Learning Finite Automata Using Label Queries

Dana Angluin[1], Leonor Becerra-Bonache[1,2*], Adrian Horia Dediu[2,3], and Lev Reyzin[1**]

[1] Department of Computer Science, Yale University
51 Prospect Street, New Haven, CT, USA
{dana.angluin,leonor.becerra-bonache,lev.reyzin}@yale.edu
[2] Research Group on Mathematical Linguistics, Rovira i Virgili University
Avinguda Catalunya, 35, 43002, Tarragona, Spain
[3] "Politehnica" University of Bucharest
Splaiul Independentei 313, 060042, Bucharest, Romania
adrianhoriadediu@yahoo.com

Abstract. We consider the problem of learning a finite automaton M of n states with input alphabet X and output alphabet Y when a teacher has helpfully or randomly labeled the states of M using labels from a set L. The learner has access to label queries; a label query with input string w returns both the output and the label of the state reached by w. Because different automata may have the same output behavior, we consider the case in which the teacher may "unfold" M to an output equivalent machine M' and label the states of M' for the learner. We give lower and upper bounds on the number of label queries to learn the output behavior of M in these different scenarios. We also briefly consider the case of randomly labeled automata with randomly chosen transition functions.

1 Introduction

The problem of learning the behavior of a finite automaton has been considered in several domains, including language learning and environment learning by robots. Many interesting questions remain about the kinds of information that permit efficient learning of finite automata.

One basic result is that finite automata are not learnable using a polynomial number of membership queries. Consider a "password machine", that is, an acceptor with $(n+2)$ states that accepts exactly one binary string of length n; the learner may query (2^n-1) strings before finding the one that is accepted. In this case, the learner gets no partial information from the unsuccessful queries.

However, Freund et al. [5] show that regardless of the topology of the underlying automaton, if its states are randomly labeled with 0 or 1, then a robot

* Supported by a Marie Curie International Fellowship within the 6th European Community Framework Programme.

** This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

taking a random walk on the automaton can learn to predict the labels while making only a polynomial number of errors of prediction. Random labels on the states provide a rich source of information that can be used to distinguish otherwise difficult-to-distinguish pairs of states.

In a different setting, Becerra-Bonache, Dediu and Tîrnăucă [3] introduced **correction queries** to model a kind of correction provided by a teacher to a learner when the learner's utterance is not grammatically correct. In their model, a correction query with a string w gives the learner not only membership information about w, but also, if w is not accepted, either the minimum continuation of w that is accepted, or the information that no continuation of w is accepted. In certain cases, corrections may provide a substantial amount of partial information for the learner. For example, for a password machine, a prefix of the password will be answered with the rest of the password. We may think of correction queries as labeling each state q of the automaton with the string r_q that is the response to any correction query w that arrives at q.

In both of these cases, labels on states may facilitate the learning of finite automata: randomly chosen labels in the work of Freund et al. and meaningfully chosen labels in the work of Becerra-Bonache, Dediu and Tîrnăucă. In this paper we explore the general idea of adding labels to the states of an automaton to make it easier to learn. That is, we allow a teacher to prepare an automaton M for learning by adding labels to its states (either carefully or randomly chosen). When the learner queries a string, the learner receives not only the original output of M for that string, but also the label attached to that state by the teacher. In an extension of this idea, we also allow the teacher to "unfold" the machine M to produce copies of a state that may then be given different labels. These ideas are also relevant to automata testing [7] – labeling and unfolding automata can make their structure easier to verify.

Depending on how labels are assigned, learning may or may not become easier. If each state is assigned a unique label, the learning task becomes easy because the learner knows which state the machine reaches on any given query. However, if the labels are all the same, they give no additional information and learning may require an exponential number of queries (as in the case of membership queries.)

Hence we focus on questions of the following sort. Given an automaton, how can a teacher use a limited set of labels to make the learning problem easier? If random labels are sprinkled on the states of an automaton, how much does that help the learner? How few labels can we use and still make the learning problem tractable? Other questions concern the structure of the automaton itself. For example, we may consider changing the structure of the automaton before labeling it. We also consider the problem of learning randomly labeled automata with random structure.

2 Preliminaries

We consider finite automata with output, defined as follows. A finite automaton M has a finite set Q of states, an initial state $q_0 \in Q$, a finite alphabet X of input symbols, a finite alphabet Y of output symbols, an output function γ mapping Q to Y and a transition function τ mapping $Q \times X$ to Q. We extend τ to map $Q \times X^*$ to Q in the usual way. A finite acceptor is a finite automaton with output alphabet $Y = \{0, 1\}$; if $\gamma(q) = 1$ then q is an accepting state, otherwise, q is a rejecting state. In this paper we assume that there are at least two input symbols and at least two output symbols, that is, $|X| \geq 2$ and $|Y| \geq 2$.

For any string $w \in X^*$, we define $M(w)$ to be $\gamma(\tau(q_0, w))$, that is, the output of the state reached from q_0 on input w. Two finite automata M_1 and M_2 are **output-equivalent** if they have the same input alphabet X and the same output alphabet Y and for every string $w \in X^*$, $M_1(w) = M_2(w)$.

