Learnability of Probabilistic Automata via Oracles

Omri Guttman, S.V.N. Vishwanathan, and Robert C. Williamson

Statistical Machine Learning Program,
National ICT Australia, RSISE, Australian National University,
Canberra, ACT, Australia
{Omri.Guttman, SVN.Vishwanathan, Bob.Williamson}@nicta.com.au

Abstract. Efficient learnability using the state merging algorithm is known for a subclass of probabilistic automata termed μ-distinguishable. In this paper, we prove that state merging algorithms can be extended to efficiently learn a larger class of automata. In particular, we show learnability of a subclass which we call μ_2-distinguishable. Using an analog of the Myhill-Nerode theorem for probabilistic automata, we analyze μ-distinguishability and generalize it to μ_p-distinguishability. By combining new results from property testing with the state merging algorithm we obtain KL-PAC learnability of the new automata class.

1 Introduction

In this paper we investigate the following question: given a finite set of samples drawn from a target distribution[1] generated by a probabilistic automaton such that the suffix distribution from any two states of the automata can be distinguished efficiently by an oracle; can we learn this distribution efficiently? The definition of an oracle as well as our notion of efficiency is defined rigorously in the following sections.

Probabilistic deterministic finite automata (or *PDFA*) are stochastic extensions of deterministic finite automata (*DFA*), a well studied class of formal models (see e.g. Hopcroft and Ullman, 1979). The PDFA class finds uses in a variety of areas in pattern recognition and machine learning including computational linguistics, time series analysis, computational biology, speech recognition, and network intrusion detection (see e.g. Vidal et al., 2005a,b). The problem of learning PDFA efficiently from sample data is therefore of significant practical importance.

Many researchers have investigated the KL-PAC learnability of PDFA. There are indications that the general problem of PDFA learning is hard. For instance, Kearns et al. (1994) showed that KL-PAC learnability of PDFA implies the computability of the *noisy parity function*, thus violating the *noisy parity assumption*, widely believed to be true in the cryptography community (see e.g. Kearns,

[1] Note that we assume the samples are drawn from the target distribution, and therefore our framework differs from the distribution-free setting.

1993). This is demonstrated by showing how by KL-PAC learning a specific family of (acyclic) PDFA, one can evaluate the noisy parity function[2].

On the other hand, Ron et al. (1995) showed that a subclass of acyclic PDFA, which they call μ-distinguishable, can be efficiently learned using the state merging algorithm. Clark and Thollard (2004) extended this result to PDFA with bounded expected suffix length from every state. Roughly speaking, for an automaton in this subclass, given any two states there exists at least one string whose suffix probability from the two states differs by at least μ. In other words, the suffix distribution of any two states of the PDFA are at least μ apart in the L_∞ distance. The state merging algorithm uses an efficient test to distinguish between suffix distributions.

However, the L_∞ distance between two distributions can often be misleading. Consider two extreme cases: let, D_1 and D_2 be distributions over $\{1, \ldots, 2n\}$. Furthermore, let $D_1(i) = 1/n$ for $i = 1 \ldots n$ and 0 otherwise, while $D_2(i) = 1/n$ for $i = n+1 \ldots 2n$ and 0 otherwise. Clearly, D_1 and D_2 have disjoint support, but $||D_1 - D_2||_\infty$ is only $1/n$. On the other hand, even if D_1 and D_2 agree on all but two elements, $||D_1 - D_2||_\infty$ could potentially be large. The class of μ-distinguishable automata may therefore be unsatisfactory, and we would like to guarantee learnability of a more general family.

In Carrasco and Oncina (1999) it was shown that corresponding to every distribution induced by a PDFA there exists a canonical PDFA with the minimal number of states which induces the same distribution. Furthermore, the suffix distributions of the states of the canonical PDFA are unique. Therefore, if we are given an oracle which can distinguish between two suffix distributions, we can learn the PDFA. In this paper we show how the state merging algorithm can use an oracle to learn μ_p-distinguishable PDFA. Our definition of μ_p distinguishability is a generalization of μ-distinguishability. The suffix distributions of any two states of a μ_p-distinguishable PDFA are at least μ apart in the L_p distance for some $1 \leq p \leq \infty$.

