac{1}{4}\epsilon$ of strings generated by D_A. Of the strings s that are accepted by H, we want to ensure that with high probability $D_H(s)/D_A(s)$ is close to 1, to allow Proposition 12 to be used.

Suppose that $n_{q,\sigma}(s)$ is large, so that s uses transition (q, σ) a large number of times. In that case, errors in the estimate of transition probability $\gamma(q, \sigma)$ can have a disproportionately large influence on the ratio $D_H(s)/D_A(s)$. What we show is that with high probability for random $s \sim D_A$, regardless of how many times transition (q, σ) typically gets used, the training sample contains a large enough subset of strings that use that transition more times than s does, so that $\gamma(q, \sigma)$ is nevertheless known to a sufficiently high precision.

We say that $s' \in \Sigma^*$ is (q, σ)-good for some transition (q, σ), if s' satisfies

$$\Pr_{s\sim D_A}(n_{q,\sigma}(s) > n_{q,\sigma}(s')) \leq \frac{\epsilon}{4n|\Sigma|}.$$

Informally, a (q, σ)-good string is one that is more useful than most in providing an estimate of $\gamma(q, \sigma)$.

Proposition 13 *Let $m \geq 1$. Let S be a sample from D_A, $|S| \geq m(32n|\Sigma|/\epsilon)\ln(2n|\Sigma|/\delta'')$. With probability $1-\delta''(2n|\Sigma|)^{-1}$, for transition (q, σ) there exist at least $\epsilon(8n|\Sigma|)^{-1}|S|$ (q, σ)-good strings in S.*

Proof: From the definition of (q, σ)-good, the probability that a string generated at random over D_A is (q, σ)-good for transition (q, σ), is at least $\epsilon(4n|\Sigma|)^{-1}$.

Applying a standard Chernoff bound (see [3], p.360), for any transition (q, σ), with high probability over samples S, the number of (q,σ)-good strings in S is at least half the expected number as follows.

$$\Pr\left(|\{s \in S \ : \ s \text{ is } (q,\sigma)\text{−good}\}| < \frac{1}{2}\left(\frac{\epsilon}{4n|\Sigma|}|S|\right)\right) \leq \exp\left(-\frac{1}{8}\left(\frac{\epsilon}{4n|\Sigma|}\right)|S|\right) \tag{5}$$

We wish to bound this probability to be at most $\delta''(2n|\Sigma|)^{-1}$, so from Equation (5),

$$\exp\left(-\frac{1}{8}\left(\frac{\epsilon}{4n|\Sigma|}\right)|S|\right) \leq \frac{\delta''}{2n|\Sigma|}$$
$$|S| \geq \left(\frac{32n|\Sigma|}{\epsilon}\right) \ln\left(\frac{2n|\Sigma|}{\delta''}\right)$$

which is indeed satisfied by the assumption in the statement. □

Notation. Suppose S is as defined in Algorithm 2. Let $M_{q,\sigma}(S)$ be the largest number with the property that at least a fraction $\epsilon(8n|\Sigma|)^{-1}$ of strings in S use (q,σ) at least $M_{q,\sigma}(S)$ times.

Informally, $M_{q,\sigma}(S)$ represents a "big usage" of transition (q,σ) by a random string — the fraction of elements of S that use (q,σ) more than $M_{q,\sigma}(S)$ times is less than $\epsilon(8n|\Sigma|)^{-1}$. The next proposition states that $M_{q,\sigma}$ is likely to be an over-estimate of the number of uses of (q,σ) required for (q,σ)-goodness.

Proposition 14 *For any (q,σ), with probability $1-\delta''(2n|\Sigma|)^{-1}$ (over random samples S with $|S|$ as given in the algorithm),*

$$\Pr_{s\sim D_A}(n_{q,\sigma}(s) > M_{q,\sigma}(S)) \leq \frac{\epsilon}{4n|\Sigma|} \tag{6}$$

Proof: This follows from Proposition 13 (plugging in $m = (2n|\Sigma|/\delta'')(64n|\Sigma|/\epsilon\delta'')^2$). □