If M is a finite automaton with output, then an **output query** with string $w \in X^*$ returns the symbol $M(w)$. This generalizes the concept of a membership query for an acceptor. That is, if M is an acceptor, an output query with w returns 1 if w is accepted by M and 0 if w is rejected by M. We note that Angluin's polynomial time algorithm to learn finite acceptors using membership queries and equivalence queries generalizes in a straightforward way to learn finite automata with output using output queries and equivalence queries [2].

If q_1 and q_2 are states of a finite automaton with output, then q_1 and q_2 are **distinguishable** if there exists a **distinguishing string** for them, namely, a string w such that $\gamma(\tau(q_1, w)) \neq \gamma(\tau(q_2, w))$, that is, w leads from q_1 and q_2 to two states with different output symbols. If M is minimized, every pair of its states are distinguishable, and M has at most one sink state.

If d is a nonnegative integer, the d**-signature tree** of a state q is the finite function mapping each input string z of length at most d to the output symbol $\gamma(\tau(q, z))$. We picture the d-signature tree of a state as a rooted tree of depth d in which each internal node has $|X|$ children labeled with the elements of X, and each node is labeled with the symbol from Y that is the output of the state reached from q on the input string z that leads from the root to this node. The d-signature tree of a state gives the output behavior in a local neighborhood of the automaton reachable from that state.

For any finite automaton M with output, we may consider its **transition graph**, which is a finite directed graph (possibly with multiple edges and self-loops) defined as follows. The vertices are the states of M and there is an edge from q to q' for each transition $\tau(q, a) = q'$. Properties of the transition graph are applied to M; that is, M is **strongly connected** if its transition graph is strongly connected. Similarly, the **out-degree** of M is $|X|$ for every node, and the **in-degree** of M is the maximum number of edges entering any node of its transition graph. For a positive integer k, we define an automaton M to be k**-concentrating** if there is some set Q' of at most k states of M such that every state of M can reach at least one state in Q'. Every strongly connected automaton is 1-concentrating.

2.1 Labelings

If M is a finite automaton with output, then a **labeling** of M is a function ℓ mapping Q to a set L of labels, the **label alphabet**. We use M to construct a new automaton M^ℓ by changing the output function to $\gamma'(q) = (\gamma(q), \ell(q))$. That is, the new output for a state is a pair incorporating the output symbol for the state and the label attached to the state. For the scenario of **learning with labels**, we assume that the learner has access to output queries for M^ℓ for some labeling ℓ of the hidden automaton M. For the scenario of **learning with unfolding and labels**, we assume that the learner has access to output queries for M_1^ℓ for some labeling ℓ of some automaton M_1 that is output-equivalent to M. In these two scenarios, the queries will be referred to as **label queries**. The goal of the learner in either scenario is to use label queries to find a finite automaton M' output-equivalent to M. Thus, the learner must discover the output behavior of the hidden automaton, but not necessarily its topology or labeling. We assume the learner is given both X and $|Q|$.

3 Learning with Labels

First, we show a lower bound on the number of label queries required to learn a hidden automaton M with n states and an arbitrary labeling ℓ.

Proposition 1. *Let L be a finite label alphabet. Learning a hidden automaton with n states and a labeling ℓ using symbols from L requires*

$$\Omega\left(\frac{|X|n\log(n)}{1+\log(|L|)}\right)$$

label queries in the worst case.

Proof. Recall that we have assumed that $|X|$ and $|Y|$ are both at least 2; we consider $|Y| = 2$. Domaratzki, Kisman and Shallit [4] have shown that there are at least

$$(|X| - o(1))n2^{n-1}n^{(|X|-1)n}$$

distinct languages accepted by acceptors with n states. Because each label query returns one of at most $2 \cdot |L|$ values, an information theoretic argument gives the claimed lower bound on the number of label queries. As a corollary, when $|X|$ and $|L|$ are constants, we have a lower bound of $\Omega(n\log(n))$ label queries. $\square$

3.1 Labels Carefully Chosen

In this section, we examine the case where the teacher is given a limit on the number of different labels he may use, and he is able to label the states after examining the automaton. Moreover, the learning algorithm may take advantage of knowing the labeling strategy of the teacher. In this setting the problem takes on an aspect of coding, and indicates the maximum extent to which labeling may facilitate efficient learning. We begin with a simple proposition.

Proposition 2. *An automaton with n states, helpfully labeled using n different labels, can be learned using $|X|n$ label queries.*

Proof. The teacher assigns a unique integer label between 1 and n to each state. The learner asks a label query with the empty string to determine the output and label of the start state, and then explores the transitions from the start state by querying each $a \in X$. After querying an input string w, the label indicates whether this state has been visited before. If the state is new, the learner explores all the transitions from it by querying wa for each $a \in X$. Thus, after querying at most $|X|n$ strings, the learner knows the structure and outputs of the entire automaton. The lower bound shows that this is asymptotically optimal if the label set L has n elements. $\square$

We next consider limiting the teacher to a constant number of different labels: a polynomial number of label queries suffices in this case.

