Learning and Discovery of Predictive State Representations in Dynamical Systems with Reset

Michael R. James MRJAMES@UMICH.EDU
Satinder Singh BAVEJA@UMICH.EDU
Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109

Abstract

Predictive state representations (PSRs) are a recently proposed way of modeling controlled dynamical systems. PSR-based models use *predictions* of observable outcomes of *tests* that could be done on the system as their state representation, and have model parameters that define how the predictive state representation changes over time as actions are taken and observations noted. Learning PSR-based models requires solving two subproblems: 1) *discovery* of the tests whose predictions constitute state, and 2) *learning* the model parameters that define the dynamics. So far, there have been no results available on the discovery subproblem while for the learning subproblem an approximate-gradient algorithm has been proposed (Singh et al., 2003) with mixed results (it works on some domains and not on others). In this paper, we provide the first discovery algorithm and a new learning algorithm for linear PSRs for the special class of controlled dynamical systems that have a *reset* operation. We provide experimental verification of our algorithms. Finally, we also distinguish our work from prior work by Jaeger (2000) on observable operator models (OOMs).

1. Introduction

Predictive state representations (PSRs; Littman, Sutton, & Singh, 2001) are a recently proposed way (inspired by the work of Jaeger (2000) and Rivest & Schapire (1994)) of building models of controlled dynamical systems. PSRs capture the state of a system as a vector of predictions or outcome probabilities for tests (or experiments) that one can do on the system. A test is a sequence of action-observation pairs and its prediction is the probability of the test's observation-sequence happening if the test's action-sequence were to be executed on the system. PSR-based models have parameters that define how the predictive state representation changes over time as actions are executed and observations noted. A novel[1] aspect of PSR-based models is that their state is expressed entirely in terms of observable quantities. In contrast, partially observable Markov decision process or POMDP-based models express state in terms of probability distributions over unobservable and hypothetical underlying-states of the system (see, e.g., Lovejoy, 1991; Littman, 1996). Despite this key difference, PSR-models are as flexible and powerful as POMDP models; indeed, Littman et al. showed that any dynamical system that can be modeled as a POMDP can also be modeled as a PSR of size no larger than that of the POMDP model. On the other hand, because of this key difference, there is a pressing and yet unmet need to develop algorithms for learning PSR-models from streams of experience because existing POMDP-based model-learning methods scale quite poorly (Littman et al., 2001; Shatkay & Kaelbling, 1997).

Learning a PSR-based model from data has two important subproblems: 1) *discovery* of the tests whose predictions constitute state, and 2) *learning* the model parameters that define how the predictive state representation is updated over time as actions are executed and observations noted. So far, there have been no results for the discovery problem, while for the learning problem an approximate gradient-based algorithm has been proposed by Singh et al. (2003) with mixed

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the first author.

[1]The state in history-window based or n^{th}-order Markov models is also expressed in terms of observable quantities but these models are less flexible than PSR models (Littman et al., 2001) and won't be considered further in this paper.

results (it works on some domains and not on others). In this paper, we present the first linear-PSR discovery algorithm and a new learning algorithm, both for the restricted class of dynamical systems that have a *reset*. We present evaluations of our algorithms on the same domains (with an added reset) that the previous gradient algorithm got mixed results on.

1.1. Related Work

PSR-based models are closely related to Jaeger's (2000; 2003) Observable Operator Models or OOMs. These were developed initially for uncontrolled dynamical systems but were then extended to input-output OOMs, or IO-OOMs, to handle controlled dynamical systems. Jaeger developed two different versions, interpretable and non-interpretable, of both OOMs and IO-OOMs; of these, only the interpretable versions are of practical interest because discovery and learning algorithms can be developed only for them. In recent work (Singh et al., 2004), we have shown that the class of dynamical systems that can be modeled by interpretable IO-OOMs is a small subset of systems that can be modeled as POMDPs which in turn is a strict subset of systems that can be modeled as PSRs, i.e., that PSRs are significantly more general than interpretable IO-OOMs.