In order to distinguish between suffix distributions we use property testing algorithms. Given the ability to perform local queries, property testing algorithms aim to determine if an object satisfies a particular property or is far from satisfying it. These algorithms are typically sub-linear because they perform the task by inspecting only a fraction of samples drawn from the global object, and typically provide bounds on the probability of failure (see e.g. Ron, 2001).

For our purposes, property testing algorithms act as an oracle in the following sense: Given a norm $||\cdot||$ and two distributions D_1 and D_2 over Σ^*, the algorithm outputs *ACCEPT* with probability at least $1 - \delta$ whenever $||D_1 - D_2|| < \varepsilon$ and it outputs *REJECT* with probability at least $1 - \delta$ whenever $||D_1 - D_2|| > \varepsilon'$, $(\varepsilon' > \varepsilon)$. In particular, using a property testing algorithm described in

[2] We mention in passing that even in the weaker, L_1-PAC learning model, we have shown, and will publish elsewhere, that general PDFA learnability still violates the noisy parity assumption. This implies that the difficulty is inherent to PDFA learning, and not merely an artifact of the KL-divergence.

Batu et al. (2000) we present a modified state merging algorithm for learning μ_2-distinguishable PDFA efficiently.

Characterization of the family of PDFA which can be learned efficiently is a deep question for which only partial answers are known so far. For instance, it is well known that the noisy parity counterexample construction of Kearns et al. (1994) is not μ-distinguishable. In fact, for every $p > 1$, the automaton's μ_p-distinguishability decays exponentially with the number of states. Yet it is readily shown (Murphy, 1996) that this family of automata is μ_1-distinguishable whilst remaining hard to learn. The lower bounds for L_1 distance testing proven by Batu et al. (2000) also imply that learning μ_1-distinguishable PDFA (using state merging) is hard. We seek to understand the criterion which implies efficient PDFA learning and this paper is a step in that direction.

We begin by formalizing our notation in Sect. 2 and review the Myhill-Nerode theorem and its analog for PDFA in Sect. 3. Section 4 is devoted to a description of μ_p-distinguishability. In Sect. 5 we describe our modified state merging algorithm. In Sect. 5.1 we demonstrate a particular instance of our algorithm for learning μ_2-distinguishable automata, and establish its polynomial sample and computational complexity. We conclude with a discussion of possible extensions and interesting research directions.

2 Notation

For concepts related to PDFA learning, we adopt the notation of Clark and Thollard (2004) where applicable. We also use the notation of Batu et al. (2000) when we discuss property testing over distributions.

We use Σ to denote a discrete alphabet consisting of $|\Sigma|$ elements, and by Σ^* we denote the set of finite (but unbounded) length sequences over Σ. A subset $\mathcal{L} \subseteq \Sigma^*$ is called a *formal language*. A distribution D over Σ^* is a function which assigns a probability $D(s)$ to all sequences $s \in \Sigma^*$, such that $0 \leq D(s) \leq 1$, and $\sum_{s \in \Sigma^*} D(s) = 1$. The distribution D is also called a *stochastic language*.

A PDFA, denoted by A, is a tuple $(Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $q_f \notin Q$ is the final state, $\zeta \notin \Sigma$ is the termination symbol, $\tau : Q \times \Sigma \cup \{\zeta\} \to Q \cup \{q_f\}$ is the transition function, and $\gamma : Q \times \Sigma \cup \{\zeta\} \to [0, 1]$ is the next symbol probability function. $\gamma(q, \sigma) = 0$ for any $\sigma \in \Sigma$ when $\tau(q, \sigma)$ is not defined. From each state $q \in Q$, the sum of output probabilities must be one: $\sum_{\sigma \in \Sigma \cup \{\zeta\}} = 1$.

A labelled graph $G = (V, E)$ is defined by a set of nodes V and a set of labelled edges $E \subseteq V \times \Sigma \times V$. Every PDFA A has an associated underlying labelled graph G_A. When it is clear from context we will use G and G_A interchangeably.