Theorem 1. *For each automaton with n states, there is a helpful labeling using $2^{|X|}$ different labels such that the automaton can be learned using $O(|X|n^2)$ label queries.*

Proof. Given an automaton M of n states, the teacher chooses an outward-directed spanning tree T rooted at q_0 of the transition graph of the automaton, and labels the states of M to communicate T to the learner as follows. The label of state q is the subset of X corresponding to the edges of T from q to other nodes. The label of q directs the learner to q's children. Using at most n label queries and the structure of T, the learner can create a set S of n input strings such that for each state q of M, there is one string $w \in S$ such that $\tau(q_0, w) = q$.

In [1], Angluin gives an algorithm for learning a regular language using membership queries given a live complete sample for the language. A live complete sample for a language L is a set of strings P, that for every state q (other than the dead state) of the minimal acceptor for L, contains a string that leads from the start state to q. Given a live complete sample P, a learner can find the regular language using $O(k|P|n)$ membership queries, where k is the size of the input alphabet. A straightforward generalization of this algorithm to automata with output shows that the set S and $O(|X|n^2)$ output queries can be used to find an automaton output equivalent to M. $\square$

However, the number of queries, $O(n^2)$, does not meet the $\Omega(n \log n)$ lower bound, and the number of different labels is large. For a restricted class of automata, there is a helpful labeling with fewer labels that permits learning with an asymptotically optimal $O(n \log n)$ label queries. To appreciate the generality of Theorem 2, we note once more that every strongly connected automaton is 1-concentrating, and as we will see in Lemma 1, automata with a small input alphabet can be unfolded to have small in-degree.

Theorem 2. *Let k and c be positive integers. Any automaton in the class of c-concentrating automata with in-degree at most k can be helpfully labeled with at most $(3k|X| + c)$ labels so that it can be learned using $O(|X|n \log(n))$ label queries.*

Proof. We give the construction for 1-concentrating automata and indicate how to generalize it at the end of the proof. Given a 1-concentrating automaton M the teacher chooses as the *root* a node reachable from all other nodes in the transition graph of M. The depth of a node is the length of the shortest path from that node to the root. The teacher then chooses a spanning tree T directed inward to the root by choosing a parent for each non-root node. (One way to do this is to let the parent of a node q be the first node reached along a shortest path from q to the root.) The teacher assigns, as part of the label for each node q, an element $a \in X$ such that $\tau(q, a)$ is the parent of q.

The teacher now adds more information to the labels of the nodes, which we call color, using the colors yellow, red, green, and blue. The root is the unique node colored yellow. Let $t = \lceil \log n \rceil$; t bits are enough to give a unique identifier for every node of the graph. Each node at depth a multiple of $(t + 1)$ is colored red. For each red node v we choose a unique identifier of t bits $(c_1, c_2, \ldots, c_t)$ encoded as green and blue labels. Now consider the maximal subtree rooted at v containing no red nodes. For each level i from 1 to the depth of the subtree, all the nodes at level i of the subtree are colored with c_i (which is either blue or green.) The teacher has (so far) used $3|X| + 1$ labels – a direction and one of three colors per non-root node, and a unique identifier for the root.

Given this labeling, the learner can start from any state and reach a localization state whose identifier is known, as follows. The learner uses the parent component of the labels to go up the tree until it passes one red node and arrives at a second red node, or arrives at the root (whichever comes first), keeping track of the labels seen. If the learner reaches the root, it knows where it is. Otherwise, the learner interprets the labels seen between the first and second red node encountered as an identifier for the node v reached. This involves observing at most $(2t+2)$ labels. Thus, even if the in-degree is not bounded, a 1-concentrating automaton can be labeled so that with $O(\log(n))$ label queries the learner can reach a uniquely identified localizing state.

If each node of the tree T also has in-degree bounded by k, another component of the label for each non-root node identifies which of the k possible predecessors of its parent it is (numbered arbitrarily from 1 to at most k.) If the learner collects these values on the path from u to its localization node v, then we have an identifier for u with respect to v. Thus it takes $O(\log(n))$ label queries to learn any node's identifier. If the node has not been encountered before, its $|X|$ transitions must be explored, as in Proposition 2. This gives us a learning algorithm using $O(|X|n \log(n))$ label queries. The labeling uses at most $3k|X|+1$ different labels.

If the automaton is c-concentrating for some $c > 1$, then the teacher selects a set of at most c nodes such that every node can reach at least one of them and constructs a forest of at most c inward directed disjoint spanning trees, and proceeds as above. This increases the number of unique identifiers for the roots from 1 to c. $\square$

An open question is whether an arbitrary finite automaton with n states can be helpfully labeled with $O(1)$ labels in such a way that it can be learned using $O(|X|n \log n)$ label queries.

3.2 Labels Randomly Chosen

In this section we turn from labels carefully chosen by the teacher to an independent uniform random choice of labels for states from a label alphabet L. With nonzero probability the labeling may be completely uninformative, so results in this scenario incorporate a confidence parameter $\delta > 0$ that is an input to the learner. The goal of the learner is to learn an automaton that is output equivalent to the hidden automaton M with probability at least $(1 - \delta)$, where this probability is taken over the labelings of M. Results on random labelings can be used in the careful labeling scenario: the teacher generates a number of random labelings until one is found that has the desired properties.