2. Preliminaries

We consider finite, discrete-time, and controlled dynamical systems, henceforth simply dynamical systems, that accept actions from a discrete set $\mathcal{A}$, and produce observations from a discrete set $\mathcal{O}$. Letting $a_i \in \mathcal{A}$ and $o_i \in \mathcal{O}$ denote the action and observation at time i, the probability of some *history* $h = \{a^1 o^1 a^2 o^2 \ldots a^n o^n\}$ is the conditional probability of the observation sequence $o^1 o^2 \ldots o^n$ being obtained if the action sequence $a^1 a^2 \ldots a^n$ were executed from time 1 onward, i.e., $P(h) = prob(o_1 = o^1, o_2 = o^2, \ldots, o_n = o^n | a_1 = a^1, a_2 = a^2, \ldots, a_n = a^n)$. A dynamical system is characterized by a probability distribution over all possible histories, $P : \{\mathcal{AO}\}^* \rightarrow [0, 1]$. A *test*, like a history, is also a sequence of action-observation pairs, but unlike a history, a test is not constrained to start at time 1. The conditional probability of a test $t = a^1 o^1 a^2 o^2 \ldots a^m o^m$ at some history h (w.l.o.g., of length n), is $P(t|h) = P(ht)/P(h) = prob(o_{n+1} = o^1, o_{n+2} = o^2, \ldots, o_{n+m} = o^m | h, a_{n+1} = a^1, a_{n+2} = a^2, \ldots, a_{n+m} = a^m)$, i.e., it is the probability of obtaining the test's observation-sequence if the test's action-sequence were executed from history h onward. For ease of exposition, henceforth, we will abbreviate expressions like the right-hand side of the definition of $P(t|h)$ above as simply $prob(o^1 o^2 \ldots o^m | h a^1 a^2 \ldots a^m)$.

A set of tests $Q = \{q^1 q^2 \ldots q^{|Q|}\}$ constitutes a PSR if its *prediction-vector*, $P(Q|h) = [P(q^1|h) P(q^2|h) \ldots P(q^{|Q|}|h)]$ is a sufficient statistic for all histories, i.e., for any test t, $P(t|h) = f_t(P(Q|h))$ for some function f_t — i.e., $P(Q|h)$ captures all the information in h relevant to predicting the future. We will distinguish such a set of tests that constitutes a PSR by calling it a *core* set of tests and reserving the label Q for it. In general, the functions f_t can be linear or non-linear. In this paper we will only consider *linear PSRs* in which

$$\forall t \; P(t|h) = P^T(Q|h) m_t, \quad \text{for a weight vector } m_t \quad (1)$$

(by default vectors are assumed to be column vectors and so the transpose operator, $\cdot^T$, is used to obtain row vectors). On taking action a from some history h and observing o, one can update the prediction-vector for Q as follows: for each $q^i \in Q$

$$P(q^i|hao) = \frac{P(aoq^i|h)}{P(ao|h)} = \frac{P^T(Q|h) m_{aoq^i}}{P^T(Q|h) m_{ao}} \quad (2)$$

where we used the fact that aoq^i is just another test. Therefore, a linear-PSR model is specified by core-set Q and the *model-parameters* which are the weight vectors m_{aot} (all of size $|Q| \times 1$) for all $a \in \mathcal{A}$, all $o \in \mathcal{O}$ and for all $t \in Q \cup \{\phi\}$ (where the null string ϕ was added because of the parameters m_{ao} in the denominator of the right-hand side of Equation 2). Learning a PSR model from data involves two subproblems: the **discovery** of a core-set Q, and **learning** the model-parameters.

In building a POMDP representation of a dynamical system one starts by hypothesizing a set $\mathcal{S} = \{1, 2, \ldots, |\mathcal{S}|\}$ of unobservable states. For any history h, the counterpart to the prediction vector $P(Q|h)$ is the belief-state $b(\mathcal{S}|h) = [prob(1|h) \; prob(2|h) \; \ldots \; prob(|\mathcal{S}||h)]$ that represents the probability of the system being in each underlying state of $\mathcal{S}$ at history h. The model-parameters are a set of transition-probability matrices T^a (for all $a \in \mathcal{A}$) such that T^a_{ij} is the probability of a transition to state $j \in \mathcal{S}$ upon taking action a in state $i \in \mathcal{S}$, and a set of diagonal observation probability matrices $O^{a,o}$ for all $a \in \mathcal{A}$ and $o \in \mathcal{O}$ such that $O^{a,o}_{ii}$ is the probability of receiving observation o upon taking action a and transitioning to state i. On taking action a in history h and observing o, the belief-state can be updated as follows:

$$b^T(\mathcal{S}|hao) = \frac{b^T(\mathcal{S}|h) T^a O^{a,o}}{b^T(\mathcal{S}|h) T^a O^{a,o} 1_{|\mathcal{S}|}}$$

