Machine Learning, 18, 81–108 (1995)

Inferring Finite Automata with Stochastic Output Functions and an Application to Map Learning

THOMAS DEAN tld@cs.brown.edu
Department of Computer Science, Brown University, Providence, RI 02912

DANA ANGLUIN dana_angluin@cs.yale.edu
Department of Computer Science, Yale University, New Haven, CT 06520

KENNETH BASYE kbasye@black.clarku.edu
Department of Computer Science, Brown University, Providence, RI 02912

SEAN ENGELSON engelson-sean@cs.yale.edu
Department of Computer Science, Yale University, New Haven, CT 06520

LESLIE KAELBLING lpk@cs.brown.edu
EVANGELOS KOKKEVIS
ODED MARON oded@ai.mit.edu
Department of Computer Science, Brown University, Providence, RI 02912

Editor: Leonard Pitt

Abstract. It is often useful for a robot to construct a spatial representation of its environment from experiments and observations, in other words, to learn a map of its environment by exploration. In addition, robots, like people, make occasional errors in perceiving the spatial features of their environments. We formulate map learning as the problem of inferring from noisy observations the structure of a reduced deterministic finite automaton. We assume that the automaton to be learned has a distinguishing sequence. Observation noise is modeled by treating the observed output at each state as a random variable, where each visit to the state is an independent trial and the correct output is observed with probability exceeding $1/2$. We assume no errors in the state transition function.

Using this framework, we provide an exploration algorithm to learn the correct structure of such an automaton with probability $1 - \delta$, given as inputs δ, an upper bound m on the number of states, a distinguishing sequence s, and a lower bound $\alpha > 1/2$ on the probability of observing the correct output at any state. The running time and the number of basic actions executed by the learning algorithm are bounded by a polynomial in δ^{-1}, m, $|s|$, and $(1/2 - \alpha)^{-1}$.

We discuss the assumption that a distinguishing sequence is given, and present a method of using a weaker assumption. We also present and discuss simulation results for the algorithm learning several automata derived from office environments.

Keywords: Automata inference, noisy outputs, distinguishing sequences, map learning, spatial representation

1. Introduction

In previous work (Basye & Dean, 1989, Basye et al., 1989), we have argued that robot map learning—inferring the spatial structure of an environment relevant for navigation—can be reduced to inferring the labeled graph induced by the robot's perceptual and locomotive capabilities. Following Kuipers and Byun (Kuipers 1978, Kuipers & Byun, 1988) and Levitt et al. (1987), we assume that the robot has sensory capabilities that enable it to partition space into regions referred to as *locally distinctive places* (LDPs), and that the robot is able to navigate between such regions reliably.

The graph induced by the robot's capabilities has vertices corresponding to LDPs and edges corresponding to navigation procedures. In an office environment, the LDPs might correspond to corridors and the junctions where corridors meet and the navigation procedures to control routines for traversing the corridors separating junctions (Dean et al., 1990).

We are interested in algorithms for learning the induced graph in cases where there is uncertainty in sensing. Uncertainty arises when the information available locally at an LDP is not sufficient to unambiguously identify it (e.g. all L-shaped junctions look pretty much alike to a robot whose perceptual apparatus consists solely of ultrasonic range sensors). Uncertainty also arises as a consequence of errors in sensing (e.g. occasionally a T-shaped junction might be mistaken for an L-shaped junction if one corridor of the junction is temporarily blocked or the robot is misaligned with the walls of the corridors, resulting in spurious readings from specular reflections).

In general, it is not possible for a robot to recover the complete spatial structure of the environment (Dudek et al., 1988) (e.g. the robot's sensors may not allow it to discriminate among distinct structures). As a result, we will be satisfied if the robot learns the *discernable* structure of its environment with *high confidence*. In the following sections, we will define precisely our use of the terms 'discernable' and 'high confidence.'

We are generally interested in the problem faced by a robot in obtaining a model of the dynamics of its interactions with the environment. Our motivations for studying this problem come from practical problems in robotics; however, the contributions of this paper are primarily theoretical. In this and related papers (Basye et al., 1989, Kaelbling et al., 1992), we extend the research on efficiently learning graphs and finite automata to handle more realistic sources of uncertainty. In this paper, we consider the case in which movement is certain and observation is noisy and show how a robot might exploit the determinism in movement to enable efficient learning.

We present polynomial-time algorithms for inferring an unknown deterministic finite automaton with high probability given that the learner (i) can choose the actions that determine state transitions, (ii) can observe the output associated with the state it is in with probability better than chance, and (iii) is given a distinguishing sequence. A distinguishing sequence is a sequence of actions such that for any starting state the sequence of outputs associated with the states encountered in executing that sequence uniquely identifies the starting state. To determine what state it is in the robot repeatedly executes the distinguishing sequence gathering statistics and watching for patterns in the data. Given that the robot can uniquely identify states, learning the automaton is relatively straightforward. The rest of this paper describes the details.

2. Preliminaries

To formalize the problem, we represent the interaction of the robot with its environment as a deterministic finite automaton (DFA). In the DFA representation, the states correspond to LDPs, the inputs to robot actions (navigation procedures), and the outputs to the information available at a given LDP. A DFA is a six tuple, $M = (Q, B, Y, \zeta, q_0, \gamma)$, where

- Q is a finite nonempty set of states,
- B is a finite nonempty set of inputs or basic actions,

- Y is a finite nonempty set of outputs or percepts,
- ζ is the transition function, $\zeta: Q \times B \rightarrow Q$,
- q_0 is the initial state, and
- γ is the output function, $\gamma: Q \rightarrow Y$.

Let $A = B^*$ denote the set of all finite sequences of actions, and $|a|$ denote the length of the sequence $a \in A$. Let $q\langle a \rangle$ be the sequence of outputs of length $|a| + 1$ resulting from executing the sequence a starting in q, and qa be the final state following the execution of the sequence a starting in q. An automaton is said to be *reduced* if, for all $q_1 \neq q_2 \in Q$, there exists $a \in A$ such that $q_1\langle a \rangle \neq q_2\langle a \rangle$. A reduced automaton is used to represent the *discernable* structure of the environment; you cannot expect a robot to discern the difference between two states if no sequence of actions and observations serves to distinguish them. A *homing sequence*, $h \in A$, has the property that, for all $q_1, q_2 \in Q$, $q_1\langle h \rangle = q_2\langle h \rangle$ implies $q_1 h = q_2 h$. Every automaton has a homing sequence; however, the shortest homing sequence may be as long as $|Q|^2$ (Rivest & Schapire 1989).

There are a variety of sources of supplementary knowledge that can, in some cases, simplify inference. For instance, it may help to know the number of states, $|Q|$, or the number of outputs, $|Y|$. It often helps to have some way of distinguishing where the robot is or where it was. A *reset* allows the robot to return to the initial state at any time. The availability of a reset provides a powerful advantage by allowing the robot to anchor all of its observations with respect to a uniquely distinguishable state, q_0. A homing sequence, h, allows the robot to distinguish the states that it ends up in immediately following the execution of h; the sequence of observations $q\langle h \rangle$ constitutes a unique signature for state qh. Rivest and Schapire (1989) show how to make use of a homing sequence as a substitute for a reset. A sequence, $d \in A$, is said to be a *distinguishing sequence* if, for all $q_1, q_2 \in Q$, $q_1\langle d \rangle = q_2\langle d \rangle$ implies $q_1 = q_2$. (Every distinguishing sequence is a homing sequence, but not the other way around.) A distinguishing sequence, d, allows the robot to distinguish the states that it starts executing d in; the sequence of observations $q\langle d \rangle$ constitutes a unique signature for q. Not all automata have distinguishing sequences.

3. Uncertainty in observation

In this paper, we are interested in the case in which the observations made at an LDP are corrupted by some stochastic noise process. In the remainder of this paper, we distinguish between the output function, γ, and the *observation* function, φ. We say that the output function is *unambigous* if $\forall q_1, q_2 \in Q, \gamma(q_1) = \gamma(q_2)$ implies $q_1 = q_2$; otherwise, it is said to be *ambiguous*. If the output function is unambiguous and $\varphi = \gamma$, then there is no uncertainty in observation and learning is easy.

The case in which the output function is ambiguous and $\varphi = \gamma$ has been studied extensively. The problem of inferring the smallest DFA consistent with a set of input/output pairs is NP-complete (Angluin 1978, Gold 1978).[1] Even finding a DFA polynomially close to the smallest is intractable assuming P $\neq$ NP (Pitt & Warmuth, 1989). Kearns and

Valiant (1989) show that predicting the outputs of an unknown DFA on inputs chosen from an arbitrary probability distribution is as hard as computing certain apparently hard number-theoretic predicates. Angluin (1987), building on the work of Gold (1972), provides a polynomial-time algorithm for inferring the smallest DFA given the ability to reset the automaton to the initial state at any time and a source of counterexamples. In Angluin's model, at any point, the robot can hypothesize a DFA and the source of counterexamples will indicate if it is correct and, if it is not, provide a sequence of inputs on which the hypothesized and actual DFAs generate different outputs. Rivest and Schapire show how to dispense with the reset in the general case (Rivest & Schapire 1989), and how to dispense with both the reset and the source of counterexamples in the case in which a distinguishing sequence is either provided or can be learned in polynomial time (Rivest & Schapire, 1987, Schapire, 1991).

This last result is particularly important for the task of learning maps. For many man-made and natural environments it is straightforward to determine a distinguishing sequence. In most office environments, a short, randomly chosen sequence of turns will serve to distinguish all junctions in the environment. Large, complicated mazes do not have this property, but we are not practically interested in learning such environments.