We first review the learning scenario considered by Freund et al. [5]. There is a finite automaton over the input alphabet $X = \{0, 1\}$ and output alphabet $\{+, -\}$, where the transition function and start state of the automaton are arbitrary, but the output symbol for each state is chosen independently and uniformly from $\{+, -\}$. The learner moves from state to state in the target automaton according to a random walk (the next input symbol is chosen independently and uniformly from $\{0, 1\}$) and, after learning what the next input symbol will be, attempts to predict the output ($+$ or $-$) of the next state. After the prediction, the learner is told the correct output and the process repeats with the next input symbol in the random walk. If the learner's prediction was incorrect, this counts as a **prediction mistake**. In the first scenario they consider, the learner may reset the machine to the initial state by predicting ? instead of $+$ or $-$; this counts as a **default mistake**. In this model, the learner is completely passive, dependent upon the random walk process to disclose useful information about the behavior of the underlying automaton. For this setting they prove the following.

Theorem 3 (Freund et al. [5]). *There exists a learning algorithm that takes n and δ as input, runs in time polynomial in n and $1/\delta$ and with probability at least $(1 - \delta)$ makes no prediction mistakes and an expected $O((n^5/\delta^2) \log(n/\delta))$ default mistakes.*

The main idea is to use the d-signature tree of a state as the identifier for the state, where $d \geq 2 \log(n^2/\delta)$. For this setting, there are at least n^4/δ^2 strings in a signature tree of depth d. The following theorem of Trakhtenbrot and Barzdin' [8] establishes that signature trees of this depth are sufficient.

Theorem 4 (Trakhtenbrot and Barzdin' [8]). *For any natural number d and for any finite automaton with n states and randomly chosen outputs from Y, the probability that for some pair of distinguishable states the shortest distinguishing string is of length greater than d is less than*

$$n^2(1/|Y|)^{d/2}.$$

We may apply these ideas to prove the following.

Theorem 5. *For any positive integer s, any finite automaton with n states, over the input alphabet X and output alphabet Y, with its states randomly labeled with labels from a label alphabet L with $|L| = |X|^s$ can be learned using*

$$O\left(|X|\frac{n^{1+4/s}}{\delta^{2/s}}\right)$$

label queries, with probability at least $(1-\delta)$ (with respect to the choice of labeling.)

Proof. Assume that the learning algorithm is given n, a bound on the number of states of the hidden automaton, and the confidence parameter $\delta > 0$. It calculates a bound $d = d(n, \delta)$ (described below) and proceeds as follows, starting with the empty input string. To explore the input string w, the learning algorithm calculates the d signature tree (in the labeled automaton) of the state reached by w by making label queries on wz for all input strings z of length at most d. This requires $O(|X|^d)$ queries. If this signature tree has not been encountered before, then the algorithm explores the transitions wa for all $a \in X$. Assuming that the labeling is "good", that is, that all distinguishable pairs of states have a distinguishing string in the labeled automaton of length at most d, then this correctly learns the output behavior of the hidden automaton using $O(|X|^{d+1}n)$ label queries.

To apply Theorem 4, we assume that the hidden automaton M is an arbitrary finite automaton with output with at most n states, input alphabet X and output alphabet Y. The labels randomly chosen from L then play the role of the random outputs in Theorem 4. There is a somewhat subtle issue: states distinguishable in M by their outputs may not be distinguishable in the labeled automaton by their labels alone. Fortunately, Freund et al. [5] have shown us how to address this point. In the first case, if two states of M are distinguishable by their outputs in M by a string of length at most d, then their d signature trees (in the labeled automaton) will differ. Otherwise, if the shortest distinguishing string for the two states (using just outputs) is of length at least $d+1$, then generalizing the argument for Theorem 2 in [5] from $|Y| = 2$ to arbitrary $|Y|$, the probability that this pair of states is not distinguished by the random labeling by a string of length at most d is bounded above by $(1/|Y|)^{(d+1)/2}$. Summing over all pairs of states gives the required bound.

Thus, choosing

$$d \geq \frac{2}{\log |L|} \log\left(\frac{n^2}{\delta}\right),$$

suffices to ensure that the labeling is "good" with probability at least $(1-\delta)$. If we use more labels, the signature trees need not be so deep and the algorithm does not need to make as many queries to determine them. In particular, if $|L| = |X|^s$, then the bound of $O(|X|^{d+1}n)$ on the number of label queries used by the algorithm becomes

$$O\left(|X|\frac{n^{1+4/s}}{\delta^{2/s}}\right),$$

completing the proof. □

Corollary 1. *Any finite automaton with n states can be learned using $O(|X|n^{1+\epsilon})$ label queries with probability at least $1/2$, when it is randomly labeled with $|L| = f(|X|, \epsilon)$ labels.*

Proof. With $\delta = 1/2$ a choice of $|L| \geq |X|^{4/\epsilon}$ suffices. □

We remark that this implies that there exists a careful labeling with $O(|X|^4)$ labels that achieves learnability with $O(|X|n^2)$ label queries, substantially improving on the size of the label set used in Theorem 1. An open question is whether a random labeling with $O(1)$ labels enables efficient learning of an arbitrary n state automaton with $O(n \log n)$ queries with high probability.