Theorem 15 *Suppose that H is a DFA that differs from A by the removal of a set of transitions that have probability at most $\frac{1}{2}\epsilon$ of being used by $s \sim D_A$. Then Algorithm 2 assigns probabilities $\hat{\gamma}(q,\sigma)$ to the transitions of H such the resulting distribution D_H satisfies $L_1(D_A, D_H) < \epsilon$, with probability $1-\delta''$.*

```
Algorithm 2

Input: DFA H, a subgraph of A.

generate sample S from D_A; |S| = (2n|Σ|/δ'') (64n|Σ|/εδ'')^2 (32n|Σ|/ε) ln (2n|Σ|/δ'');
For (each state q ∈ H, σ ∈ Σ)
  repeat
    for strings s ∈ S, trace paths through H;
    Let N_{q,−σ} be random variable: number of observations of
    state q up to and including the next observation of
    transition (q,σ) (include observations of q and (q,σ)
    in rejected strings);
  until(all strings in S have been traced);
  Let μ̂(N_{q,−σ}) be the mean of the observations of N_{q,−σ};
  Let γ̂(q,σ) = 1/μ̂(N_{q,σ});
For each q ∈ H, rescale γ̂(q,σ) such that ∑_{σ∈Σ} γ̂(q,σ) = 1.
```

Fig. 2. Finding Transition Probabilities

Proof: Recall Proposition 14, that with probability $1 - \delta''(2n|\Sigma|)^{-1}$,

$$\Pr_{s\sim D_A}(n_{q,\sigma}(s) > M_{q,\sigma}) \leq \frac{\epsilon}{4n|\Sigma|}$$

By definition of $M_{q,\sigma}(S)$, at least $|S|\epsilon(8n|\Sigma|)^{-1} > (2n|\Sigma|/\delta'')(64n|\Sigma|/\epsilon\delta'')^2$ members of $S_{q,\sigma}$ use (q,σ) at least $M_{q,\sigma}(S)$ times. Hence for any (q,σ), with probability $1-\delta''(2n|\Sigma|)^{-1}$, there are $M_{q,\sigma}(S)(2n|\Sigma|/\delta'')(64n|\Sigma|/\epsilon\delta'')^2$ uses of transition (q,σ).

Consequently, (again with probability $1 - \delta''(2n|\Sigma|)^{-1}$ over random choice of S), for any (q,σ) the set S generates a sequence of independent observations of state q, which continues until at least $M_{q,\sigma}(S)(2n|\Sigma|/\delta'')(64n|\Sigma|/\epsilon\delta'')^2$ of them resulted in transition (q,σ).

Let $N_{q,-\sigma}$ denote the random variable which is the number of times q is observed before transition (q,σ) is taken. Each time state q is visited, the selection of the next transition is independent of previous history, so we obtain a sequence of independent observations of $N_{q,-\sigma}$. So, with probability $1-\delta''(2n|\Sigma|)^{-1}$, the number of observations of $N_{q,-\sigma}$ is at least $M_{q,\sigma}(S)(2n|\Sigma|/\delta'')(64n|\Sigma|/\epsilon)^2$.

Recall Chebyshev's inequality, that for random variable X with mean μ and variance σ^2, for positive k,

$$\Pr(|X-\mu|>k) \leq \frac{\sigma^2}{k^2}.$$

$N_{q,-\sigma}$ has a discrete exponential distribution with mean $\gamma(q,\sigma)^{-1}$ and variance $\leq \gamma(q,\sigma)^{-2}$. Hence the empirical mean $\hat{\mu}(N_{q,-\sigma})$ is a random variable with mean $\gamma(q,\sigma)^{-1}$ and variance at most $\gamma(q,\sigma)^{-2}(M_{q,\sigma})^{-1}(2n|\Sigma|/\delta'')^{-1}(64n|\Sigma|/\epsilon\delta'')^{-2}$. Applying Chebyshev's inequality with $\hat{\mu}(N_{q,-\sigma})$ for X, and $k=\gamma(q,\sigma)^{-1}\epsilon\delta''(64n|\Sigma|\sqrt{M_{q,\sigma}})^{-1}$, we have

$$\Pr\left(|\hat{\mu}(N_{q,-\sigma})-\gamma(q,\sigma)^{-1}|>\gamma(q,\sigma)^{-1}\left(\frac{\epsilon\delta''}{64n|\Sigma|\sqrt{M_{q,\sigma}}}\right)\right) \leq \frac{\delta''}{2n|\Sigma|}.$$