The case in which $\varphi \neq \gamma$ is the subject of this paper. In particular, we are interested in the case in which there is a probability distribution governing what the robot observes in a state. There are several alternatives for the sample space of the distribution governing the robot's observations.

- Each location is a single independent trial. Errors are persistent: visiting multiple times doesn't help.
- Each visit to a location is an independent trial. Visiting multiple times helps.
- Each location is associated with a stochastic process that depends on the time of visitation. Visiting at widely spaced times often helps.
- Each location/direction pair is treated distinctly. The direction from which you enter a location matters.

In this paper, we concentrate on the case in which each visit to a location is an independent trial. To avoid pathological situations, we assume that the robot observes the actual output with probability better than chance; that is,

$$\forall q \in Q, \Pr(\varphi(q) = \gamma(q)) \geq \alpha > 0.5,$$

where, in this case, $\varphi(q)$ is a random variable ranging over Y. This model is a special case of the hidden Markov model (Levinson et al. 1983) in which state transitions are deterministic and the stochastic processes associated with the observations of states are restricted by the above requirement. The closer α is to $1/2$, the less reliable the robot's observations. In the following section, we provide algorithms that allow the robot to learn the structure of its environment with probability $1 - \delta$ for a given $0 < \delta < 1$. We are interested in algorithms that learn the environment in a total number of steps that is polynomial in $1/\delta$, $1/(\alpha - 1/2)$, $|Q|$, $|B|$, and $|Y|$. We assume that the robot is given α.

We do not consider the case in which the robot remains in the same state by repeating the empty sequence of actions and observes the output sufficiently often to get a good idea of the correct output. Our rationale for not considering this case is that the independence assumption regarding different observations of the same state is not even approximately satisfied if the robot does not take any overt action between observations, whereas if there are overt actions between observations, the observations are more likely to be independent.

Note that it is often possible to determine whether your observations are seriously corrupted (e.g. you notice fog, rain or some other obscuring process). This ability effectively improves the accuracy of your observations.

4. Learning algorithms

In the following, we present a high-probability, polynomial-time procedure, LOCALIZE, for *localizing the robot* (directing the robot to a state that it can distinguish from all other states), and then show how this procedure can be used to learn environments in which the robot is given a distinguishing sequence.[2] Finally, we discuss how LOCALIZE might be used to learn a distinguishing sequence in certain cases in which a distinguishing sequence is guaranteed to exist. We do not assume a reset. We do, however, assume that the transition graph is strongly connected (that is, every state is reachable by some sequence of actions from every other state), thereby avoiding the possibility that the robot can become trapped in some strongly connected component from which it can never escape.

4.1. The localization procedure

The procedure LOCALIZE works by exploiting the fact that movement is deterministic. The basic idea is to execute repeatedly a fixed sequence of actions until the robot is certain to be "going in circles" repeating a fixed sequence of locations visited, corresponding to a cyclic walk in the underlying deterministic automaton. If we knew the period of repetition of the sequence of locations, we could keep separate statistics on the outputs observed at each location. These statistics could then be used to deduce (with high probability) the correct outputs at those locations, and hence to localize the robot by supplying a signature for the state the robot is in.

The problem then is to figure out the period of repetition of the walk with high probability. We keep statistics for the alternative hypotheses for the period of the cycle, which are then analyzed to determine with high probability the true period of the cycle.

This analysis is complicated by the fact that distinct states may be mistakenly conflated if only local information is used. This can be seen in the simple example of a machine with six states, one input symbol, a, and three output symbols, b, c, and d. Inputs cause transitions from state i to state $i + 1 \bmod 6$. Output probabilities are as follows:

State	b	c	d
0	.51	.49	0
1	0	0	1
2	0	1	0
3	.51	0	.49
4	0	1	0
5	0	0	1

We can now gather output statistics for this machine while repeating the distinguishing sequence, and test different hypotheses as to the walk's period. For the hypothesis of period two, we expect to see the output with frequencies:

States	b	c	d
{0,2,4}	.17	.83	0
{1,3,5}	.17	0	.83

This table looks quite plausible, so we could be led to the (erroneous) conclusion that the automaton has two states, with outputs c and d. This sort of 'state conflation' can be avoided by combining information gotten by considering different period hypotheses, as we shall see below.

Recall that the Markov assumption implies a stationary probability distribution over the outputs at each state. If the states are numbered from 1 to n and the outputs are σ_j for $j = 1, 2, \ldots, k$, let $\alpha_{i,j}$ denote the probability of observing symbol σ_j given that the robot is in state i.

We assume an upper bound m on the number of states of the automaton. Let $s = b_1 b_2 \cdots b_{|s|}$ be a sequence of one or more basic actions; we assume that s is a distinguishing sequence for the underlying automaton. Let q_i be the state reached after executing the sequence of actions s^{m+i}, that is, m repetitions of s followed by i repetitions of s. The first m repetitions ensure that the robot is in the cyclic walk. The sequence of states $q_0, q_1, q_2, \ldots$ is periodic; let p denote the least period of the cycle. Note that $p \leq m$. Our main goal is to determine (with high probability) the value of p.

For each $\ell = 0, \ldots, |s| - 1$, we also consider the sequence of states q_i^ℓ reached after executing the sequence of actions $s^{m+i} b_1 b_2 \cdots b_\ell$. That is, q_i^ℓ is the state reached from q_i by executing the first ℓ basic actions from the sequence s. For each ℓ, the sequence $q_0^\ell, q_1^\ell, q_2^\ell, \ldots$ is also periodic of period p.

For each ℓ, consider the sequence of (correct) outputs from the states q_i^ℓ:

$$\gamma_i^\ell = \gamma(q_i^\ell).$$

The output sequence

$$\gamma_0^\ell, \gamma_1^\ell, \gamma_2^\ell, \ldots.$$

Table 1. Sequences of visited states.

Step #	States visited						
0	0	10	5	0	10	5	...
1	1	11	6	1	11	6	...
2	2	12	7	2	12	7	...
3	3	13	8	3	13	8	...
4	4	14	9	4	14	9	...
5	5	0	10	5	0	10	...
6	6	1	11	6	1	11	...
7	7	2	12	7	2	12	...
8	8	3	13	8	3	13	...
9	9	4	14	9	4	14	...

is also periodic, of some least period p_ℓ dividing p. Since we are not assuming the outputs are unambiguous, p_ℓ may be smaller than p. We can show, however, that p will be the least common multiple (LCM) of all the p_ℓ's.

It is clear that p is a multiple of each p_ℓ, so suppose instead that the LCM p'' of all the p_ℓ's is a proper divisor of p. This implies that as we traverse the cycle of p distinct states,

$$q_0, q_1, \ldots q_p,$$

the output sequences $q_i\langle s\rangle$ must repeat with period $p'' < p$, which implies that two distinct states have the same sequence of outputs, contradicting the assumption that s is a distinguishing sequence.

Thus, it would suffice if we could determine each of the values p_ℓ, and take their LCM. In fact, what we will be able to do is to determine (with high probability) values r_ℓ such that p_ℓ divides r_ℓ and r_ℓ divides p – this also will be sufficient. To avoid the superscripts, we describe the procedure for the sequence $q_0, q_1, q_2, \ldots$; it is analogous for the others.

Consider any candidate period $\pi \leq m$, and let g be the greatest common divisor of p and π (GCD(p, π)). For each $0 \leq i \leq \pi - 1$, consider the sequence of states visited every π repetitions of s, starting with $m + i$ repetitions of s. This will be the sequence of states

$$q_i, q_{i+\pi}, q_{i+2\pi}, q_{i+3\pi}, \ldots.$$

Since q_i is periodic of period p, this sequence visits each state of the set $\{q_{i+kg}\colon k = 0, 1, \ldots, p/g - 1\}$ in some order, and then continues to repeat this cycle of p/g states.

Here's an example for $p = 15$, $\pi = 10$, $g = 5$. In this case, row r of Table 1 gives the indices of the states visited every 10 repetitions of s, starting from $m + r$ repetitions of s, assuming that the basic period of the states under repetitions of s is $q_0, q_1, \ldots, q_{14}$.

Note that in Table 1, the set of states visited for row r is the same as the set of states visited for row $r + 5$, for $0 \leq r \leq 4$. This holds in general: the set of states visited every π repetitions of s starting from $m + r$ repetitions of π is the same as the set of states visited every π repetitions of s starting from $m + r + g$ repetitions of π. Thus, the sequence of sets $S_0, S_1, \ldots, S_{\pi-1}$ is periodic with period g.

In the special case $\pi = p$, row r will consist exclusively of visits to state q_r. In the special case $g = 1$, that is, π and p relatively prime, each row will consist of repetitions (in some fixed order) of a visit to each of the p states.

4.1.1. What we observe. Of course, what we can observe is just the stochastically determined output at each state the robot visits. The overall operation of the algorithm will be to repeat the action sequence s a total of $m + m^2t$ times, where t will be chosen to ensure that we know the true output frequencies accurately, with high probability. For each candidate period $\pi \leq m$ we form a table with π rows numbered 0 to $\pi - 1$, and k columns, one for each possible output symbol σ_j. In row r and column j we record the relative frequency of observations of symbol σ_j after executing $s^{m+r+v\pi}$ for $v = 0, 1, \ldots, mt - 1$. Since $\pi \leq m$ and $r \leq \pi - 1$, these observations are all available in a run containing $m + m^2t$ repetitions of s.