4 Unfolding Finite Automata

We now consider giving more power to the teacher. Because many automata have the same output behavior, we ask what happens if a teacher can change the underlying machine (without changing its output behavior) before placing labels on it. In Sections 3.1 and 3.2, the teacher had to label the machine given to him. Now we will examine what happens when a teacher can unfold an automaton before putting labels on it. That is, given M, the teacher chooses another automaton M' with the same output behavior as M and labels the states of M' for the learner.

4.1 Unfolding and then Labeling

We first remark that unfolding an automaton M from n to $O(n \log n)$ states allows a careful labeling with just 2 labels to encode a description of the machine.

Proposition 3. *Any finite automaton with n states can be unfolded to have $N = O(|X|n\log(n) + n\log(|Y|))$ states and carefully labeled with 2 labels, in such a way that it can be learned using N label queries.* □

Proof. The total number of automata with output having n states, input alphabet X and output alphabet Y is at most

$$n^{|X|n+1}|Y|^n.$$

Thus, $N = O(|X|n\log(n) + n\log(|Y|))$ bits suffice to specify any one of these machines.

The teacher chooses $a \in X$ and unfolds the target automaton M as follows. The strings a^i for $i = 0, 1, \ldots, N-1$ each send the learner to a newly created state, which act (with respect to transitions on other input symbols and output behavior) just as their counterparts in the original machine. The remaining states are unchanged. The unfolded automaton is output equivalent to M. The teacher then specifies M using by labeling these N new states with the bits of the specification of M. The learner simply asks a sequence of N queries on strings of the form a^i to receive the encoding of the hidden machine. □

This method does not work if we restrict the unfolding to $O(|X|n)$ states, but we show that this much unfolding is sufficient to reduce the in-degree of the automaton to $O(|X|)$.

Lemma 1. *Let M be an arbitrary automaton of n states. There is an automaton M' with the same output behavior as M, with at most $(|X|+1)n$ states whose in-degree is bounded by $2|X|+1$.*

Proof. Given M, we repeat the following process until it terminates. While there is some state q with in-degree greater than $2|X|+1$, split q into two copies, dividing the incoming edges as evenly as possible between the two copies, and duplicating all $|X|$ outgoing edges for the second copy of q.

It is clear that each step of this process preserves the output behavior of M. To see that it terminates, for each node q let $f(q)$ be the maximum of 0 and $d_{in}(q) - (|X|+1)$, where $d_{in}(q)$ is the in-degree of q. Consider the potential function Φ that is the sum of $f(q)$ for all nodes q in the transition graph. Φ is initially at most $|X|n - (|X|+1)$, and each step reduces it by at least $1 = (|X|+1) - |X|$. Thus, the process terminates after no more than $|X|n$ steps producing an output-equivalent automaton M' with no more than $(|X|+1)n$ states and in-degree at most $2|X|+1$. □

In particular, an automaton with a sink state of high in-degree will be unfolded by this process to have multiple copies of the sink state. Using this idea for degree reduction, the teacher may use linear unfolding and helpful labeling to enable a strongly connected automaton to be learned with $O(n \log n)$ label queries.

Corollary 2. *For any strongly connected automaton M of n states, there is an unfolding M' of M with at most $(|X|+1)n$ states and a careful labeling of M' using $O(|X|^2)$ labels that allows the behavior of M to be learned using $O(|X|^2 n \log n)$ label queries.*

Proof. Given a strongly connected automaton M with n states, the teacher uses the method of Lemma 1 to produce an output equivalent machine M' with at most $(|X|+1)n$ states and in-degree bounded by $2|X|+1$. This unfolding may not preserve the property of being strongly connected, but there is at least one state q that has at most $(|X|+1)$ copies in the unfolded machine M'. Because M is strongly connected, every state of M' must be able to reach at least one of the copies of q, so M' is $(|X|+1)$-concentrating. Applying the method of Theorem 2, the teacher can use $3(2|X|+1)|X| + (|X|+1)$ labels to label M' so that it can be learned with $O(|X|^2 n \log n)$ label queries. □

We now consider uniform random labelings of the states when the teacher is allowed to choose the unfolding of the machine.

Theorem 6. *Any automaton with n states can be unfolded to have $O(n \log(n/\delta))$ states and randomly labeled with 2 labels, such that with probability at least $(1-\delta)$, it can be learned using $O(|X|n(\log(n/\delta))^2)$ queries.*

Proof. Given n and δ, let $t = \lceil \log(n^2/\delta) \rceil$. The teacher chooses $a \in X$ and unfolds the target machine M to construct the machine M' as follows. M' has nt states (q, i) where q is a state of M and $0 \leq i \leq (t-1)$. The start state is $(q_0, 0)$, where q_0 is the start state of M. The output symbol for (q, i) is $\gamma(q, a^i)$, where γ is the output function of M. For $0 < i < (t-1)$, the a transition from (q, i) is to $(q, (i+1))$. The a transition from $(q, t-1)$ is to $(q', 0)$, where $q' = \tau(q, a^t)$ and τ is the transition function of M. For all other input symbols b with $b \neq a$, the b transition from (q, i) is to $(q', 0)$, where $q' = \tau(q, a^i b)$.