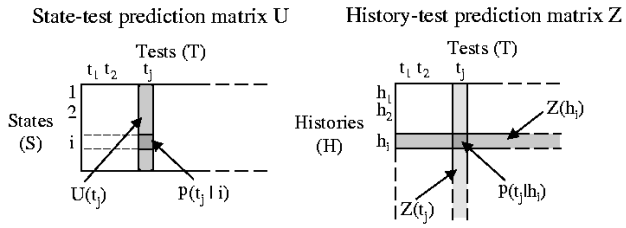

Figure 1. Graphical representation of state-test prediction matrix and history-test prediction matrix.

where 1_n is a $n \times 1$ vector of all 1's. Learning a POMDP model from data also involves discovery and learning subproblems, where the discovery problem is to figure out how many states to use, while the learning problem is to estimate the model-parameters.

We can relate PSR-models to POMDP-models using a **conceptual** *state-test prediction matrix* U (see Figure 1) whose rows correspond to the states in $\mathcal{S}$ and whose columns correspond to all possible tests arranged in order of increasing size, and within a size in some lexicographic order (we let $\mathcal{T}$ denote the set of all possible tests in this ordering). The entry U_{ij} is the prediction for the test t_j (the one associated with column j), given that the state of the system is i. Then, by construction, for any h and any t_j, $P(t_j|h) = b^T(S|h)U(t_j)$, where $U(t_j)$ is the jth column of U, and for any set of tests C, the vector $P^T(C|h) = b^T(S|h)U(C)$ where $U(C)$ is the submatrix corresponding to the tests in C. Thus, the matrix U allows us to translate belief-states into prediction vectors.

We can also motivate and derive PSR core-tests Q from the U matrix as follows. Let M be the set of tests corresponding to any maximal-set of linearly-independent columns of U. Then for any test t, $U(t) = U(M)w_t$, for some vector of weights w_t (by construction of M the column in U for any test must be linearly dependent on the columns of sub-matrix $U(M)$). Therefore for all t, $p(t|h) = b^T(S|h)U(t) = b^T(S|h)U(M)w_t = P^T(M|h)w_t$ which is exactly the requirement for M to constitute a linear-PSR (see Equation 1). Note that the number of linearly independent columns is the rank of the matrix U and is therefore upper-bounded by the number of rows $|\mathcal{S}|$. Thus, through this conceptual matrix U we have proven that for any POMDP, there exists a linear-PSR with $\leq |\mathcal{S}|$ core-tests and that any maximal-set of linearly-independent columns of U correspond to core-tests Q. *Note that the core-tests are not unique.* Littman et al. also prove by a constructive argument that there exists a core-set Q in which no test is longer than $|\mathcal{S}|$. These results together suggest the power and flexibility of PSRs as a representation.

3. Discovery and Learning using History-Test Prediction Matrix

In this section we derive a discovery and learning algorithm for dynamical systems with reset and provide some partial analysis supporting our algorithms.

First, note that if we could somehow estimate the matrix U from data, then we could discover core-tests Q by finding a maximal-set of linearly-independent columns of the estimated U. But, we cannot directly estimate U because its rows correspond to a set $\mathcal{S}$ that is unobservable. A key insight in this paper is to replace the state-test prediction matrix U used in the original PSR paper (Littman et al., 2001) by a new *history-test prediction matrix* $\mathcal{Z}$ that as we will show can be estimated from data in certain classes of dynamical systems (e.g., those with reset), and that has many of the same desirable properties that U has. The history-prediction matrix $\mathcal{Z}$ (see Figure 1) has columns corresponding to the tests in $\mathcal{T}$ just as for U, but has rows corresponding to all possible histories $\mathcal{H}$ instead of the hypothetical $\mathcal{S}$ as in U. Note that $\mathcal{H} = \mathcal{T} \cup \phi$ where ϕ is the null history. The entry $\mathcal{Z}_{ij} = p(t_j|h_i)$, is the prediction of the test t_j given history h_i. Note that $\mathcal{Z}$, unlike U, is expressed entirely in terms of observable quantities and thus, as we will show next, can be directly used to derive discovery and learning algorithms.