All transitions which emit the ζ symbol point to the final state. The functions γ and τ can be recursively extended to strings: $\gamma(q, \sigma_1\sigma_2 \ldots \sigma_k) = \gamma(q, \sigma_1) \cdot \gamma(\tau(q, \sigma_1), \sigma_2 \ldots \sigma_k)$, and when $\gamma(q, \sigma_1\sigma_2 \ldots \sigma_k) > 0$, we have $\tau(q, \sigma_1\sigma_2 \ldots \sigma_k) = \tau(\tau(q, \sigma_1), \sigma_2 \ldots \sigma_k)$.

A PDFA thus induces a distribution, or in other words, a *stochastic regular language* over Σ^*, with $P^A(s) = \gamma(q_0, s\zeta)$, the probability of generating the string s. We define $P^A_q(s) = \gamma(q, s\zeta)$ to be the *suffix distribution* of the state q.

Given two distributions D_1 and D_2 over Σ^*, we will use the following notions of distance:

– For $1 \leq p < \infty$, the L_p distance between D_1 and D_2 is defined as:

$$\|D_1 - D_2\|_p = \sum_{s \in \Sigma^*} \left[|D_1(s) - D_2(s)|^p \right]^{1/p}.$$

– In the $p \to \infty$ limit, the resulting L_∞ distance, is defined as:

$$\|D_1 - D_2\|_\infty = \max_{s \in \Sigma^*} |D_1(s) - D_2(s)|.$$

– The KL-divergence between D_1 and D_2 is defined as:

$$\mathrm{KL}(D_1 \| D_2) = \sum_{s \in \Sigma^*} D_1(s) \log \left(\frac{D_1(s)}{D_2(s)} \right).$$

Note that the KL divergence is *not* a metric. As shown in Cover and Thomas (1991), for any pair of distributions D_1 and D_2, it holds that $\mathrm{KL}(D_1 \| D_2) \geq \frac{1}{2 \ln 2} \|D_1 - D_2\|_1^2$, which in turn upper-bounds all L_p-distances. Where appropriate, we will use the notation $\mathrm{KL}(A || \widehat{A})$ to denote $\mathrm{KL}(P^A || P^{\widehat{A}})$.

We will use the following notion of learnability, due to Clark and Thollard (2004):

Definition 1 (KL-PAC). *Given a class of distributions $\mathcal{D}$ over Σ^*, an algorithm* KL-PAC *learns $\mathcal{D}$ if there is a polynomial $q(\cdot)$ such that for all $D \in \mathcal{D}$, $\varepsilon > 0$ and $\delta > 0$, the algorithm is given a sample S of size m and produces a hypothesis $\widehat{D}$, such that* $\Pr \left[KL(D \parallel \widehat{D}) > \varepsilon \right] < \delta$ *whenever $m > q(1/\varepsilon, 1/\delta, |D|)$. By $|D|$ we denote some measure of the complexity of the target. The algorithm's running time is bounded by a polynomial in m plus the total length of the strings in S.*

The definition of L_p-PAC learnability is analogous, with the corresponding change in distance measure.

3 Characterization Theorem for Suffix Distributions

In the theory of formal languages, the well known Myhill-Nerode theorem provides a necessary and sufficient condition for a language to be *regular*, and thus accepted by a DFA (see e.g. Hopcroft and Ullman, 1979). The theorem states that a language is *regular* if and only if it is the union of some equivalence classes of a *right invariant equivalence relation* of finite *index*. Definitions of the key concepts in the theorem are:

– For a given pair of strings $x, y \in \Sigma^*$ and a formal language $\mathcal{L}$, a *right invariant equivalence relation* "$\sim$" is defined by:

$$x \sim y \iff (xz \in \mathcal{L} \iff yz \in \mathcal{L}) \quad \forall z \in \Sigma^*.$$

– The *index* of a language is defined as the number of its equivalence classes.