To see that M' is an unfolding of M, that is, M' is output equivalent to M, we show that each state (q, i) of M' is output equivalent to state $\tau(q, a^i)$ of M. By construction, these two states have the same output. If $i < (t-1)$ then the a transition from (q, i) is to $(q, i+1)$, which has the same output symbol as $\tau(q, a^{i+1})$. The a transition from $(q, t-1)$ is to $(q', 0)$, where $q' = \tau(q, a^t)$, which has the same output symbol as $\tau(\tau(q, a^{t-1}), a)$. If $b \neq a$ is an input symbol, then the b transition from (q, i) is to $(q', 0)$ where $q' = \tau(q, a^i b)$, which has the same output symbol as $\tau(\tau(q, a^i), b)$.

Suppose M' is randomly labeled with two labels. For each state q of M, define its label identifier in M' to be the sequence of labels of (q, i) for $i = 0, 1, \ldots, (t-1)$. For two distinct states q_1 and q_2 of M, the probability that their label identifiers in M' are equal is $(1/2)^t$, which is at most δ/n^2. Thus, the probability that there exist two distinct states q_1 and q_2 with the same label identifier in M' is at most δ.

Given n and δ, the learning algorithm takes advantage of the known unfolding strategy to construct states (j, i) for $0 \leq j \leq n-1$ and $0 \leq i \leq (t-1)$ with a transitions from (j, i) to $(j, i+1)$ for $i < (t-1)$. It starts with the empty input string and uses the following exploration strategy. Given an input string w that is known to arrive at some $(q, 0)$ in M', the learning algorithm makes label queries on wa^i for $i = 0, 1, \ldots, (t-1)$ to determine the label identifier of q in M'. If this label identifier has not been seen before, the learner uses the next unused $(j, 0)$ to represent q and records the outputs and labels for the states (j, i) for $i = 0, 1, \ldots, (t-1)$. It must also explore all unknown transitions from the states (j, i). If distinct states of M receive distinct label identifiers in M', the learner learns a finite automaton output equivalent to M using $O(|X|nt^2)$ label queries. $\square$

5 Automata with Random Structure

We may also ask whether randomly labeled finite automata are hard to learn "on average". We consider automata with randomly chosen transition functions and random labels. The model of random structure that we consider is as follows. Let the states be q_i for $i = 0, 1, \ldots, (n-1)$, where q_0 is the start state. For each state q_i and input symbol $a \in X$, choose j uniformly at random from $0, 1, \ldots, (n-1)$ and let $\tau(q_i, a) = q_j$.

Theorem 7. *A finite automaton with n states, a random transition function and a random labeling can be learned using $O(n\log(n))$ label queries, with high probability. The probability is over the choice of transition function and labeling.*

Proof. This was first proved by Korshunov in [6]; here we give a simpler proof. Korshunov showed that the signature trees only need to be of depth asymptotically equal to $\log_{|X|}(\log_{|L|}(n))$ for the nodes to have unique signatures with high probability. We use a method similar to signature trees, but simpler to analyze. Instead of comparing signature trees for two states to tell whether or not they are distinct, we compare the labels along at most four sets of transitions, which we call **signature paths** – like a signature tree consisting only of four paths.

Lemmas 2 and 3 show that given X and n there are at most four signature paths, each of length $3\log(n)$, such that for a random finite automaton of n states with input alphabet X and for any pair s_1 and s_2 of different states, the probability is $O\left(\frac{\log^6(n)}{n^3}\right)$ that s_1 and s_2 are distinguishable but not distinguished by any of the strings in the four signature paths. By the union bound, the probability that there exist two distinguishable states that are not distinguished by at least one of the strings in the four signature paths is at most

$$\binom{n}{2}\left(O\left(\frac{\log^6(n)}{n^3}\right)\right) = o(1).$$

Hence, by running at most four signature paths, each of length $3\log(n)$, per newly reached state, we get unique labels on the states. Then for each of the n states, we can find their $|X|$ transitions, and learn the machine, as in Proposition 2. □

We now turn to the two lemmas used in the proof of Theorem 7. We first consider the case $|X| > 2$. If $a, b, c \in X$ and ℓ is a nonnegative integer, let $D_\ell(a, b, c)$ denote the set of all strings a^i, b^i, and c^i such that $0 \le i \le \ell$.

Lemma 2. *Let s_1 and s_2 be two different states in a random automaton with $|X| > 2$. Let $a, b, c \in X$ and $\ell = 3\log(n)$. The probability that s_1 and s_2 are distinguishable, but not by any string in $D_\ell(a, b, c)$ is $O\left(\frac{\log^6(n)}{n^3}\right)$.*

Proof. We analyze the three (attempted) paths from two states s_1 and s_2, which we will call $\pi^1_{s_1}, \pi^2_{s_1}, \pi^3_{s_1}$ and $\pi^1_{s_2}, \pi^2_{s_2}, \pi^3_{s_2}$, respectively. Each path will have length $3\log(n)$. We define each of the π_i as a set of nodes reached by its respective set of transitions.