But first, some notation. For any set of tests C, let $\mathcal{Z}(C)$ ($U(C)$) denote the submatrix of $\mathcal{Z}$ (U) containing the columns corresponding to the tests in C. Also, let $Q_{\mathcal{T}}$ denote the tests corresponding to a maximal-set of linearly independent columns of $\mathcal{Z}$ and let $Q_{\mathcal{H}}$ be the histories corresponding to a maximal-set of linearly independent rows of $\mathcal{Z}$. Again, note that both $Q_{\mathcal{T}}$ and $Q_{\mathcal{H}}$ are not unique.

Lemma 1 *For any dynamical system and its corresponding history-test prediction matrix $\mathcal{Z}$, the tests corresponding to any maximal-set of linearly-independent columns of $\mathcal{Z}$ constitute a linear-PSR. Furthermore, if the dynamical system is a POMDP with state-test prediction matrix U, the size of the linear-PSR derived from $\mathcal{Z}$ will be no more than the size of the linear-PSR derived from U.*

Proof For any test t, $\mathcal{Z}(t) = \mathcal{Z}(Q_{\mathcal{T}})w_t$ for some w_t (because every column of $\mathcal{Z}$ is linearly dependent on the columns of $\mathcal{Z}(Q_{\mathcal{T}})$). $P(t|h)$ is the element corresponding to row h in vector $\mathcal{Z}(t)$, while $P^T(Q_{\mathcal{T}}|h)$ is the row of $\mathcal{Z}(Q_{\mathcal{T}})$ corresponding to h. Therefore, $\forall t\, P(t|h) = P^T(Q_T|h)w_t$, which in turn implies that the set $Q_{\mathcal{T}}$ constitutes a linear-PSR (see Equation 1).

For the second part of the lemma, let $b^T(S|\mathcal{H})$ be a $(\infty \times |S|)$ matrix whose i^{th} row corresponds to the belief-state for history h_i. Then $\mathcal{Z} = b^T(S|\mathcal{H})U$. By a standard result from linear algebra the rank of a product of matrices is less than or equal to the smaller of the ranks of the two matrices being multiplied. Therefore the rank of $\mathcal{Z}$ is upper bounded by the rank of U. The proof then follows from the fact that the size of the linear PSR is equal to the rank. □

Lemma 1 shows that $Q_{\mathcal{T}}$ constitutes a linear-PSR, and hence we will refer to $Q_{\mathcal{T}}$ as core-tests. By analogy we will refer to $Q_{\mathcal{H}}$ as core histories.

Lemma 2 *For a dynamical system that is a POMDP, the set of belief-state vectors corresponding to the set Q_H of core-histories are linearly independent.*

Proof Let $H \subseteq \mathcal{H}$ be an arbitrary finite set of histories, and let $b(\mathcal{S}|H)$ be the associated $|\mathcal{S}| \times |H|$ belief-state matrix (each column is a belief-state). If $b(\mathcal{S}|h)$, the belief-state for some history h, is linearly dependent on the set of belief-states associated with the histories in H, i.e., if $b^T(S|h) = w^T b^T(\mathcal{S}|H)$, then $\mathcal{Z}(h) = P^T(\mathcal{T}|h) = b^T(\mathcal{S}|h)U(\mathcal{T}) = w^T b^T(\mathcal{S}|H)U(\mathcal{T}) = w^T P(\mathcal{T}|H) = w^T \mathcal{Z}(H)$. In other words if the belief-state for some history h is linearly dependent on the belief-states of some set of histories H, then the row corresponding to h in $\mathcal{Z}$ will be linearly dependent on the rows corresponding to histories H in $\mathcal{Z}$. Contrapositively, this implies that if we find a set of rows of $\mathcal{Z}$ that are linearly independent of each other then their associated belief-states are linearly independent. □

Lemma 2 is interesting because it shows that core-histories are a kind of *basis* set for the space of histories $\mathcal{H}$, in that the belief-state vectors corresponding to core-histories span the full space of feasible belief-state vectors.