A analogous theorem for PDFA is due to Carrasco and Oncina (1999). The concept of a right equivalence relation for a stochastic language is as follows:

– For a given pair of strings $x, y \in \Sigma^*$, and a stochastic language $\mathcal{L}$ generated by a PDFA A, a *stochastic right invariant equivalence relation* "$\sim$" is defined by:
$$x \sim y \quad \Longleftrightarrow \quad P^A(xz) = P^A(yz) \quad \forall z \in \Sigma^*.$$

The modified theorem now states that a stochastic language is generated by a PDFA if and only if it is the union of some equivalence classes of a stochastic right invariant equivalence relation of finite index.

A *canonical generator* is defined to be the PDFA with the minimal number of states generating a given stochastic regular language $\mathcal{L}$ (Carrasco and Oncina, 1999). Using the definition of the right invariant equivalence relation and the minimality of the canonical generator, it can be shown that the suffix distribution of every state q of the canonical generator is distinct. In other words, given an oracle which can distinguish between suffix distributions, we can learn the distribution induced by a PDFA by identifying the right invariant equivalence classes. In Sect. 4 we show that the state merging algorithm for learning μ-distinguishable automata implicitly uses such an oracle which exploits the L_∞ distance between suffix distributions. We then extend the algorithm to use other oracles, guaranteeing learnability of a larger class of PDFA.

4 μ_p-Distinguishability

The general PDFA learning problem is hard, and therefore additional conditions are required before attempting to provide an efficient learning algorithm. The concept of μ-distinguishability was first introduced by Ron et al. (1995), where it was shown to be sufficient for KL-PAC learning of *acyclic* PDFA.

Definition 2 (μ-distinguishability). *Given a PDFA $A = (Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$, two states $q, q' \in Q$ are μ-distinguishable if there exists a string $s \in \Sigma^*$ such that $|\gamma(q, s\zeta) - \gamma(q', s\zeta)| \geq \mu$. A PDFA A is μ-distinguishable if every pair of states in it are μ-distinguishable.*

Recently, Clark and Thollard (2004) extended the result to general PDFA learning, while imposing an additional condition, namely an upper bound on the expected suffix length from all states.

We seek to understand which criteria will in fact enable efficient testing for discrimination between distributions. For instance, μ-distinguishability is equivalent to demanding that $||P_q^A - P_{q'}^A||_\infty \geq \mu$ for every $q, q' \in Q$. This provides for an efficient and accurate test for deciding whether or not two distributions over strings are similar (i.e. drawn from the same suffix distribution). We argue in Sect. 5 that such efficient and accurate testing can in fact be achieved using relaxed conditions, which in turn also guarantee KL-PAC learnability.

4.1 Definition of μ_p-Distinguishability

Definition 3 (μ_p-distinguishability). *Given a PDFA $A = (Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$, two states $q, q' \in Q$ are termed μ_p-distinguishable if the L_p distance between their respective suffix distributions is at least μ. Written out in terms of the individual suffix probabilities, the condition becomes:*

$$\|P_q^A - P_{q'}^A\|_p \geq \mu.$$

A PDFA is μ_p-distinguishable if every pair of states in it are μ_p-distinguishable.

Note that for $p = \infty$ we recover the original μ-distinguishability condition. Moreover, for any distribution D over Σ^*, if $1 \leq p_1 < p_2 \leq \infty$ then we have $\|D\|_{p_1} \geq \|D\|_{p_2}$; hence the μ_{p_1}-distinguishable class *properly* contains the μ_{p_2}-distinguishable class.

5 State Merging with Oracles

In order to describe how state merging algorithms can use oracles to learn PDFA distributions, we first provide a modular analysis of the proof due to Clark and Thollard (2004), and then extend it to deal with oracles. In particular, we show that the state merging algorithm may be decoupled into two parts:

- A construction algorithm which iteratively builds the PDFA graph and sets the transition probabilities.
- An oracle, which enables efficient and accurate testing for whether or not two sample sets were drawn from two distinct suffix distributions.

Given such an oracle, the state merging algorithm will induce a PDFA such that with high probability, the KL-divergence between target and induced distributions can be made arbitrarily small.