We first look at the probability that the following event does not happen: that both $|\pi^1_{s_1}| > 3\log(n)$ and $|\pi^1_{s_2}| > 3\log(n)$, and that $\pi^1_{s_1} \cap \pi^1_{s_2} = \emptyset$, that is the probability that both of these strings succeed in reaching $3\log(n)$ different states, and that they share no states in common. We call the event that two sets of states π_1 and π_2 have no states in common, and both have size at least

l, $S(\pi_1, \pi_2, l)$ (success) and the failure event $F(\pi_1, \pi_2, l) = \overline{S(\pi_1, \pi_2, l)}$. So,

$$\begin{aligned} P(F(\pi^1_{s_1}, \pi^1_{s_2}, 3\log(n))) &\leq \sum_{i=1}^{3\log(n)} \left(\frac{i + |\pi^1_{s_1}|}{n}\right) + \sum_{i=1}^{3\log(n)} \left(\frac{i + |\pi^1_{s_2}|}{n}\right) \\ &\leq 2 \sum_{i=1}^{3\log(n)} \left(\frac{i + 3\log(n)}{n}\right) \\ &= O\left(\frac{\log^2(n)}{n}\right). \end{aligned}$$

Now we look at the probability that $F(\pi^2_{s_1}, \pi^2_{s_2}, 3\log(n))$ given that we failed on the first paths, or $F(\pi^1_{s_1}, \pi^1_{s_2}, 3\log(n))$, with $l = 3\log(n)$,

$$\begin{aligned} P\left(F(\pi^2_{s_1}, \pi^2_{s_2}, l) | F(\pi^1_{s_1}, \pi^1_{s_2}, l)\right) &\leq \sum_{i=1}^{3\log(n)} \left(\frac{i + |\pi^2_{s_1}| + |\pi^1_{s_1}| + |\pi^1_{s_2}|}{n}\right) \\ &\quad + \sum_{i=1}^{3\log(n)} \left(\frac{i + |\pi^2_{s_2}| + |\pi^1_{s_1}| + |\pi^1_{s_2}|}{n}\right) \\ &\leq 2 \sum_{i=1}^{3\log(n)} \left(\frac{i + 9\log(n)}{n}\right) \\ &= O\left(\frac{\log^2(n)}{n}\right). \end{aligned}$$

Now, we will compute the probability that $F(\pi^3_{s_1}, \pi^3_{s_2}, 3\log(n))$ given failures on the previous two pairs of states. Let $l = 3\log(n)$,

$$\begin{aligned} P\left(F(\pi^3_{s_1}, \pi^3_{s_2}, l) | F(\pi^1_{s_1}, \pi^1_{s_2}, l), F(\pi^2_{s_1}, \pi^2_{s_2}, l)\right) &\leq 2 \sum_{i=1}^{3\log(n)} \left(\frac{i + 25\log(n)}{n}\right) \\ &= O\left(\frac{\log^2(n)}{n}\right). \end{aligned}$$

Last, we compute the probability none of these pairs of paths made it to $l = 3\log(n)$, or $P(\text{failure}) = P\left(F(\pi^1_{s_1}, \pi^1_{s_2}, l), F(\pi^2_{s_1}, \pi^2_{s_2}, l), F(\pi^3_{s_1}, \pi^3_{s_2}, l)\right)$

$$\begin{aligned} P(\text{failure}) &= P(F(\pi^1_{s_1}, \pi^1_{s_2}, l)) \cdot P\left(F(\pi^2_{s_1}, \pi^2_{s_2}, l) | F(\pi^1_{s_1}, \pi^1_{s_2}, l)\right) \cdot \\ &\quad P\left(F(\pi^3_{s_1}, \pi^3_{s_2}, l) | F(\pi^1_{s_1}, \pi^1_{s_2}, l), F(\pi^2_{s_1}, \pi^2_{s_2}, 1)\right) \\ &= O\left(\frac{\log^2(n)}{n}\right) O\left(\frac{\log^2(n)}{n}\right) O\left(\frac{\log^2(n)}{n}\right) \\ &= O\left(\frac{\log^6(n)}{n^3}\right). \end{aligned}$$

Thus, given two distinct states with corresponding nonoverlapping signature paths of length $3\log(n)$, the probability that all of the randomly chosen labels along the paths will be the same is $2^{3\lg(n)} = \frac{1}{n^3} = O\left(\frac{\log^6(n)}{n^3}\right)$, which is the probability that no string in $D_\ell(a,b,c)$ distinguishes s_1 from s_2. □

When $|X| = 2$, we do not have enough alphabet symbols to construct three completely independent paths as in the proof of Lemma 2, but four paths suffice. If $a, b \in X$ and ℓ is a nonnegative integer, let $D_\ell(a,b)$ denote the set of all strings a^i, b^i, ab^i and ba^i such that $0 \leq i \leq \ell$.