Note that for $x,y>0$ and $\frac{1}{2}>\xi>0$, if $|y-x|<x\xi$ then $|y^{-1}-x^{-1}|<2x^{-1}\xi$, and applying this to the left-hand side of the above, we deduce

$$\Pr\left(|\hat{\gamma}(q,\sigma)-\gamma(q,\sigma)|>2\gamma(q,\sigma)\left(\frac{\epsilon\delta''}{64n|\Sigma|\sqrt{M_{q,\sigma}}}\right)\right) \leq \frac{\delta''}{2n|\Sigma|}.$$

The rescaling at the end of Algorithm 2 (which may be needed as a result of infrequent transitions not being included in the hypothesis automaton) loses a factor of at most 2 from the upper bound on $|\gamma(q,\sigma)-\hat{\gamma}(q,\sigma)|$. Overall, with high probability $1-\delta''(2n|\Sigma|)^{-1}$,

$$|\hat{\gamma}(q,\sigma)-\gamma(q,\sigma)| \leq \left(\frac{\epsilon\delta''\gamma(q,\sigma)}{16n|\Sigma|\sqrt{M_{q,\sigma}}}\right). \tag{7}$$

For $s\in\Sigma^*$ let $n_q(s)$ denote the number of times the path of s passes through state q. By definition of $M_{q,\sigma}(S)$, for any transition (q,σ) with high probability $1-\epsilon(4n|\Sigma|)^{-1}$,

$$E_{s\sim D_A}[n_q(s)] < M_{q,\sigma}(S)/\gamma(q,\sigma). \tag{8}$$

For $s\sim D_A$ we upper bound the expected log-likelihood ratio,

$$\log\left(\frac{D_H(s)}{D_A(s)}\right) = \sum_{i=1}^{|s|}\frac{\hat{\gamma}(q_i,\sigma_i)}{\gamma(q_i,\sigma_i)}$$

where σ_i is the i-th character of s and q_i is the state reached by the prefix of length $i-1$.

Suppose A generates a prefix of s and reaches state q. Let random variable X_q be the contribution to $\log(D_H(s)/D_A(s))$ when A generates the next character.

$$
\begin{aligned}
E[X_q] &= \sum_\sigma \gamma(q,\sigma)\log\left(\frac{\hat\gamma(q,\sigma)}{\gamma(q,\sigma)}\right)\\
&= \sum_\sigma \gamma(q,\sigma)[\log(\hat\gamma(q,\sigma)) - \log(\gamma(q,\sigma))]
\end{aligned}
$$

We claim that (with high probability $1-\delta''(2n|\Sigma|)^{-1}$)

$$
\log(\hat\gamma(q,\sigma)) - \log(\gamma(q,\sigma)) \leq |\hat\gamma(q,\sigma)-\gamma(q,\sigma)|\left(\frac{1}{\gamma(q,\sigma)}\right)A_{q,\sigma} \tag{9}
$$

for some $A_{q,\sigma} \in [1-\epsilon\delta''(8n|\Sigma|\sqrt{M_{q,\sigma}})^{-1}, 1+\epsilon\delta''(8n|\Sigma|\sqrt{M_{q,\sigma}})^{-1}]$. The claim follows from (7) and the inequality, for $|\xi| < x$, that $\log(x+\xi)-\log(x) \leq \xi x^{-1}(1+2\xi/x)$ (plug in $\gamma(q,\sigma)$ for x). Consequently,

$$
\begin{aligned}
E[X_q] &\leq \sum_\sigma \gamma(q,\sigma)\left(\frac{1}{\gamma(q,\sigma)}\right)A_{q,\sigma}[\hat\gamma(q,\sigma)-\gamma(q,\sigma)]\\
&= \sum_\sigma A_{q,\sigma}[\hat\gamma(q,\sigma)-\gamma(q,\sigma)]\\
&= \sum_\sigma [\hat\gamma(q,\sigma)-\gamma(q,\sigma)] + \sum_\sigma B_{q,\sigma}[\hat\gamma(q,\sigma)-\gamma(q,\sigma)]
\end{aligned}
$$

for some $B_{q,\sigma} \in [-\epsilon\delta''(8n|\Sigma|\sqrt{M_{q,\sigma}})^{-1}, \epsilon\delta''(8n|\Sigma|\sqrt{M_{q,\sigma}})^{-1}]$. The first term vanishes, so we have