If in this table each row has a "majority output" whose observed frequency is at least $\frac{1}{2} + \frac{1}{2}\mathbf{sep}$, where

$$\mathbf{sep} = \left(\alpha - \frac{1}{2}\right),$$

the table is said to be *plausible*.[3] Otherwise, the table is ignored. If the table is plausible, we then take the sequence of π majority outputs determined by the rows and find the minimum period π' (dividing π) such that the sequence of majority outputs has period π'. As our candidate for the period of the sequence of outputs we take the LCM of all the numbers π' obtained this way from plausible tables.

4.1.2. Justification. Why does this work? When $\pi = p$, the rows of the table correspond exactly to the distinct states q_r, so with high probability in each row we will get a frequency of at least $\frac{1}{2} + \frac{1}{2}\mathbf{sep}$ for the correct output (provided t is large enough) from each state, and therefore actually have the sequence of correct outputs for the cycle of p states, whose period is (by hypothesis) p'. Thus, with high probability the table corresponding to $\pi = p$ will be plausible, and one of the values π' will be p' itself.

When $\pi \neq p$, as we saw above, the set of states S_r visited in row r is determined by $g = \text{GCD}(\pi, p)$ and $r \bmod g$. In the limit as t becomes large each state in S_r is visited equally often and the expected frequency of σ_j for row r is just the average of $\alpha_{i,j}$ over $i \in S_r$. Since the sets S_r are periodic over r with period g, the expected frequencies for a given symbol in rows $0, 1, 2, \ldots, \pi - 1$ is periodic, with least period dividing g. Thus, provided t is large enough, if the table for π is plausible, the value π' will be a divisor of g (with high probability), and so a divisor of p.

Thus, with high probability, the value we determine will be the LCM of p' and a set of values π' dividing p, and therefore will be a multiple of p' and a divisor of p, as claimed. Recall that we must repeat this operation for the sequences q_i^ℓ determined by proper prefixes of the action sequence s, and take the LCM of all the resulting numbers. However, the basic set of observations from $m + m^2t$ repetitions of s can be used for all these computations.

4.1.3. Required number of trials. In this section, we use straightforward tail-bound arguments to determine a sufficient bound on t to guarantee that with probability $1 - \delta$, the observed output frequencies converge to within $\frac{1}{2}\mathbf{sep} = \frac{1}{2}(\alpha - \frac{1}{2})$ of their true values. Recall that p denotes the least period of the sequence $q_0, q_1, q_2, \ldots$.

First consider the frequency table for π. Let $g = \text{GCD}(p, \pi)$ and $p = gh$. Let $S_r = \{q_{i_1}, q_{i_2}, \ldots, q_{i_h}\}$ be the set of states visited in row r of the table. Let v_u denote the number of visits to state q_{i_u} used in calculating row r. The expected frequency of observations of output σ_j in row r is

$$f_{r,j} = \sum_{u=1}^{h} \frac{v_u}{mt} \alpha_{i_u,j}.$$

The total number of visits to states in row r is mt, and the states are visited in a fixed cyclic order. Since $h \leq m$, each state in S_r is visited at least t times in row r. More precisely, for each u, v_u is either $\lfloor mt/h \rfloor$ or $\lceil mt/h \rceil$.

To ensure that n events occur with probability $1 - \delta$, it is enough to ensure that each of them occurs with probability $1 - \frac{\delta}{n}$. If we choose t sufficiently large that for each state i in S_r the observed frequency of output σ_j is within $\frac{1}{2}\mathbf{sep}$ of $\alpha_{i,j}$ with probability at least $1 - \delta/|s|km^3$, then with probability at least $1 - \delta/|s|km^2$, the observed frequency of symbol σ_j in row r will be within $\frac{1}{2}\mathbf{sep}$ of its expected value $f_{r,j}$, and with probability at least $1 - \delta/|s|m$, each of the at most km entries in the table for π will be within $\frac{1}{2}\mathbf{sep}$ of its expected value. In this case, the probability will be at least $1 - \delta/|s|$ that all the values in all the (at most m) tables will be within $\frac{1}{2}\mathbf{sep}$ of their expected values. We repeat this operation for each of the $|s|$ proper prefixes of s, so, in this case the probability is at least $1 - \delta$ that all the observed frequencies in all the tables considered for each prefix of s will be within $\frac{1}{2}\mathbf{sep}$ of their expected values.

We consider what happens in the case that all the entries in all the tables are within $\frac{1}{2}\mathbf{sep}$ of their expected values. In the case $\pi = p$, the expected value of the correct output for state q_r and hence row r of the table is at least α, which means that the observed frequencies for the correct outputs will all exceed $\frac{1}{2} + \frac{1}{2}\mathbf{sep}$, and the table will be plausible and the "majority outputs" will be the correct outputs, of period p' by hypothesis.

In the case of $\pi \neq p$, no output σ_j whose expected frequency $f_{r,j}$ in a row is at most $\frac{1}{2}$ can have an observed frequency exceeding $\frac{1}{2} + \frac{1}{2}\mathbf{sep}$. Thus, each output with an observed frequency exceeding $\frac{1}{2} + \frac{1}{2}\mathbf{sep}$ has a true frequency of greater than $\frac{1}{2}$, and so is uniquely determined for that row. This guarantees that if the table for π is plausible, then the "majority output" from row r is uniquely determined by the set S_r, and the sequence of "majority outputs" will be periodic of some period dividing $g = \text{GCD}(\pi, p)$, as claimed. Thus, assuming all the values in all the tables are within $\frac{1}{2}\mathbf{sep}$ of their expected values, the LCM of the values of π' will correctly determine a value q that is a multiple of p' and a divisor of p. Since this is true for each prefix of s, the correct value of p is determined.

According to Hoeffding's inequality, given a variable X that is an average of t identical Bernoulli random variables with mean μ, and any error bound $0 < \tau < 1$, we have:

$$\Pr(|X - \mu| \geq \tau) \leq 2e^{-2t\tau^2}.$$

Since the number of observations of symbol σ_j in state i in t visits to state i is such a variable, with mean $\alpha_{i,j}$, and we are interested in

$$\tau = \frac{1}{2}\mathbf{sep} = \frac{(2\alpha - 1)}{4},$$

and in achieving probability of failure at most $\frac{\delta}{|s|km^3}$, it suffices if

$$t \geq \frac{8}{(2\alpha - 1)^2} \ln \frac{2|s|km^3}{\delta}.$$

Recall that the total number of repetitions of s is $m + m^2 t$, so s must be executed at least

$$\mathcal{N} = m + \frac{8m^2}{(2\alpha - 1)^2} \ln \frac{2|s|km^3}{\delta}$$

times. The number of basic actions executed is thus

$$O\left(\frac{|s|m^2}{(\alpha - \frac{1}{2})^2} \ln \frac{|s|km^3}{\delta}\right).$$

Recall that $|s|$ is the length of the distinguishing sequence, m is an upper bound on the number of states in the automaton, $\alpha > \frac{1}{2}$ is a known lower bound on the probability of the correct output from a state, k is a bound on the number of possible outputs, and δ is a bound on the probability that the procedure will fail. Note that the dependence on $|s|$ is polynomial, the dependence on k is logarithmic.

4.1.4. The LOCALIZE *algorithm.* Based on the discussion above, we now present LOCALIZE.

1. For simplicity, we assume that all the possible outputs are known and correspond to the integers $1, \ldots, k$. Build a table $T(\pi, \ell, r, j)$ of size $m \times |s| \times m \times k$. Initialize all the table entries to zero.[4]

2. Execute the sequence s m times to ensure that the robot is in a closed walk that it will continually traverse for as long as it continues to execute s.[5]

3. Initialize the sequence counter: $R \leftarrow 0$ and the step counter: $c \leftarrow 0$.

4. Execute s at least $\mathcal{N}$ times, incrementing R each time. After executing each individual step, do:

   (a) Increment the step counter: $c \leftarrow c + 1$. Let $\ell = c \bmod |s|$, and j be the output observed immediately following execution.

   (b) For each $\pi = 1, 2, \ldots, m - 1$, the table entry $T(\pi, \ell, R \bmod \pi, j)$ is incremented by 1.

5. Let

$$F(\pi, \ell, r, j) = \frac{T(\pi, \ell, r, j)}{\sum_{j=0}^{k} T(\pi, \ell, r, j)}.$$

6. Let P be the LCM of all π' such that there exist π and ℓ such that for all $r < \pi$ there exists j such that $F(\pi, \ell, r, j) > \frac{1}{2} + \frac{1}{2}\mathbf{sep}$ and π' is the period of the outputs $\arg\max_j F(\pi, \ell, r, j)$ for $r = 0, 1, \ldots, \pi - 1$.

7. Conclude that the robot is currently located at the state corresponding to row $r = R \bmod P$ in the main table for P, and return, as the hypothesis for the correct outputs of the distinguishing sequence s from this state, the sequence of outputs $\arg\max_j F(P, \ell, r, j)$ for $\ell = 0, 1, \ldots, |s| - 1$ concatenated with the single output $\arg\max_j F(P, 0, (r+1) \bmod P, j)$.

4.2. *The map learning procedure*

Now we can define a procedure, BUILDMAP, for learning maps given a distinguishing sequence, s. Suppose for a moment that LOCALIZE always returns the robot to the same state and that the robot can always determine when it is in a state that it has visited before. In this case, the robot can learn the connectivity of the underlying automaton by performing what amounts to a depth-first search through the automaton's state transition graph. The robot does not actually traverse the state-transition graph in depth-first fashion; it cannot manage a depth-first search since, in general, it cannot backtrack. Instead, it executes sequences of actions corresponding to paths through the state transition graph starting from the root of the depth-first search tree by returning to the root each time using LOCALIZE. When, in the course of the search, a state is recognized as having been visited before, an appropriate arc is added to the inferred automaton and the search 'backtracks' to the next path that has not been completely explored.