Lemma 3. *Let s_1 and s_2 be two different states in a random automaton with $|X| = 2$. Let $a, b \in X$ and $\ell = 3\log(n)$. The probability that s_1 and s_2 are distinguishable, but not by any string in $D_\ell(a,b)$ is $O\left(\frac{\log^6(n)}{n^3}\right)$.*

The proof of Lemma 3 is a case analysis using reasoning similar to that of Lemma 2; we include an outline. If s_1 and s_2 are assigned different labels, then they are distinguished by the empty string, so assume that they are assigned the same label. If we consider $\tau(s_1,a)$ and $\tau(s_2,a)$, there are four cases, as follows. (1) We have $\tau(s_1,a) \neq \tau(s_2,a)$ and neither one is s_1 or s_2. In this case, an argument analogous to that in Lemma 2 shows that the probability that the paths a^i, ab^i and b^i fail to produce a distinguishing string for s_1 and s_2 is bounded by $O(\log^6(n)/n^3)$. (2) Exactly one of $\tau(s_1,a)$ and $\tau(s_2,a)$ is in the set $\{s_1,s_2\}$. This happens with probability $O(1/n)$, and in this case we can show that the probability that the paths a^i and b^i do not produce a distinguishing string for s_1 and s_2 is bounded by $O(\log^4(n)/n^2)$, for a total failure probability of $O(\log^4(n)/n^3)$ for this case. (3) Both of $\tau(s_1,a)$ and $\tau(s_2,a)$ are in the set $\{s_1,s_2\}$. This happens with probability $O(1/n^2)$, and in this case we can show that the probability that the path b^i does not produce a distinguishing string for s_1 and s_2 is bounded by $O(log^2(n)/n)$, for a total failure probability of $O(\log^2(n)/n^3)$ for this case. (4) Neither of $\tau(s_1,a)$ and $\tau(s_2,a)$ is in the set $\{s_1,s_2\}$, but $\tau(s_1,a) = \tau(s_2,a)$. This happens with probability $O(1/n)$, and we proceed to analyze four parallel subcases for $\tau(s_1,b)$ and $\tau(s_2,b)$.

(4a) We have $\tau(s_1,b) \neq \tau(s_2,b)$ and neither of them is in the set $\{s_1,s_2\}$. We can show that the probability that the paths b^i and ba^i do not produce a distinguishing string for s_1 and s_2 is bounded by $O(\log^4(n)/n^2)$, for a failure probability of $O(\log^4(n)/n^3)$ in this subcase, because the probability of case (4) is $O(1/n)$. (4b) Exactly one of $\tau(s_1,b)$ and $\tau(s_2,b)$ is in the set $\{s_1,s_2\}$. In this subcase, we can show that the probability that the path b^i fails to produce a distinguishing string for s_1 and s_2 is bounded by $O(\log^2(n)/n)$, for a total failure probability in this subcase of $O(\log^2(n)/n^3)$, because the probability of case (4) is $O(1/n)$ and the probability that one of $\tau(s_1,b)$ and $\tau(s_2,b)$ is in $\{s_1,s_2\}$ is $O(1/n)$. (4c) Both of $\tau(s_1,b)$ and $\tau(s_2,b)$ are in $\{s_1,s_2\}$. The probability of this happening is $O(1/n^2)$, for a total probability of this subcase of $O(1/n^3)$, because the probability of case (4) is $O(1/n)$. (4d) We have $\tau(s_1,b) = \tau(s_2,b)$. Then because we are in case (4), $\tau(s_1,a) = \tau(s_2,a)$ and the labels assigned s_1 and s_2 are equal, so the states s_1 and s_2 are equivalent and therefore indistinguishable.

Acknowledgments

We would like to thank the anonymous referees for helpful comments.

References

1. Angluin, D. A note on the number of queries needed to identify regular languages. *Information and Control 51*, 1 (1981), 76–87.
2. Angluin, D. Queries and concept learning. *Machine Learning 2*, 4 (1987), 319–342.
3. Becerra-Bonache, L., Dediu, A. H., and Tîrnăucă, C. Learning DFA from correction and equivalence queries. In *ICGI* (2006), pp. 281–292.
4. Domaratzki, M., Kisman, D., and Shallit, J. On the number of distinct languages accepted by finite automata with n states. *Journal of Automata, Languages and Combinatorics 7*, 4 (2002).
5. Freund, Y., Kearns, M. J., Ron, D., Rubinfeld, R., Schapire, R. E., and Sellie, L. Efficient learning of typical finite automata from random walks. *Information and Computation 138*, 1 (1997), 23–48.
6. Korshunov, A. The degree of distinguishability of automata. *Diskret. Analiz. 10*, 36 (1967), 39–59.
7. Lee, D., and Yannakakis, M. Testing finite-state machines: State identification and verification. *IEEE Trans. Computers 43*, 3 (1994), 306–320.
8. Trakhtenbrot, B. A., and Barzdin', Y. M. *Finite Automata: Behavior and Synthesis.* North Holland, Amsterdam, 1973.