$$
\begin{aligned}
E[X_q] &\leq \sum_\sigma B_{q,\sigma}[\hat\gamma(q,\sigma)-\gamma(q,\sigma)]\\
&= \frac{\epsilon\delta''}{8n|\Sigma|}\sum_\sigma\left(\frac{1}{\sqrt{M_{q,\sigma}}}\right)[\hat\gamma(q,\sigma)-\gamma(q,\sigma)]\\
&\leq \frac{\epsilon\delta''}{8n|\Sigma|}\sum_\sigma \frac{\gamma(q,\sigma)}{M_{q,\sigma}}
\end{aligned}
$$

where the last inequality uses (7). For $s \sim D_A$, the expected contribution to $\log(D_H(s)/D_A(s))$ from all $n_q(s)$ usages of state q is, using (8), at most

$$
E[n_q(s)]\left(\frac{\epsilon}{8n|\Sigma|}\right)\sum_\sigma \frac{1}{E[n_q(s)]} = \left(\frac{\epsilon\delta'' n_q(s)}{8n|\Sigma|}\right)|\Sigma|\left(\frac{1}{n_q(s)}\right) = \frac{\epsilon\delta''}{8n}
$$

The total expected contribution from all n states q, each being used $n_q(s)$ times is

$$
\sum_{q\in Q}\frac{\epsilon\delta''}{8n} = \frac{\epsilon\delta''}{8}. \tag{10}
$$

Using Markov's inequality, there is a probability at most δ'' that $\log(D_H(s)/D_A(s))$ is more than $\epsilon/8$.

Finally, in order to use Proposition 12, note that $(D_H(s)/D_A(s)) \in \left[1 - \frac{1}{4}\epsilon, 1 + \frac{1}{4}\epsilon\right]$ follows from $\log(D_H(s)/D_A(s)) \in \left[-\frac{1}{8}\epsilon, \frac{1}{8}\epsilon\right]$.

□

The sample size expression is polynomial in $1/\delta''$; we can convert the algorithm into one that is logarithmic in $1/\delta''$ as follows. If we run the algorithm x times using $\delta'' = \frac{1}{10}$, we obtain x values for the likelihood of a string, rather than just one. It is not hard to show that for $x = O(\log(1/\delta''))$, the median will be accurate with probability $1 - \delta''$.

6 Discussion and Conclusions

We can now of course put these two algorithms together using any values of δ' and δ'' that add up to at most δ (δ being the overall uncertainty bound). By combining the results of Theorem 11 and Theorem 15, we get the following.

Theorem 16 *Given an automaton with alphabet Σ and at most n states, which is μ-distinguishable for some parameter μ, then Algorithm 1 and Algorithm 2 run in time polynomial in the above parameters (also ϵ and δ), producing a model which with probability at least $1-\delta$ differs (in L_1 distance) from the original automaton by at most ϵ.*

Our algorithms are structurally similar to previous algorithms for learning PDFAs. One change worth noting that we have made, is that for each algorithm a single sample is taken at the beginning, and all elements of that sample are treated the same way. Previous related algorithms (including the version of this paper in [11]) typically draw a sample at each iteration, so as to ensure independence between iterations. In practice it is natural and realistic to assume that every measurement is extracted from all the data.

We have shown that as a result of using the variation distance as criterion for precise learning, we can obtain sample-size bounds that do not involve the length of strings generated by unknown PDFAs. In the appendix we show why the KL-divergence requires a limit on the expected length of strings that the target automaton generates. Furthermore, our approach has addressed the issue of extracting more information from long strings than short strings, which is necessary in order to estimate heavily-used transitions with higher precision. Perhaps the main open problem is the issue of learnability of PDFAs where there is no a priori "distinguishability" of states.

References

[1] N. Abe, J. Takeuchi and M. Warmuth. Polynomial Learnability of Stochastic Rules with respect to the KL-divergence and Quadratic Distance. *IEICE Trans. Inf. and Syst., Vol E84-D(3) pp. 299-315 (2001).*

[2] N. Abe and M.K. Warmuth. On the Computational Complexity of Approximating Distributions by Probabilistic Automata. *Machine Learning, 9, pp. 205-260 (1992).*

[3] M. Anthony and P.L. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press (1999).