The learning algorithm is given the following parameters as input: an alphabet Σ, an upper bound on the expected length of strings generated from *any* state of the target L, an upper bound on the number of states of the target n, a confidence parameter δ and a precision parameter ε. Pseudocode for the state merging algorithm is given in Algorithm A (Appendix A), while a full description can be found in Sect. 3 of Clark and Thollard (2004).

For our purposes, an oracle is a black-box which can distinguish between suffix distributions. More formally:

Definition 4 (Oracle). *Given a class $\mathcal{H}$ of PDFA, an oracle $\mathcal{O}_{\mathcal{H}}$ is said to* (δ, m)-match *the class $\mathcal{H}$ if for any PDFA $A \in \mathcal{H}$ and for any pair of states q, q' in A, given sample sets of at least m samples drawn as suffixes from q and q', the oracle can determine with probability at least $1 - \delta$ whether or not the two sets were drawn from suffix distributions of the same state.*

The learning algorithm we use is analogous to the state merging algorithm described in Clark and Thollard (2004), with the oracle $\mathcal{O}_{\mathcal{H}}$ testing whether to merge a hypothesized *candidate* state (see Definition 5) with an existing one, or to construct a new state. This leads to our main result:

Theorem 1. *Let $\mathcal{H}$ be a class of PDFAs over the alphabet Σ, $\epsilon > 0$, $\delta > 0$, L and n positive integers, and $\delta_1, \delta_2, \epsilon_1, m_2$ as defined in* (1) *and* (2) *(Appendix B).*

Suppose $\mathcal{O}_{\mathcal{H}}$ is a (δ_1, m_1)-matching oracle for $\mathcal{H}$. For every n-state PDFA $A \in \mathcal{H}$ such that the expected length of the string generated from every state is less than L, with probability $1 - \delta$ *over a random draw of* $\max(m_1, m_2)$ *samples generated by A, Algorithm 1 produces an hypothesis PDFA $\widehat{A}$ such that* $\mathrm{KL}(A \parallel \widehat{A}) < \epsilon$.

Our proof closely follows Clark and Thollard (2004), with a number of key changes:

- In the original proof, distinguishability between two states' suffix distributions (with high probability) is inherent to the sample size bounds. In our case, the oracle provides the distinguishing test, so the size of the multiset drawn at each step of the algorithm is reformulated to reflect this (see (1), Appendix A).
- In our case, two sources of randomness are present: the randomly drawn sample sets and the oracle. To bound the probability of error, both sources need to be accounted for.

With the exceptions noted above, the original proof is directly transferable to our setting. The proof sketch can be found in Appendix B.

5.1 Learning μ_2 Efficiently

We use a result on testing distribution proximity in L_2 due to Batu et al. (2000) to show learnability of the μ_2-distinguishable class. The result applies to distributions over discrete sets, and the computational complexity does not depend on the size of the set. Specifically, for Algorithm *L_2-Distance-Test* described in Appendix C the following is proven:

Theorem 2 (Batu et al. (2000)). *Given a parameter δ and $m = O(\frac{1}{\varepsilon^4})$ samples drawn from distributions D_1 and D_2 over a set of n elements, if $\|D_1 - D_2\|_2 \le \varepsilon/2$, Algorithm L_2-Distance-Test will output* ACCEPT *with probability at least* $(1 - \delta)$. *If $\|D_1 - D_2\|_2 \ge \varepsilon$ then the algorithm outputs* REJECT *with probability at least* $1 - \delta$. *The running time of the algorithm is* $O\left(\varepsilon^{-4} \log \frac{1}{\delta}\right)$.