The algorithm we present below is more complicated because our localization procedure does not necessarily always put the robot in the same final state and because we are not able to immediately identify the states we encounter during the depth-first search. The first problem is solved by performing many searches in parallel,[6] one for each possible (root) state that LOCALIZE ends up in. Whenever the LOCALIZE is executed the robot knows (with high probability) what state it has landed in, and can take a step of the depth-first search that has that state as the root node. The second problem is solved by using a number of executions of the distinguishing sequence from a given starting state to identify that state with high probability.

The algorithm, informally, proceeds in the following way. The robot runs LOCALIZE, ending in some state q. Associated with that state is a depth-first search which is in the process of trying to identify the node at the end of a particular path. The actions for that path are executed, then the distinguishing sequence is executed. The results of the distinguishing sequence are tabulated, then the robot begins this cycle again with the localization procedure. Eventually, the current node in some search will have been explored enough times for a

high-probability determination to be made about its unique signature. Once that is done, if a node with the same signature has been identified previously in this search, the two nodes are identified in the hypothesized state transition graph and the search backtracks. If the signature is new, then a new node with the corresponding signature is added to the hypothesized graph and the search proceeds to the next level of depth. We now explain the algorithm more formally.

4.2.1. The BUILDMAP *algorithm.* With each state, q, that the robot ends up in using LOCALIZE, we associate a set, V_q, of pairs $\langle x, y \rangle$, where x is a sequence of basic actions representing a path through the state transition graph and y is a high-probability estimate of the sequence of outputs obtained by executing the distinguishing sequence, s, after executing the actions in x starting from q. That is, with high probability $qx\langle s \rangle = y$. In addition, each q has a current path, x_q and signature estimation table, T_q. The signature estimation table is indexed in one dimension by $|s|$, the length of the distinguishing sequence, and in the other dimension by k, the number of possible outputs. The set of states in the hypothesized automaton is acquired over time from the output of LOCALIZE and the high-probability signatures obtained in identifying states. We will refer to states and their signatures interchangeably. Each time a new signature is encountered, the robot initializes the associated data structures as follows.

1. $V_q \leftarrow \{\langle \lambda, q \rangle\}$
2. $x_q \leftarrow \text{First}(B)$
3. For each $0 \leq i \leq |s|$ and $1 \leq j \leq k$, $T_q[i, j] \leftarrow 0$.

Here we assume that the set of basic actions, B, is ordered and that First(B) denotes the first action in B. The null sequence is denoted λ. Using the above initialization subroutine, the map-learning procedure BUILDMAP is described as follows.

1. Execute LOCALIZE, ending in some state q.
2. If q is new, then initialize it.
3. Execute x_q.
4. For current output σ_j, increment $T_q[0, j]$.
5. For i from 1 to $|s|$,
   (a) Execute next basic action in s, getting output σ_j.
   (b) Increment $T_q[i, j]$.
6. If sum of the entries in a row of T_q equals $\mathcal{M}$ (in this case the distinguishing sequence has been executed $\mathcal{M}$ times from the end of the current path), then
   (a) Let $y = s_1 \ldots s_{|s|}$, where $s_i = \arg\max_j T_q[i, j]$.

(b) Add $\langle x_q, y \rangle$ to V_q.

(c) If there is no other entry in V_q with signature equal to y, then set x_q to the concatenation of x_q and First(B), else backtrack by setting x_q to the next unexplored path in a depth-first search of V_q or return V_q if no such path exists.

(d) For all $0 \leq i \leq |s|$ and $1 \leq j \leq k$, $T_q[i,j] \leftarrow 0$.

7. Go to Step 1.

4.2.2. Required number of steps. The above procedure will return the results of the first depth-first search that is finished. The returned V_q contains all of the information required to construct the state transition graph for the automaton with high probability assuming that LOCALIZE succeeds with high enough probability and that the estimations used in identifying states are correct with high enough probability. In the following, we show how to guarantee that these assumptions are satisfied.

The result of the depth-first search is correct if all of its arcs have been correctly identified. This requires at most $m|B|$ correct node identifications; one for executing each basic action from each state.

A node is correctly identified if its signature is correctly determined, which will be the case if each of its $|s|$ components is correct. A component of the final signature is correct whenever the maximal frequency in the associated row of T is the one for the correct output. An individual observation of a signature element is correct if LOCALIZE succeeded (we started where we think we started) and the state is perceived correctly; this will happen with probability $\alpha\beta$, where $\beta = 1 - \delta$ is the probability that LOCALIZE succeeds. In the worst case, the next most likely element of the table will have expected frequency $1 - \alpha\beta$, giving a separation between these two probabilities of $2\alpha\beta - 1$. The row will yield the correct answer if all of its entries are within $\alpha\beta - 1/2$ of their expected values. Using Hoeffding's inequality as before, after $\mathcal{M}$ trials, the probability that an individual table element is in error is bounded above by

$$2e^{-2\mathcal{M}(\alpha\beta - 1/2)^2}.$$

In order to guarantee that the total error probability of the algorithm is less than η, we must ensure that the error probability for a single table entry is less than $\eta/(|s|km|B|)$. This will be the case if

$$\mathcal{M} \geq \frac{1}{2(\alpha\beta - 1/2)^2} \ln \frac{2|s|km|B|}{\eta}.$$

At worst, we must perform $|Q|$ depth-first searches,[7] each of which requires $|Q||B|$ nodes to be identified.[8] To identify a node, we must execute the distinguishing sequence there $\mathcal{M}$ times. To do that requires $\mathcal{N} + 1$ executions of the distinguishing sequence; $\mathcal{N}$ of them to localize to the root of the search, plus 1 to identify the node. In addition, each attempt at node identification requires the path to be followed from the root of the tree to the node; the length of the path is bounded above by $|Q|$. Thus, the number of steps required is at most

$$|Q|^2|B|\mathcal{M}((\mathcal{N}+1)|s| + |Q|),$$

which is

$$O(|Q|^3|B|\mathcal{M} + |Q|^2|B|\mathcal{M}\mathcal{N}|s|) .$$

Substituting in our bounds for $\mathcal{M}$ and $\mathcal{N}$, in order to assure an error probability less than η, we must take a number of steps on the order of

$$\frac{|Q|^3|B|}{(\alpha\beta - 1/2)^2} \ln \frac{|s|km|B|}{\eta} + \frac{|s|^2|Q|^2m^2|B|}{(\alpha\beta - 1/2)^2(2\alpha - 1)^2} \ln \frac{|s|km|B|}{\eta} \ln \frac{|s|km^3}{1-\beta} .$$

The parameter β must be chosen so that $1/(2\alpha) < \beta < 1$; under that constraint, it may be chosen so as to minimize the number of steps while keeping total error less than η. For purposes of analysis, we might choose β to be in the middle of its allowed range, or $1/(4\alpha) + 1/2$. Making this substitution, the $\alpha\beta - 1$ terms become $(\alpha - \frac{1}{2})/2$, and the $1/(1-\beta)$ term becomes $4\alpha/(2\alpha - 1)$, which is bounded above by $2/(\alpha - \frac{1}{2})$. As a result, the entire algorithm is polynomial in $1/(\alpha - \frac{1}{2})$, $|Q|$, $|B|$, k, m, and $|s|$.

4.3. Learning distinguishing sequences

The above procedures rely on the robot being given a distinguishing sequence. If we are not given a distinguishing sequence but know that such sequences exist, it would be useful to be able to learn one. In this section, we consider the problem of learning distinguishing sequences.

4.3.1. Do we need a distinguishing sequence? The homing sequence algorithm of Rivest and Schapire (1989) learns a DFA environment in polynomial time by means of exploration and counterexamples. A *counterexample* is an action sequence that leads to an observation different from the prediction of the current map. The counterexamples are necessary to this result; there is no polynomial-time algorithm to learn arbitrary strongly-connected DFA environments using exploration only, even if the goal is identification with high probability.

By contrast, exploration-only suffices for the polynomial-time learning algorithm we present above. Of course, there are two provisos: (1) the target DFA environment has a distinguishing sequence and (2) the algorithm is given a distinguishing sequence for the DFA. The question arises whether we could dispense with assumption (2): assuming only that the DFA environment has a distinguishing sequence, could there be a polynomial-time algorithm to discover one incrementally using exploration only?

For one subclass of DFAs with distinguishing sequences, the answer is yes, namely, the permutation automata, as shown by the results of Rivest and Schapire. However, for the full class of DFAs with distinguishing sequences, the answer is no. That is, the class of DFA environments with distinguishing sequences cannot be learned in polynomial time using exploration only, even if the goal is identification with high probability.

In the remainder of this section, we sketch an adversary argument to prove this claim. The class of automata used in the proof was arrived at by modifying the standard class of

'password automata' to have distinguishing sequences without compromising their 'cryptographic' attributes. This shows that our algorithm will not be able to do entirely without a distinguishing sequence. However, in the next section we show how to use a random generator of sequences that produces correct distinguishing sequences only some small (at least inverse polynomial) fraction of the time.

Lemma 1. *Any algorithm that can learn the class of DFA environments with distinguishing sequences by exploration only will take exponential time on some environments, even if the goal is identification with high probability and a reset operation is available.*

Proof. Let $n \geq 1$. We describe a particular class C_n of DFA environments with the property that each DFA in C_n has $O(n)$ states and a distinguishing sequence of length $O(n^2)$, and any algorithm that identifies every element of C_n by exploration only must take a nonpolynomial (in n) number of basic actions when the target is some DFA in C_n.