[4] A. Clark and F. Thollard. PAC-learnability of Probabilistic Deterministic Finite State Automata. Journal of Machine Learning Research 5 pp. 473-497 (2004).

[5] T.M. Cover and J.A. Thomas. *Elements of Information Theory* Wiley Series in Telecommunications. John Wiley & Sons (1991).

[6] M. Cryan, L. A. Goldberg and P. W. Goldberg. Evolutionary Trees can be Learned in Polynomial Time in the Two-State General Markov Model. *SIAM Journal on Computing 31(2) pp. 375-397 (2001).*

[7] P. Dupont, F. Denis and Y. Esposito. Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition, 38, pp. 1349-1371 (2005)*

[8] P.W. Goldberg. Some Discriminant-based PAC Algorithms. *Journal of Machine Learning Research, Vol.7, pp. 283-306 (2006).*

[9] C. de la Higuera and J. Oncina. Learning Stochastic Finite Automata. *Procs. of the 7th International Colloquiuum on Grammatical Inference (ICGI), LNAI 3264, pp. 175-186 (2004).*

[10] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire and L. Sellie. On the Learnability of Discrete Distributions. *Procs. of the 26th Annual ACM Symposium on Theory of Computing, pp. 273-282 (1994).*

[11] N. Palmer and P.W. Goldberg. PAC-learnability of Probabilistic Deterministic Finite State Automata in terms of Variation Distance *Procs. of the 16th conference on Algorithmic Learning Theory (ALT); LNAI 3734, pp. 157-170 (2005).*

[12] N. Palmer and P.W. Goldberg. PAC Classification via PAC Estimates of Label Class Distributions. *Tech rept. 411, Dept. of Computer Science, University of Warwick (2004). Available from arXiv as cs.LG/0607047*

[13] D. Ron, Y. Singer and N. Tishby. On the Learnability and Usage of Acyclic Probabilistic Finite Automata. *Journal of Computer and System Sciences, 56(2), pp. 133-152 (1998).*

7 Appendix

We show that in order to learn a PDFA with respect to KL Divergence, an upper bound on the expected length of string output by the target PDFA must be known.

Theorem 17 *Consider the target automaton A, shown in Figure 3.*

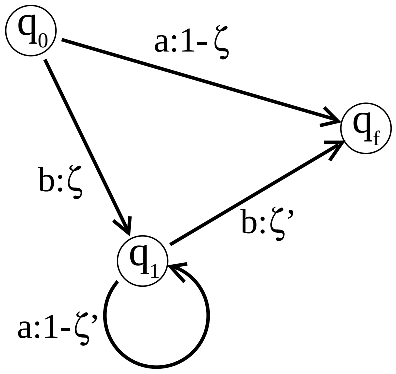

Fig. 3. Target PDFA A.

Suppose we wish to construct, with probability at least $1-\delta$, a distribution D_H such that $D_{KL}(D_A, D_H) < \epsilon$, using a finite sample of strings generated by D_A. There is no algorithm that achieves this using a sample size that depends only on ϵ and δ.

Proof: A outputs the string a with probability $1-\zeta$, and outputs a string of the form $b(a)^*b$ with probability ζ.

Suppose an algorithm draws a sample S (from D_A) from which it is to construct D_H, with $|S| = f(\epsilon, \delta)$. Let $\zeta = \frac{1}{2|S|}$ be the probability that a random string is of the form $b(a)^n b$. Notice that S will be composed (entirely or almost entirely) of observations of string a. Therefore there is no way that the algorithm can accurately gauge the probability ζ' (see Figure 3).

For $i \in \mathbf{N}$ let P_i be a probability distribution over the length ℓ of output strings, where $P_i(1) = (1-\zeta)$ and over all values of ℓ greater than 1 the distribution is a discrete exponential distribution defined as follows.