Note that the running time of the test is independent of n. As a consequence of Theorem 2, the L_2-Distance-Test algorithm can serve as a $\left(\delta, \frac{C_2}{\varepsilon^4}\right)$-matching oracle for the μ_2-distinguishable PDFA class, where C_2 is a constant hidden in the asymptotic notation. Thus, as a direct consequence of Theorem 1 and the L_2 matching oracle, we have the following lemma:

Lemma 1. *The μ_2-distinguishable class is efficiently learnable.*

6 Discussion and Conclusion

We introduced μ_p-distinguishable automata, which generalize the concept of μ-distinguishable automata. Using a new modularized analysis we extended the proof of KL-PAC learnability of μ-distinguishable automata via the state merging algorithm. This new insight allows us to extend state merging to use oracles. We use an existing L_2 property testing algorithm to learn μ_2-distinguishable automata efficiently via state merging.

In the noisy parity PDFA family, the μ_1-distinguishability is a constant, while the μ_2-distinguishability is $O(2^{-n/2})$ and the μ_∞-distinguishability is $O(2^{-n})$ (where n denotes the number of states). By setting $n = \alpha \log t$ we obtain for this family a μ_2-distinguishability of $O(t^{-\alpha/2})$, and a μ_∞-distinguishability of $O(t^{-\alpha})$. Thus, we have exhibited a class for which our modified state merging algorithm outperforms the original by an arbitrary polynomial factor.

A natural problem for further research regards the efficient learnability of μ_p-distinguishable PDFA classes for the different values of p. We conjecture that for $p > 1$, these classes are indeed efficiently learnable.

Acknowledgements

We thank Alex Smola for insightful discussions. National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program.

References

T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 259–269. IEEE Computer Society, 2000.

R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *Theoret. Inform. and Appl.*, 33(1): 1–20, 1999.

A. Clark and F. Thollard. PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004.

T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.

J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Massachusetts, first edition, 1979.

M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proc. 25th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 392–401. ACM Press, New York, NY, 1993.

M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proc. 26th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 273–282, 1994.

K. Murphy. Passively learning finite automata. Technical report, Santa Fe Institute, 1996.

D. Ron. Property testing. In S. Rajasekaran, P. Pardalos, J. Reif, and J. Rolim, editors, *Handbook of Randomized Computing*, volume II, pages 597–649. Kluwer Academic, 2001.

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 31–40. ACM Press, New York, NY, 1995.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – Part I. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2005a. to appear.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – Part II. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2005b. to appear.

A Pseudocode for the State Merging Algorithm

Pseudocode for the state merging algorithm is provided in Algorithm A below. The algorithm is guaranteed to learn (with high probability) a PDFA class $\mathcal{H}$, for which a matching oracle is provided.

The number of oracle queries performed at each step of the state merging algorithm is upper bounded by $n^2|\Sigma|$, as there are at most n nodes in the graph at any time and at most $n|\Sigma|$ candidate nodes. When the algorithm runs correctly there are at most $n|\Sigma|+2$ steps. Therefore, over a complete run of the algorithm the number of oracle calls is at most $n^2|\Sigma|(n|\Sigma|+2)$.

B Proof Sketch of Theorem 1

We present a proof sketch using the notation of Clark and Thollard (2004), and provide exact references to the original proof where appropriate. We begin by decomposing our proof into the following *modules*:

(i) Given sufficiently many samples, a sample set is likely to be "good". Namely, *every* string will appear with an empirical probability that is close to its actual probability.
(ii) Assuming an oracle which matches the PDFA family under consideration, at each step the hypothesis graph will be isomorphic to a subgraph of the target with high probability.
(iii) With high probability, when the algorithm stops drawing samples, there will be in the hypothesis graph a state representing each frequent state in the target. In addition, all frequent transitions will also have a representative edge in the graph.
(iv) After the algorithm terminates, (again with high probability) all transition probability estimates will be close to their correct target values.
(v) A KL-PAC result between target and hypothesis is derived using the previous modules.