Lemma 3 *The set of tests and histories corresponding to a set of linearly independent columns and rows of any submatrix of $\mathcal{Z}$ are subsets of core-tests and core-histories respectively.*

Proof A set of columns (rows) that are linearly independent in any submatrix of $\mathcal{Z}$ are also linearly independent in the full matrix $\mathcal{Z}$. The proof follows from Lemmas 1 and 2 □

Lemma 3 frees us from having to deal with the infinite matrix $\mathcal{Z}$ and instead allows us to consider finite, and hopefully small, submatrices of $\mathcal{Z}$ in building discovery and learning algorithms.

3.1. Analytical Discovery and Learning (ADL) Algorithm

We present our algorithm for discovery and learning in 2 stages. In the first stage, we will develop an algorithm for discovery alone as well as an algorithm for both discovery and learning under the assumption that the algorithm has the ability to somehow compute prediction $P(t|h)$ exactly for any test t and history h. In the second stage, we will remove that assumption by allowing the algorithm to instead experiment with the dynamical system itself and empirically estimate $P(t|h)$ — the second-stage algorithms will be the first-stage algorithms with the estimated test-predictions replacing the true test-predictions but with additional machinery to deal with potential problems introduced by inaccurate test-predictions.

Under the assumption that we can compute $P(t|h)$ exactly, the *analytical discovery algorithm* (AD) works iteratively as follows. At the first iteration the algorithm computes a submatrix of $\mathcal{Z}$, denoted $\mathcal{Z}_1$, containing all histories up to length one and all tests up to length one. Then it calculates the rank of this submatrix, ν_1, the set of core-tests found so far, $Q_{\mathcal{T}_1}$ (these are any ν_1 linearly independent tests in $\mathcal{Z}_1$), and the set of core-histories found so far, $Q_{\mathcal{H}_1}$ (these are any ν_1 linearly independent histories in $\mathcal{Z}_1$). At the next iteration, it computes the submatrix of $\mathcal{Z}$ with columns corresponding to the union of the tests in $Q_{\mathcal{T}_1}$ and all one-step extensions to all tests in $Q_{\mathcal{T}_1}$, and with rows corresponding to the union of the histories in $Q_{\mathcal{H}_1}$ and all one-step extensions to all histories in $Q_{\mathcal{H}_1}$. The rank ν_2, core-tests $Q_{\mathcal{T}_2}$, and core-histories $Q_{\mathcal{H}_2}$ are then calculated. This process is repeated until the rank remains the same for two consecutive iterations (this is AD's *stopping condition*). If the algorithm stops after iteration i, then it returns $Q_{\mathcal{T}_i}$ and $Q_{\mathcal{H}_i}$ as the discovered core-tests and core-histories respectively.

For any controlled dynamical system with a history-prediction matrix $\mathcal{Z}$ of rank n, the AD algorithm cannot execute for more than $n+1$ iterations; for to continue iterating the rank must increase by least one at every iteration. Lemma 3 shows that the core-tests and core-histories returned by the AD algorithm are all *correct*, i.e., there exist core-tests and core-histories of the dynamical system that are supersets of the discovered core-tests and histories. Or in other words, the discovered core tests and histories correspond to linearly independent columns and rows of the full $\mathcal{Z}$ matrix. However, note that this does not show that the AD algorithm cannot stop prematurely (i.e., before discovering all n core tests). Thus it is possible that

the algorithm may stop without having discovered all the core-tests and core-histories. We empirically investigated this possibility and report the results in the section on empirical results. Subsequent to our experimental work we were able to answer this question theoretically as well (and we present a very brief sketch of our theoretical result in the appendix).