Each environment has $4n+1$ states, numbered 0 to $4n$, $2n+1$ basic actions of the form, $b, a_0, a_1, \ldots, a_{2n-1}$, and three output symbols, $*, 0, 1$. State 0 is the start state and state $4n$ is the dead state. All states have output symbol $*$ except state $2n$, which has output symbol either 0 or 1. Since state $2n$ is the only state with a distinguished output, we call it the *signal state*.

Choose any subset of cardinality n from the set $0, 1, \ldots, 2n-1$, say

$$i_0 < i_1 < \ldots < i_{n-1}.$$

Note that there are $\binom{2n}{n}$ ways to form such a subset. Let

$$G = \{a_{i_0}, a_{i_1}, \ldots, a_{i_{n-1}}\}.$$

The set G determines the transitions of the DFA as follows. Let $0 \leq j \leq 4n-1$. If j is even, let $k = j/2 \bmod n$ and define

1. $\delta(j, a_{i_k}) = (j+1) \bmod 4n$,
2. $\delta(j, b) = (j-1) \bmod 4n$,
3. $\delta(j, a_{i_r}) = j$ for $a_{i_r} \in G - \{a_{i_k}\}$.

If j is odd, let $k = (j-1)/2 \bmod n$ and define

1. $\delta(j, a_{i_k}) = (j-1) \bmod 4n$,
2. $\delta(j, b) = (j+1) \bmod 4n$,
3. $\delta(j, a_{i_r}) = j$ for $a_{i_r} \in G - \{a_{i_k}\}$.

All other transitions are to the dead state, $4n$; in particular, all transitions on the n symbols a_i not from the set G are to the dead state. Call the symbols in G *good choices* and the other a_i's *bad choices*. Figure 1 depicts a generic example of a C_2 DFA.

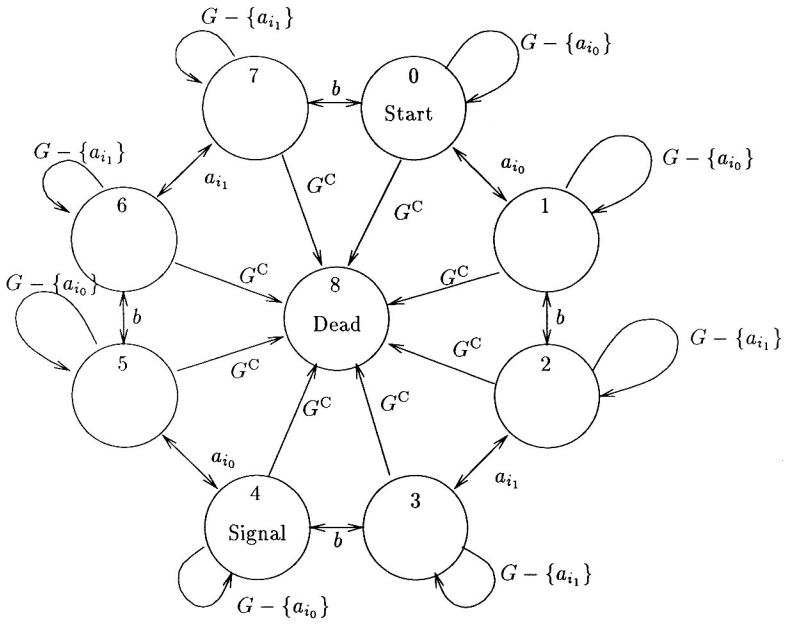

Figure 1. Generic example of C_2 DFA. G^C denotes the set of bad choices.

Intuitively, the $4n$ non-dead states form a ring that can be traversed from the start state 0 in increasing order back to the start state by the string

$$(a_{i_0} b a_{i_1} b \cdots a_{i_{n-1}} b)^2,$$

and in the reverse direction by the string

$$(b a_{i_{n-1}} b a_{i_{n-2}} \cdots b a_{i_0})^2.$$

The left half of either of these strings will proceed from the start state to the signal state, $2n$.

Note that symbols a_{i_k} that are good choices either move one state around the ring or stay in the same state. Hence the string

$$d = (a_{i_0} a_{i_1} \cdots a_{i_{n-1}} b)^{2n}$$

starting from an even numbered state j in the ring moves around the ring in increasing order until it comes back to j. If we execute d starting from an odd numbered state j, we move around the ring in decreasing order until we come back to j. Since in either case we are bound to encounter the signal state in a unique location in the string, the string d is a distinguishing sequence for this DFA. (Note that if we execute d from the dead state, we do not encounter the signal state at all.) The length of d is $O(n^2)$.

Thus, every DFA environment in the class C_n has a distinguishing string, determined by the corresponding set G. Intuitively, what prevents us from using this information to

learn elements of the class C_n? If we are given an unknown element of C_n, we do not know G, that is, which subset of n of the $2n$ symbols a_i are "good choices." If we try to guess this information, we are liable to include a "bad choice" that will leave us in the dead state capable of gathering no more useful information until we issue a reset. Moreover, to identify the environment, we must visit the signal state, which means we have to execute all and only the "good choices" appropriately interspersed with b's.

These intuitions may be turned into an adversary argument as follows. Suppose A is an algorithm that identifies with high probability every environment in C_n by exploration only, possibly using resets. The adversary will simulate A, returning certain observations in response to the successive basic actions executed by A. The adversary gets to pick the DFA subject to its being consistent with the observations it has already returned. The adversary keeps track of the set T of possible environments, as well as the current state of the learning algorithm in each one. Initially T contains all the elements of C_n, each one in its initial state.

In response to each basic action executed by A, the adversary updates the current state of A for each element of T. As long as there are more than two elements of T, the adversary returns the observation * and removes from T any DFAs that don't have output * at the current state. If there are only two elements of T, then if they both have the same output at their current states, the adversary returns that observation to A. Otherwise, the two DFAs have different outputs, and the adversary selects one of them and answers consistent with that one for all remaining actions.

Note that for every choice of G, there are two corresponding DFAs in C_n that have the same transition function and only differ in the output (0 or 1) from the signal state; any sequence of actions from A will reach the same state in both machines. How many DFAs can be eliminated from T by an observation of *? We claim the answer is at most two. To be eliminated, a DFA must be in its signal state. But, for a DFA corresponding to G to be in its signal state, the sequence of actions since the last reset (or the beginning of the sequence, if there have been no resets) must contain all of the actions from G and no action a_i not in G. Since this condition cannot be satisfied simultaneously for two distinct sets G, at most the two DFAs corresponding to a single G can be eliminated from T by an observation of *.

If the algorithm A gives its output before T is reduced to one element, then it must make an error when the target environment is at least one of the elements of T. This shows that any deterministic algorithm A that successfully identifies every element of C_n must, when the target is some element of C_n, execute enough basic actions to eliminate all but one element of C_n; this requires at least 2^n basic actions, since $\binom{2n}{n}$ is bounded below by 2^n.

If the algorithm A is randomized we consider the possible runs of A against the adversary described above until A halts. Let $\mathbf{T}$ be the random variable that is the value of T when A halts. If the probability that A executes more than 2^{n-1} basic actions before it halts is at least one half, we are done. Otherwise, consider the runs in which A halts before executing 2^{n-1} basic actions. For these runs, $\mathbf{T}$ contains more than half the elements of C_n. Thus, if we consider instead an adversary that chooses uniformly at random a DFA from C_n and returns observations consistent with that DFA, the probability is at least $1/4$ that the output of A will be different from the chosen DFA. Thus, A does not succeed in identification with high probability. □

If it is desired that the target DFAs be strongly connected, it suffices to add the "reset" as a basic action, to take the DFAs back to their start states. Note that in the construction above, the number of basic actions is proportional to the number of states, or "places." What if the number of basic actions is bounded by a constant? A somewhat more elaborate construction shows that even in this case the class of DFAs with distinguishing sequences cannot be learned in polynomial time using exploration only. The key idea is to use binary coding for the symbols a_i, guaranteeing that the new states necessary to implement the coding have outputs that distinguish them "locally" – still requiring the correct sequence of "good choices" to distinguish them globally. The details of this refinement are omitted.

4.3.2. Using a 'weak' distinguishing sequence oracle. In the last section, we showed that, in general, without either a source of counterexamples or a distinguishing sequence we cannot learn a DFA in polynomial time. However, all is not lost. Suppose that we are given an oracle that generates candidate distinguishing sequences, such that it (a) generates sequences s of length at most ϑ, polynomial in $|Q|$, and (b) generates correct distinguishing sequences with probability $1 - \zeta$, bounded below by an inverse polynomial in $|Q|$. We assume that sequences are generated independently. Then we define a procedure that, given two different candidate maps of the environment, determines with high probability which, if either, is in fact the correct map. That is, the procedure fails iff at least one of its inputs is a correct map and it outputs an incorrect map. We then use the oracle to generate a series of possible distinguishing sequences. Each sequence is used to build a map, which is then compared against the current candidate map; the 'better' one is kept as the new candidate. If enough such comparisons are performed, we can ensure that the output of this procedure is a correct map of the environment (and its generating sequence a distinguishing sequence) with high probability $(1 - \xi)$. We describe the procedure in more detail below.