The following definitions in Clark and Thollard (2004) are redefined for the purpose of our proof:

The definition of a *candidate node* is replaced (or rather generalized) by the following:

Algorithm 1. State merging with oracle

```
Input: $\Sigma$ (input alphabet), $L$ (upper bound on the expected length of strings
       generated from any state), $n$ (upper bound on the number of states),
       $\mathcal{O}_{\mathcal{H}}$ (a $(\delta_1, m_1)$-matching oracle for $\mathcal{H}$), $\delta$ (confidence parameter), $\varepsilon$
       (precision parameter). The algorithm is also supplied with a random
       source of strings generated independently by $A$, the target PDFA.
Output: $\widehat{A}$, a hypothesis PDFA such that $\mathrm{KL}(A||\widehat{A}) < \varepsilon$ with probability at
        least $1 - \delta$.
Data: The algorithm maintains a graph $G = (V, E)$ with labelled edges (i.e.
      $E \subseteq V \times \Sigma \times V$), which holds the current hypothesis about the structure
      of the target automaton.
repeat
    Draw $N$ strings from $A$
    foreach $u \in V$ and $\sigma \in \Sigma$ which does not yet label an edge out of $u$ do
        Hypothesize a candidate node, referred to as $(u, \sigma)$
        Compute $S_u$ (multiset of suffixes of this node)
        if $|S_u| \geq m_0$ then
            foreach $v \in V$ do
                Query $\mathcal{O}_{\mathcal{H}}$ to compare $S_{u,\sigma}$ with $S_v$
                if $\mathcal{O}_{\mathcal{H}}$ returns ACCEPT then
                    Add arc labelled with $\sigma$ from $u$ to $v$.
                end
            end
            if $\mathcal{O}_{\mathcal{H}}$ returns REJECT on all comparisons then
                Create new node to graph $G$
                Add an edge labelled with $\sigma$ from $u$ to the new node
            end
        end
    end
until no candidate node has a sufficiently large suffix multiset
Complete $G$ by adding a ground node which represents low frequency states
Add a final state $\hat{q}_f$ and transitions labelled with $\zeta$ from each state to $\hat{q}_f$
```

Definition 5 (Candidate node). *A candidate node is a pair (u, σ) where u is a node in the graph G underlying the current hypothesis PDFA, and $\sigma \in \Sigma$ where $\tau(u, \sigma)$ is undefined. It will have an associated suffix multiset $S_{u,\sigma}$. A candidate node (u, σ) and a node v in a hypothesis graph G are* similar *if and only if the matching oracle $\mathcal{O}_{\mathcal{H}}$ returns* ACCEPT *when queried with $\hat{S}_{u,\sigma}$ and $\hat{S}_v$, where $\hat{S}$ denotes the empirical distribution induced by a multiset S.*

Definition 7 of a *good multiset* is now relaxed, and the original condition $L_\infty(\hat{S}, P_q^A) < \mu/4$ (which guaranteed distinguishability) is now dropped:

Definition 6 (good multiset). *A multiset S is ε_1-good for a state q iff for every $\sigma \in \Sigma \cup \{\zeta\}$, $|(S(\sigma)/|S|) - \gamma(q, \sigma)| < \varepsilon_1$.*

The threshold m_0 on the minimal multiset size for testing distribution proximity now becomes $m_0 = \max(m_1, m_2)$, where m_1 is the number of samples required by the matching oracle $\mathcal{O}_{\mathcal{H}}$ to guarantee an error probability of at most

δ_1, and m_2 is the multiset size shown in Clark and Thollard (2004) to guarantee a good sample (according to Definition 6 above) with probability at least $1-\delta_2$:

$$m_2 = \frac{1}{2\varepsilon_1^2}\log\left(\frac{24n|\Sigma|(|\Sigma|+1)(n|\Sigma|+2)}{\delta_2}\right), \quad \text{with} \tag{1}$$
$$\varepsilon_1 = \frac{\varepsilon^2}{16(|\Sigma|+1)(L+1)^2}.$$

The number of samples drawn at each iteration of the algorithm now becomes:

$$N = \frac{4n|\Sigma|L^2(L+1)^3}{\varepsilon_3^2}\max\left(2n|\Sigma|m_0, 4\log\frac{2(n|\Sigma|+2)}{\delta}\right), \quad \text{with}$$
$$\varepsilon_3 = \frac{\varepsilon}{2(n+1)\log\left(4(L+1)(|\Sigma|+1)/\varepsilon\right)}.$$

Rigorous statements and proofs of modules (i), (iii) and (iv) are given in Clark and Thollard (2004). References to the relevant sections follow.