We also define an *analytical discovery and learning* (ADL) algorithm. Recall that the learning problem is to compute the various m vectors ($\forall a \in \mathcal{A}, o \in \mathcal{O}, t \in Q_{\mathcal{T}}$), in the following prediction equations:

$$P(ao|Q_{\mathcal{H}}) = P(Q_{\mathcal{T}}|Q_{\mathcal{H}})m_{ao} \text{ and}$$

$$P(aot|Q_{\mathcal{H}}) = P(Q_{\mathcal{T}}|Q_{\mathcal{H}})m_{aot}$$

where $Q_{\mathcal{T}}$ and $Q_{\mathcal{H}}$ are the unknown set of complete core tests and histories. Note that these are the same equations as in the update Equation 2 except that we have separated the numerator and the denominator and clustered the equations into a matrix form. The vectors $P(ao|Q_{\mathcal{H}})$ and $P(aot|Q_{\mathcal{H}})$ are composed of elements corresponding to $P(ao|h)$ and $P(aot|h)$ for all $h \in Q_{\mathcal{H}}$. The matrix $P(Q_{\mathcal{T}}|Q_{\mathcal{H}})$ is the submatrix of $\mathcal{Z}$ with columns corresponding to the tests in $Q_{\mathcal{T}}$ and rows corresponding to $Q_{\mathcal{H}}$. By the definitions of $Q_{\mathcal{T}}$ and $Q_{\mathcal{H}}$, $P(Q_{\mathcal{T}}|Q_{\mathcal{H}})$ is a square and invertible matrix. Under our assumption that the algorithm can compute $P(t|h)$ for any t and h, it could compute $P(Q_{\mathcal{T}}|Q_{\mathcal{H}})$, $P(ao|Q_{\mathcal{H}})$ and $P(aot|Q_{\mathcal{H}})$ and then calculate $m_{ao} = P^{-1}(Q_{\mathcal{T}}|Q_{\mathcal{H}})P(ao|Q_{\mathcal{H}})$, and $m_{aot} = P^{-1}(Q_{\mathcal{T}}|Q_{\mathcal{H}})P(aot|Q_{\mathcal{H}})$.

Of course, $Q_{\mathcal{T}}$ and $Q_{\mathcal{H}}$ are unknown. And so ADL first runs the analytical discovery algorithm to obtain correct but potentially incomplete core tests and histories $Q'_{\mathcal{T}}$ and $Q'_{\mathcal{H}}$ (where the prime symbol on Q denotes this possibility of incompleteness). The algorithm then computes $P(Q'_{\mathcal{T}}|Q'_{\mathcal{H}})$, $P(ao|Q'_{\mathcal{H}})$ and $P(aot|Q'_{\mathcal{H}})$ and finally calculates $m'_{ao} = P^{-1}(Q'_{\mathcal{T}}|Q'_{\mathcal{H}})P(ao|Q'_{\mathcal{H}})$, and $m'_{aot} = P^{-1}(Q'_{\mathcal{T}}|Q'_{\mathcal{H}})P(ao|Q'_{\mathcal{H}})$. Again, $P(Q'_{\mathcal{T}}|Q'_{\mathcal{H}})$ is invertible because $Q'_{\mathcal{T}}$ and $Q'_{\mathcal{H}}$ are correct if incomplete. Finally, if analytical discovery returns a complete set of core tests and histories then it is clear that ADL will compute an exact PSR-based model of the dynamical system, else it will only be an approximate model.

3.2. Discovery and Learning (DL) algorithm

Of course, the analytical discovery and learning algorithm will rarely be feasible because in general there will be no way of computing $P(t|h)$ exactly. So now we turn to the more realistic case where the algorithm has access to the controlled dynamical system which it wishes to model and can somehow use that system to estimate $P(t|h)$ rather than compute it exactly. In this paper we consider dynamical systems with reset, i.e., systems in which the algorithm can choose to reset the system to the null history (corresponding to the first row of the $\mathcal{Z}$ matrix). Note that while this is a strong assumption, it is not as restrictive as it may seem, e.g., a reset can always be done in any simulated system, and it can also be achieved (inefficiently) in any episodic system in which the system automatically resets after each episode.