First, we develop the procedure COMPARE for comparing two different maps that chooses a correct map for the environment with probability at least $1 - \kappa$. Given two non-equivalent DFAs and the current state of the robot in each, q_1 and q_2, a sequence of inputs a (a *discriminating sequence*) can be generated (in polynomial time) such that $\gamma(q_1 a) \neq \gamma(q_2 a)$. $|a|$ is bounded by the product of the sizes of the two DFAs (and thus, by m^2). Thus, after we execute a, at least one of the DFAs will predict an output different from that observed. A correct DFA will predict the observed output with probability at least α, while the incorrect map will do so with at most $1 - \alpha$ probability. Hence, we repeatedly generate and execute discriminating sequences, incrementally maintaining a record of each DFA's frequency of prediction error. If we perform enough such trials, if either DFA is correct, it will be the one with the lower estimated prediction error (with high probability). We need to perform enough trials so that the estimated frequency of prediction error is within $\frac{1}{2}(\alpha - \frac{1}{2})$ of the true value, with probability at least $1 - \kappa$. We can now derive $\mathcal{D}$, the number of discriminations needed to do a successful comparison with probability at least $1 - \kappa$. Using the Hoeffding bounds as previously, we get

$$\mathcal{D} \geq \frac{2}{(2\alpha - 1)^2} \ln \frac{2}{\kappa}.$$

COMPARE is invoked with two triples, *i.e.*, COMPARE($\langle M_1, q_1, s_1\rangle$, $\langle M_2, q_2, s_2\rangle$), each consisting of an automaton, the current state of the robot in that automaton, and a distinguishing sequence used in constructing that automaton. The procedure returns one of the two triples.

1. $n_1 \leftarrow n_2 \leftarrow 0$.
2. Repeat $\mathcal{D}$ times:
   (a) If M_1 starting at q_1 is equivalent to M_2 starting at q_2, then return $\langle M_1, q_1, s_1\rangle$;
   (b) Else:
      i. Find a discriminating sequence a such that $\gamma(q_1 a) \neq \gamma(q_2 a)$,
      ii. Execute a and observe output y.
      iii. If $y \neq \gamma(q_1 a)$, $n_1 \leftarrow n_1 + 1$.
      iv. If $y \neq \gamma(q_2 a)$, $n_2 \leftarrow n_2 + 1$.
      v. $q_1 \leftarrow q_1 a$, $q_2 \leftarrow q_2 a$.
3. If $n_1 \leq n_2$, return $\langle M_1, q_1, s_1\rangle$;
4. Else return $\langle M_2, q_2, s_2\rangle$.

Now, given this comparison procedure and a distinguishing sequence oracle as described above, we can determine the number of comparisons C needed to ensure that with probability at least $1 - \xi$ the map output is correct (and its generating sequence is a distinguishing sequence). We first get bounds on the probability of not getting a correct map output, given C comparisons (we assume an initial map is generated first). The probability of a single map-building step not resulting in a correct map is $\leq \zeta + (1-\zeta)\eta = 1-(1-\eta)(1-\zeta)$, since either the sequence used was not a distinguishing sequence (the oracle 'failed'), or it was and the map-building step failed. Note that each map-building step is assumed to be independent from the others (as provided by the algorithm described above). Thus, the probability of not generating a correct map over C trials is bounded by $(1-(1-\eta)(1-\zeta))^C$. The probability of not accepting a correct map in some comparison is bounded by κC. Hence, the total probability of failure is bounded by $(1-(1-\eta)(1-\zeta))^C + \kappa C$. We now wish to choose C so that this probability is less than ξ. We do this by finding bounds that cause each term to be less than $\frac{\xi}{2}$. First, we get a lower bound on C:

$$
\begin{aligned}
(1-(1-\eta)(1-\zeta))^C &\leq \frac{\xi}{2} \\
\text{It suffices if} \quad e^{-(1-\eta)(1-\zeta)C} &\leq \frac{\xi}{2} \\
(1-\eta)(1-\zeta)C &\geq \ln\frac{2}{\xi} \\
C &\geq \frac{1}{(1-\eta)(1-\zeta)}\ln\frac{2}{\xi}
\end{aligned}
$$

Given this lower bound on $\mathcal{C}$, we can determine an upper bound on κ so that $\mathcal{C}$ comparisons suffice. We have:

$$\begin{aligned} \kappa\mathcal{C} &\leq \frac{\xi}{2} \\ \kappa &\leq \frac{\xi}{2\mathcal{C}} \\ \kappa &\leq \frac{\xi(1-\eta)(1-\zeta)}{4\ln\frac{2}{\xi}} \end{aligned}$$

We can now substitute our bound for κ to get a complete bound for $\mathcal{D}$:

$$\mathcal{D} \geq \frac{2}{(2\alpha-1)^2}\ln\left\{\frac{8}{\xi(1-\eta)(1-\zeta)}\ln\frac{2}{\xi}\right\}.$$

The total number of exploration steps required is determined by $\mathcal{C}+1$ invocations of BUILDMAP plus $\mathcal{C}$ invocations of COMPARE, each requiring at most $\mathcal{D}$ executions of action sequences of length at most m^2. Thus, using the bound for BUILDMAP given above, we have a (loose) bound on the number of primitive actions required to identify an automaton with high probability,

$$(\mathcal{C}+1)(|Q|^2|B|\mathcal{M}((\mathcal{N}+1)\vartheta+|Q|)+\mathcal{D}m^2),$$

which is polynomial in the relevant variables. Note that, like β for BUILDMAP, η is a free parameter that may be chosen to minimize the number of steps taken. Intuitively, we can allow each map to be less probably correct, if we check more of them. Of course, η must be inverse polynomial in the number of DFA states.

The complete procedure for learning map with high probability given an oracle for generating distinguishing sequences is described below, where BUILDMAP is assumed to take a distinguishing sequence as input and return as output an automaton and the current state of the robot in that automaton.

1. $s_1 \leftarrow$ oracle output.
2. $\langle M_1, q_1\rangle \leftarrow$ BUILDMAP(s_1).
3. Repeat $\mathcal{C}$ times:
   (A) $s_2 \leftarrow$ oracle output.
   (B) $\langle M_2, q_2\rangle \leftarrow$ BUILDMAP(s_2).
   (C) $\langle M_1, q_1, s_1\rangle \leftarrow$ COMPARE$(\langle M_1, q_1, s_1\rangle, \langle M_2, q_2, s_2\rangle)$
4. Return $\langle M_1, q_1, s_1\rangle$.

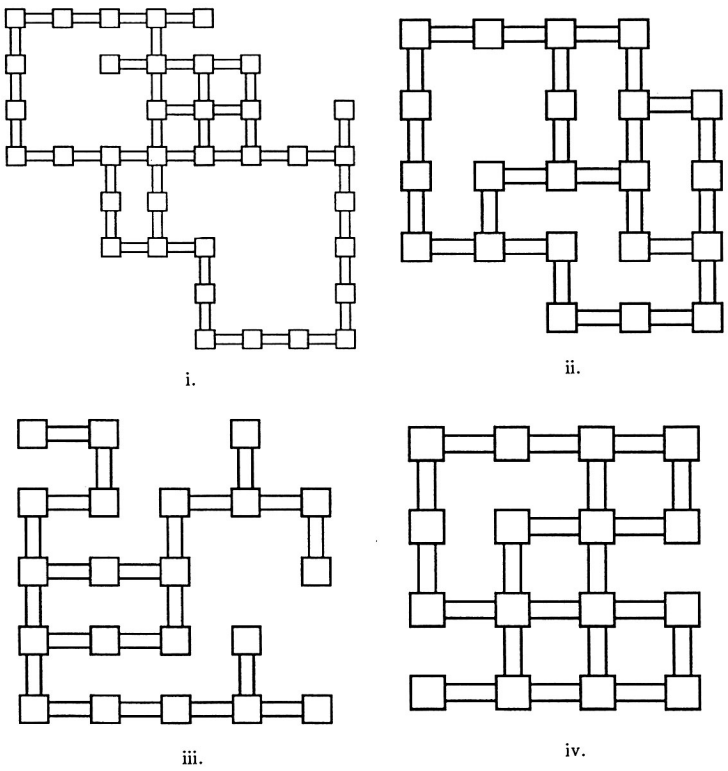

Figure 2. Graphs for office environments.

5. Empirical investigations

This paper is primarily concerned with showing that certain problems have efficient (in the sense of polynomial time) solutions. We suspect that the polynomial functions we have provided to bound the performance of our algorithms are pessimistic. To investigate these suspicions, we implemented the learning algorithms for LOCALIZE and BUILDMAP and experimented with them in simulated environments. In this section, we describe some of those experiments and analyze the resulting data.

All of our results require environments with distinguishing sequences. We hypothesize that many natural environments and office environments in particular possess short distinguishing sequences. To test this hypothesis, we constructed a variety of office environments and determined the length of the shortest distinguishing sequence, assuming that such a sequence existed. Figure 2.i depicts the state-transition graph for the the fifth floor of the Brown CS Department. Three other graphs typical of the ones that we used in our experiments are shown in Figure 2.ii through 2.iv. The length of the shortest distinguishing

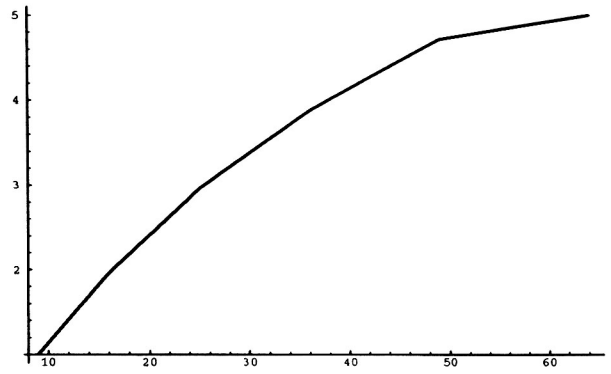

Figure 3. Length of the shortest distinguishing sequence as a function of the number of states in the environment.

sequence for Figure 2.i is four. The lengths of the shortest distinguishing sequences for Figures 2.ii, 2.iii, and 2.iv are two, three, and two respectively.