The "good sample" concept of module (i) is defined in Definitions 7, 8, 10 and 11 of Sect. 4.2. Note that the original definition of a good multiset is now relaxed to Definition 6 above. The probability of obtaining a good sample is lower-bounded in Sect(s). 6.1 and 6.2. Specifically, using Chernoff bounds it is shown that for a sample size of m_2 the condition of Definition 6 is assured with probability of error less than $e^{-2m_2\varepsilon_1^2}$, which equals $\frac{\delta_2}{24n|\Sigma|(|\Sigma|+1)(n|\Sigma|+2)}$.

Module (iii) is proven in Lemmas 12 and 13 of Sect(s). 4.2 and 4.4 respectively, and our proof requires no changes. Module (iv) is discussed in Sect. 4.3, where again no changes are necessary. The modifications to module (ii) are addressed in the following lemma:

Lemma 2. *Given a PDFA class $\mathcal{H}$ and a (δ_1, m_1)-matching oracle $\mathcal{O}_{\mathcal{H}}$, with probability at least $1-\delta_1 n^2|\Sigma|$, at each step of the algorithm the hypothesis graph will be isomorphic to a subgraph of the target PDFA.*

Proof. The algorithm queries $\mathcal{O}_{\mathcal{H}}$ at most $n^2|\Sigma|$ times at each step. By applying the union bound and using the definition of a matching oracle we obtain the lemma. □

Finally, we prove module (v) and derive a KL-PAC bound. For the original μ-distinguishable class, a KL-PAC bound is proved by Clark and Thollard (2004). The authors show that assuming a good sample had been drawn, the KL-PAC bound follows. In our framework, an additional degree of randomness is present due to the probabilistic nature of the oracle. However, if this probability of error is managed, the same KL-divergence bound between target and hypothesis PDFA (namely ε) follows.

By setting

$$\delta_1 = \frac{\delta}{2n^2|\Sigma|(n|\Sigma|+2)}, \tag{2a}$$

$$\delta_2 = \delta/2, \tag{2b}$$

using multisets of size $m_0 = \max(m_1, m_2)$, and applying the union bound, the probability of error obtained is not greater than δ, and we retain the ε accuracy. The proof of Theorem 1 follows.

C Pseudocode for the L_2 Distance Test

Pseudocode for the L_2 distance test is provided in Algorithm C below. r_D denotes the number of self-collisions in the set F_D, namely the count of $i < j$ such that the i^{th} sample in F_D is same as the j^{th} sample in F_D. Similarly, $c_{D_1 D_2}$, the number of collisions between D_1 and D_2 is the count of (i, j) such that the i^{th} sample in D_1 is same as the j^{th} sample in D_2.

Algorithm 2. L_2-Distance-Test

```
Input: $D_1, D_2, m, \varepsilon, \delta$
Result: ACCEPT or REJECT
repeat
    Let $F_{D_1}$ = a set of $m$ samples from $D_1$
    Let $F_{D_2}$ = a set of $m$ samples from $D_2$
    Let $r_{D_1} = |F_{D_1} \cap F_{D_1}|$ (the number of self-collisions in $F_{D_1}$)
    Let $r_{D_2} = |F_{D_2} \cap F_{D_2}|$
    Let $Q_{D_1}$ = a set of $m$ samples from $D_1$
    Let $Q_{D_2}$ = a set of $m$ samples from $D_2$
    Let $c_{D_1 D_2} = |Q_{D_1} \cap Q_{D_2}|$
    Let $r = \frac{2m}{m-1}(r_{D_1} + r_{D_2})$
    Let $s = 2c_{D_1 D_2}$
    if $r - s > m^2 \varepsilon^2 / 2$ then REJECT else ACCEPT
until $O\left(\log\left(\frac{1}{\delta}\right)\right)$ iterations
REJECT if the majority of iterations reject
ACCEPT otherwise
```