An infinite sequence $\{n_1, n_2, \ldots\}$ exists (see Lemma 18), such that P_1 has a probability mass of $\frac{1}{4|S|}$ (half of the probability of generating a string of length greater than 1) over the interval $\{1, ..., n_1\}$, P_2 has probability mass of $\frac{1}{4|S|}$ over the interval $\{n_1+1, ..., n_2\}$, and in general P_i has probability of $\frac{1}{4|S|}$ over the interval $\{n_{i-1}+1, ..., n_i\}$. Let s_ℓ denote the string $ba^{(\ell-2)}b$ (with length

ℓ). Given any distribution[4] D_H, for any $0 < \omega < 1$ there exists an interval $I_k = \{n_{k-1}+1, ..., n_k\}$ such that

$$\sum_{\ell \in I_k} D_H(s_\ell) \leq \omega.$$

To lower-bound the KL divergence, we now redistribute the probability distribution of D_H in order to minimise the incurred KL divergence (from the true distribution D_A), subject only to the condition that I_k still contains at most ω of the probability mass. In order to minimise the KL divergence, by a standard convexity argument, the algorithm must distribute the probability in direct proportion to D_A.

$$\begin{aligned}
\forall \ell \in I_k : \frac{D_H(\ell)}{D_A(\ell)} &= 4|S|\omega \\
\forall \ell \notin I_k : \frac{D_H(\ell)}{D_A(\ell)} &= \left(\frac{1-\omega}{1-\frac{1}{4|S|}}\right)
\end{aligned}$$

It follows that the KL divergence can be written in terms of $|S|$ and ω in the following way:

$$\begin{aligned}
D_{KL}(D_A||D_H) &= \sum_{\ell \in \mathbf{N}} D_A(s_\ell). \log\left(\frac{D_A(s_\ell)}{D_H(s_\ell)}\right) \\
&= \sum_{\ell \in I_k} D_A(s_\ell). \log\left(\frac{1}{4|S|\omega}\right) + \sum_{\ell \notin I_k} D_A(s_\ell). \log\left(\frac{1-\frac{1}{4|S|}}{1-\omega}\right) \\
&= \left(\frac{1}{4|S|}\right) \log\left(\frac{1}{4|S|\omega}\right) + \left(1 - \frac{1}{4|S|}\right) \log\left(\frac{1-\frac{1}{4|S|}}{1-\omega}\right) \\
&\geq \left(\frac{1}{4|S|}\right) (-2\log(|S|) - \log(\omega)) + \log\left(1 - \frac{1}{4|S|}\right)
\end{aligned}$$

Suppose that $\omega < 2^{-2\left(|S|\left(\epsilon - \log\left(1-\frac{1}{4|S|}\right)\right)+\log(|S|)\right)}$. We can deduce that $D_{KL}(D_A, D_H) > \epsilon$.

It has been shown that for any specified ϵ, given any hypothesis distribution D_H, an exponential distribution D_A exists such that $D_{KL}(D_A, D_H) > \epsilon$. $\square$

[4] Note that this is a representation independent result - the distribution need not be generated by an automaton.

Lemma 18 *Given any positive integer $n_i \geq 1$ in the domain of string lengths, an exponential probability distribution exists such that at least $\frac{1}{4|S|}$ of the probability mass lies in the range $\{n_i + 1, ..., n_{i+1}\}$.*

Proof: If we look at strings of length greater than 1, then given some value n_i, there is some exponential distribution over these strings such that there exists an interval $\{n_i + 1, ..., n_{i+1}\}$ containing half of the probability mass of the distribution.

For an automaton A' (of a form similar to Figure 3), the probability of an output string having a length greater than 1 is $\frac{1}{2|S|}$. Let $\ell_b = \ell - 1$ for those strings with $\ell > 1$ (where ℓ_b represents the number of characters following the initial b), and let s_{ℓ_b} represent the string starting with b which has length ℓ. For any value of n_i, we can create a distribution:

$$D_{A'}(s_{\ell_b}) = \left(\frac{1}{2|S|}\right)\left(\frac{\ln\left(\frac{4}{3}\right)}{n_i + 1}\right)\exp\left(-\left(\frac{\ln\left(\frac{4}{3}\right)}{n_i + 1}\right)\ell\right)$$

A fraction $\frac{1}{8|S|}$ of the probability mass lies in the interval $\{2, ..., n_i\}$. There exists a value $n_{i+1} = \left\lceil (n_i + 1)\left(\ln(4)/\ln\left(\frac{4}{3}\right)\right)\right\rceil$ such that at least $\frac{1}{4|S|}$ of the probability mass lies in the interval $\{n_i + 1, ..., n_{i+1}\}$. □