We generated a large number of graphs by starting with a $d \times d$ grid of locations and constructing a graph of n edges by selecting n pairs of adjacent locations according to a uniform distribution without replacement. The actions available at a location consist of movement in each of the four directions (e.g. N, E, W, S) along axes of the grid; if there is not an edge in a particular direction, the action corresponds to a self transition. The outputs for locations encode the junction type (e.g. L-shaped or T-shaped) and orientation (e.g. facing N, E, W, or S) for a total of sixteen possible outputs, including the degenerate output corresponding to a location with no adjacent corridors. The probability that the robot observes the correct output is α and the probability that it observes an output other than the correct one is $\frac{(1-\alpha)}{15}$. For fixed d with n in the range of d to d^2, the length of the shortest distinguishing sequence is pretty much constant. For the graphs that we have looked at, the length of the shortest distinguishing sequence seems to increase roughly as the square root of the number of states. Figure 3 shows the length of the shortest distinguishing sequence as a function of the number of states in the environment, averaging over sets of environments.

It should be noted that while the outputs encode information about orientation the algorithm does not exploit this information. We ran similar experiments ignoring orientation with only four outputs corresponding to crossing-, straight-, L- and T-shaped junctions with no significant difference in performance. It should also be noted that simpler algorithms would suffice for the case in which the robot can reverse its actions. Again, our algorithm does not exploit this property; we used the undirected graphs because they were easier to generate, as they did not require a check for strong connectedness. We hand-crafted a small number of directed graphs and the algorithms actually performed better than in the corresponding undirected versions of the same graphs.

The theoretical results indicate that for a DFA consisting of 21 states LOCALIZE needs as many as 76206 steps for $\alpha = 0.8$. In our simulations, however, LOCALIZE is successful 100% of the time with no more than 50 steps using a distinguishing sequence of length three.

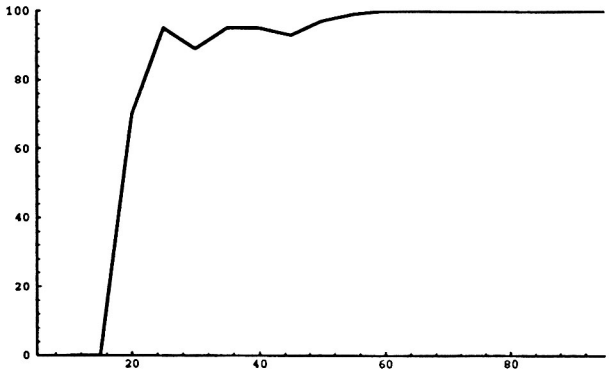

Figure 4. Percentage of correct state identifications for LOCALIZE as a function of the number of repetitions of a distinguishing sequence of length four in the environment of Figure 2.i.

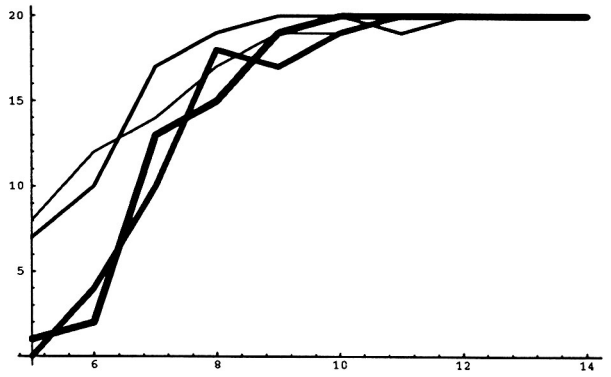

Figure 5. Number of correct map identifications out of 20 for BUILDMAP as a function of the number of experiments for each state transition.

We also observed that the performance of LOCALIZE is largely insensitive to α, continuing to perform with 100% accuracy having executed 50 steps with α as low as 0.5. We believe this is largely due to the fact that errors are distributed uniformly over the incorrect outputs; it is straightforward to construct alternative error distributions that require a lot more work on the part of LOCALIZE. Figure 4 shows a graph of the percentage of correct state identifications for LOCALIZE running on the environment of Figure 2.i as a function of the number repetitions of the distinguishing sequence. This graph typifies the performance of LOCALIZE running on the range of graphs that we considered in our experiments.

We observed similar results for BUILDMAP in that the map-learning algorithm performed quite well with far fewer steps than required by the theoretical results. Figure 5 shows a graph of the percentage of correct map identifications for BUILDMAP as a function of the

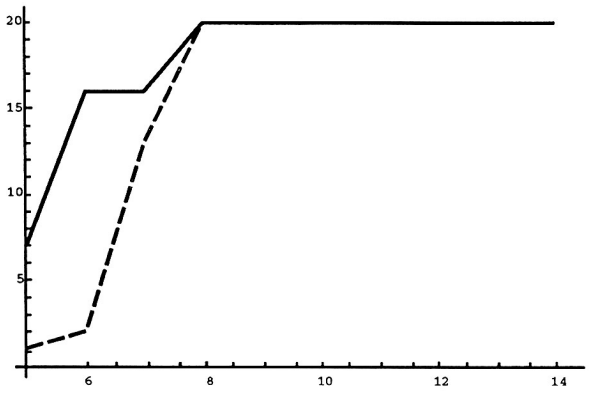

Figure 6. Comparison of BUILDMAP running on directed and undirected versions of the same graph.

number of experiments performed for gathering statistics to infer each state transition. The four lines in Figure 5 correspond to the four graphs in Figure 2 with BUILDMAP generally converging more quickly the smaller the graph.

As mentioned earlier, BUILDMAP achieves no advantage from the fact that our sample graphs are undirected. Figure 6 compares the performance of BUILDMAP running on directed and undirected versions of the same graph. The dashed line corresponds to the undirected graph. In this and similar cases, BUILDMAP actually performed better on the directed versions of undirected graphs.

In the process of experimenting with BUILDMAP, we discovered a simple modification to the algorithm that improves both the theoretical and empirical performance markedly. Executing the distinguishing sequence tends to force the robot into a subset of the set of all states; we refer to the states that the robot ends up in after executing a distinguishing sequence as *sinks*. For instance, there are six sinks for the automaton shown in Figure 2.iii given a particular distinguishing sequence of length three. If there is a single sink or the robot always ends up in the same sink, then the robot will construct a single depth-first search tree of size $|Q|$. In practice, however, the robot tends to bounce back and forth between sinks, constructing several depth-first search trees each of which may grow to some significant percentage of $|Q|$ before one of the trees is complete. Much of the work in constructing the final automaton can be duplicated in constructing different search trees. Since the output of the distinguishing sequence provides a unique signature, the robot can construct a global table of signatures summarizing the information in all of the search trees and thus avoid duplicating search effort. Statistics gathered in one search tree can be combined with statistics gathered in another.

6. Related work

In this paper, we focus on the case in which movement is certain and observation is noisy, and show how a robot might exploit the determinism in movement to enable efficient learning. In

the more general case in which both movement and observation are noisy, we are somewhat pessimistic regarding efficient learning methods even if the robot is given a distinguishing sequence. In (Basye, 1992, Basye et al., to appear), we consider some assumptions about the robot's ability to systematically explore its environment and the source of recognition errors that will allow the robot to learn a model efficiently.

Besides the work mentioned earlier on inferring the smallest consistent automaton, there is other research within the theoretical machine learning community on learning DFAs and their stochastic counterparts. Rudich (1985) provides algorithms for inferring the graph structure and transition probabilities of Markov chains in the limit. Aslam and Rivest (1990) consider the problem of inferring an undirected, degree-bounded, edge-labeled graph from the sequence of edge labels observed in a walk of the graph. Tzeng (1992) considers the problem of exactly learning automata with probabilistic transitions in polynomial time assuming a learning model similar to Angluin's (1987), involving a source of counterexamples. Anyone seriously interested in the problems of inferring automata should read Moore's seminal paper (1956) establishing the conventions for the type of automata employed in this paper and introducing the notions of state identification and distinguishing sequence for such automata.

Yannakakis and Lee (1991) have shown that it is PSPACE-complete to determine whether or not a DFA has a distinguishing sequence. (There exist machines whose shortest distinguishing sequence is exponential in length.) However, they also show that it can be determined in polynomial time whether a DFA has an *adaptive* distinguishing sequence, and, if so, find such a sequence of length $O(|Q|^2)$ in polynomial time. An adaptive distinguishing sequence (as distinct from the *preset* distinguishing sequences employed in this paper) is not really a sequence at all but rather a decision tree whose branches are determined by the robot's observations. As such, we can not directly substitute adaptive distinguishing sequences for the preset distinguishing sequences used in LOCALIZE and BUILDMAP.

Several disciplines besides computer science are concerned with problems similar to the one explored in this paper. The literature on adaptive control (Goodwin & Sin, 1984) and signal processing (Widrow & Stearns, 1985) provides a variety of system identification algorithms, but researchers in these areas are primarily interested in learning in the limit. Hidden Markov Models (HMMs) have both probabilistic transitions and probabilistic outputs. Levinson et al. (1983) provide a survey of HMM learning methods and their application to speech understanding. Rimey and Brown (1991) show how HMMs can be applied in learning action sequences to control camera movements in machine vision. The work on HMMs generally assumes that the underlying automaton can be reset to its initial state and is primarily concerned with learning in the limit.

One important direction for future research involves exploiting structure available in the state space. The algorithms described in this paper are polynomial in the size of the state space; however, the state space is likely to be exponential in the number of features provided by the robot's perceptual capabilities. An interesting problem for future research concerns inferring and then taking advantage of independence among perceptual features. Donald and Jennings (1991) describe methods for constructing representations of space based on equivalence classes induced by a robot's perceptual capabilities. Rivest and Schapire's *diversity* representation of the state space (Rivest & Schapire, 1987) provides an approach

to reducing the storage required in representing automata. Bachrach (1992) shows how to implement Rivest and Schapire's learning algorithm in a connectionist architecture. In some problems, it is advantageous for a representation to coalesce distinct environmental states that require the same response from the robot. Whitehead and Ballard (1991) deal with the problems that arise in reinforcement learning when perception maps different states to the same internal representation.

7. Conclusions

In this paper, we provide general methods for the inference of finite automata with stochastic output functions. We demonstate that it is possible to exploit determinism in movement to enable efficient learning. In previous work (Basye & Dean, 1989, Basye et al., 1989), we concentrated on problems in which the state transition function is stochastic (e.g., in the context of map learning, this means that the robot's navigation procedures are prone to error). In future work, we intend to combine our results to handle uncertainty in both movement and observation. We are also interested in identifying and exploiting additional structure inherent in real environments (e.g., office environments represent a severely restricted class of planar graphs) and in exploring more forgiving measures of performance (e.g., it is seldom necessary to learn about the entire environment as long as the robot can navigate efficiently between particular locations of interest).

With regard to more forgiving measures of performance, it may be possible to extend the techniques of this paper to find ϵ-approximations rather than exact solutions by using the *probably approximately correct* learning model of Valiant (1984). Such extensions would require the introduction of a distribution, Pr, governing both performance evaluation and exploration. As one simple example, suppose that the robot operates as follows. Periodically the robot is asked to execute a particular sequence of actions chosen according to Pr; we assume that, prior to being asked to execute the sequence, it has performed localization and that immediately following it executes the distinguishing sequence. Performance is measured in terms of the robot's ability to plan paths between locations identified by their distinguishing sequence signatures, where the paths are generated according to Pr. The problem with this approach is that the starting locations of the sequences are not determined solely by Pr. In fact, all that Pr governs is $\Pr(s|q_{\text{start}})$, where q_{start} is the start state. In order for the above approach to work, the marginal distribution governing the possible starting locations of the robot, $\Pr(q_{\text{start}})$, must be the same for both training and performance evaluation. In future research, we will be exploring a number of different approaches to specifying ϵ-approximations.

Acknowledgments

This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601, by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency

of the Department of Defense under Contract No. IRI-8905436. Dana Angluin is supported by NSF Grant CCR-9014943. Sean Engelson is supported by the Fannie and John Hertz Foundation.

Notes

1. There has to be some requirement on the size of the DFA, otherwise the robot could choose the DFA corresponding to the complete chain of inputs and outputs.
2. In (Basye, 1992, Basye et al., to appear), we consider a more efficient approach for the case in which γ is unambiguous which, under more restrictive conditions than those required in this paper, infers the underlying automaton by estimating the transition probabilities on the observed states (i.e., the probability of observing i next given that the robot observed j last).
3. The arguments in this paper require that $\alpha > \frac{1}{2}$ and they exploit the separation between α and $\frac{1}{2}$. It appears to be straightforward to relax this requirement somewhat. Let P_{ij} be the probability of observing output y_j given that the robot is in the state q_i. Let $P_i^* = \max_j\{P_{ij}\}$. We assume that there exists one output that is observed more frequently than any other (i.e., if $P_{ij} = P_{ik} = P_i^*$ then $j = k$). Let s_i be the difference between the most frequently observed and the second most frequently observed outputs (*i.e.*, $s_i = \min_j\{P_i^* - P_{ij} | P_i^* < P_{ij}\}$), and s be a lower bound on the s_i (i.e., $s = \min_i\{s_i\}$). We see no obstacle (other than a slightly more complicated proof) to extending our arguments to work with a separation of s. We have observed empirically that the algorithms described in this paper work well without modification in many cases in which $\alpha \leq \frac{1}{2}$.
4. If the outputs are not known, then the table can be constructed incrementally, adding new outputs as they are observed.
5. Following Step 2, the next action should be the first action in s.
6. The idea of using multiple searches starting from the states resulting from LOCALIZE is similar to the way in which Rivest and Schapire (1989) run multiple versions of Angluin's L^* algorithm (Angluin, 1987) starting from the states the robot ends up in after executing a given homing sequence.
7. Note that we must use m, an *a priori* bound on the number of state in order to choose $\mathcal{M}$ and $\mathcal{N}$ to guarantee a certain level of confidence. The number of nodes that are identified in the depth-first search need not be estimated in advance, however, so we use $|Q|$, the actual size of the state space here, rather than m.
8. In the BUILDMAP algorithm presented here, a number of searches may be carried out in parallel, potentially duplicating work unecessarily. In fact, though, the different depth-first searches need not be treated as independent; a global signature table can be constructed by the separate searches working together. This removes a factor of $|Q|$ from the number of steps required, but complicates the algorithm somewhat.

References

Angluin, D. (1978). On the complexity of minimum inference of regular sets. *Information and Control* 39:337–350.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation* 75:87–106.

Aslam, J.A., & Rivest, R.L. (1990). Inferring graphs from walks. In *Proceedings COLT-88.*

Bachrach, J.R. (1992). Connectionist modeling and control of finite state environments. Technical Report 92-6, University of Massachusetts at Amherst Department of Computer and Information Science.

Basye, K., & Dean, T. (1989). Map learning with indistinguishable locations. In *Proceedings of the Fifth Workshop on Uncertainty in AI.* 7–13.

Basye, K., Dean, T., & Vitter, J.S. (1989). Coping with uncertainty in map learning. In *Proceedings IJCAI 11.* IJCAII. 663–668.

Basye, K., Dean, T., & Kaelbling, L. (to appear). Learning dynamics: System identification for perceptually challenged agents. *Artificial Intelligence.*

Basye, K. (1992). A framework for map construction. Technical report, Brown University Department of Computer Science, Providence, RI.

Dean, T., Basye, K., Chekaluk, R., Hyun, S., Lejter, M., & Randazza, M. (1990). Coping with uncertainty in a control system for navigation and exploration. In *Proceedings AAAI-90*. AAAI. 1010–1015.

Donald, B., & Jennings, J. (1991). Sensor interpretation and task-directed planning using perceptual equivalence classes. Technical Report CU-CS-TR, Cornell University Computer Science Department.

Dudek, G., Jenkins, M., Milios, E., & Wilkes, D. (1988). Robotic exploration as graph construction. Technical Report RBCV-TR-88-23, University of Toronto.

Gold, E.M. (1972). System identification via state characterization. *Automatica* 8:621–636.

Gold, E.M. (1978). Complexity of automaton identification from given sets. *Information and Control* 37:302–320.

Goodwin, G.C., & Sin, K.S. (1984). *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, New Jersey.

Kaelbling, L., Basye, K., Dean, T., Kokkevis, E., & Maron, O. (1992). Robot map-learning as learning labeled graphs from noisy data. Technical Report CS-92-15, Brown University Department of Computer Science.

Kearns, M., & Valiant, L.G. (1989). Cryptographic limitations on learning boolean functions and finite automata. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*. 433–444.

Kuipers, B.J., & Byun, Y.-T. (1988). A robust, qualitative method for robot spatial reasoning. In *Proceedings AAAI-88*. AAAI. 774–779.

Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive Science* 2:129–153.

Levinson, S.E., Rabiner, L.R., & Sondhi, M.M. (1983). An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *The Bell System Technical Journal* 62(4):1035–1074.

Levitt, T.S., Lawton, D.T., Chelberg, D.M., & Nelson, P.C. (1987). Qualitative landmark-based path planning and following. In *Proceedings AAAI-87*. AAAI. 689–694.

Moore, E.F. (1956). Gedanken-experiments on sequential machines. In *Automata Studies*. Princeton University Press, Princeton, New Jersey. 129–153.

Pitt, L., & Warmuth, M.K. (1989). The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*. 421–432.

Rimey, R.D., & Brown, C.M. (1991). Controlling eye movements with hidden Markov models. *International Journal of Computer Vision* 7(1):47–65.

Rivest, R.L., & Schapire, R.E. (1987). Diversity-based inference of finite automata. In *Proceedings of the Twenty Eighth Annual Symposium on Foundations of Computer Science*. 78–87.

Rivest, R.L., & Schapire, R.E. (1989). Inference of finite automata using homing sequences. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*. 411–420.

Rudich, S. (1985). Inferring the structure of a Markov chain from its output. In *Proceedings of the Twenty Sixth Annual Symposium on Foundations of Computer Science*. 321–326.

Schapire, R.E. (1991). The design and analysis of efficient learning algorithms. Technical Report MIT/LCS/TR-493, MIT Laboratory for Computer Science.

Tzeng, W.-G. (1992). Learning probabilistic automata and Markov chains via queries. *Machine Learning* 8:151–166.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM* 27:1134–1142.

Whitehead, S.D., & Ballard, D.H. (1991). Learning to perceive and act by trial and error. *Machine Learning* 7.

Widrow, B., & Stearns, S.D. (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, N.J.

Yannakakis, M., & Lee, D. (1991). Testing finite state machines. In *ACM Symposium on Theoretical Computing*. ACM Press. 476–485.

Received December 13, 1993
Accepted April 11, 1994
Final Manuscript April 21, 1994