The key advantage of dynamical systems with reset is that we can repeat history and thereby repeatedly generate an *iid* sample from $P(t|h)$ for any (feasibly-sized) t and h, by the following algorithm. First, we generate the history by repeatedly reseting the system and executing h's action sequence, until h's observation sequence occurs. Then, we execute t's action sequence and return success if t's observation sequence occurs, otherwise return failure. However, this sampling procedure is obviously grossly inefficient. We use the following sampling algorithm instead. If, for whatever reason, the algorithm takes action sequence $a_1a_2 \ldots a_k$ after a reset and observes sequence $o_1o_2 \ldots o_k$, it can update the estimated prediction for all the contiguous history-test pairs contained in this sequence. For example, consider the sequence $a_1o_1a_2o_2$. Within this sequence, we can gather data for history ϕ and successful test a_1o_1; history ϕ and successful test $a_1o_1a_2o_2$; and history a_1o_1 and successful test a_2o_2. In addition, there are many unsuccessful tests: e.g., for history a_1o_1 all tests with $a_2o : (o \neq o_2)$ are unsuccessful, and so on. Note that while this sampling algorithm produces a lot of data from each sequence the empirical predictions for the constituent history-test pairs are correlated. On the other hand the samples obtained for any history-test pair from different sequences are uncorrelated and thus any bad effect of correlation among the errors in the estimated predictions decreases rapidly with increasing number of data sequences.

The discovery and learning (DL) algorithm we present in this section is nearly identical to the ADL algorithm of the previous section. DL takes an input parameter, n, which is a lower-bound on the number of samples we will generate for any prediction estimate we use in DL. Just like in ADL, the DL(n) algorithm will first do discovery and then learning. Discovery in ADL proceeds in iterations computing the rank, a set of core tests, and a set of core histories for a submatrix of $\mathcal{Z}$ at each iteration. Discovery in DL(n) proceeds in iterations in exactly the same manner using the sampling algorithm defined above to generate at least n samples for each estimated-$P(t|h)$ that we would have computed exactly in ADL. In DL(n) rank, core tests

and core histories are computed from the estimated submatrix of $\mathcal{Z}$ instead of the exact submatrix of $\mathcal{Z}$ as in ADL. The main difficulty in DL(n) is that because of the errors in the estimated predictions, rows and columns that were linearly dependent in ADL may become linearly independent in DL(n) and conversely rows and columns that were linearly independent in ADL may become linearly dependent in DL(n). This will introduce errors in computing rank, core tests and core histories in DL(n). Note that mistakes in one direction are acceptable, i.e., if the estimated predictions in DL(n) make two rows appear dependent while they would be independent in ADL, this may just make DL(n) run for more iterations than ADL but it would not introduce incorrect core tests or histories. A mistake in the other direction is more problematic, i.e., if the estimated predictions in DL make two rows independent when they are truly dependent, then DL(n) can overestimate rank and find incorrect core tests and histories. Our solution is to be conservative in computing rank of the estimated submatrix of $\mathcal{Z}$.

Golub and VanLoan, (1996, Section 5.5.8) consider the question of estimating rank of an unknown matrix, A, given a noisy estimate of that matrix, $\hat{A}$, which is exactly the case for DL. Their method involves determining a singular value cutoff σ_{cutoff}, then computing the singular values of $\hat{A}$ and defining the estimated rank as the number of singular values above the cutoff. They define $\sigma_{\text{cutoff}} = \epsilon\|\hat{A}\|_\infty$, where ϵ is the average error in the matrix entries. To find an estimate for ϵ in DL(n) we use Chebyshev's inequality (with a large certainty parameter) to compute a bound on the error in each estimated prediction based on n, the number of samples that went into the estimate (here we assume iid samples, i.e., we ignore the correlations introduced in the sampling algorithm). We use this bound to calculate the average error and from that the singular value cutoff which then gets used in calculating the estimated rank. Note that the smaller the n the larger the singular value cutoff will be and the more conservative our estimate of rank will be.

In each iteration (say k^{th}) of DL(n) we first compute an estimated rank, $\hat{\nu}_k$, and then select core histories and core tests by selecting the rows and columns that are most likely to be linearly independent. Given $\hat{\mathcal{Z}}_k$ at iteration k, we incrementally remove histories and tests until exactly $\hat{\nu}_k$ rows and columns remain. At each removal step, the history (test) to be removed is determined by removing each candidate history (test) and computing how well-conditioned the resulting matrix is. The history (test) whose removal yields the most well-conditioned matrix is then removed. The goal of this process is to find the $(\hat{\nu}_k \times \hat{\nu}_k)$ submatrix of $\hat{\mathcal{Z}}_k$ that is most well-conditioned[2]. Having the most well-conditioned full-rank submatrix serves two purposes. First, the rows and columns of this submatrix are most likely to turn out to be linearly independent with more sampling. Second, this submatrix is most suitable for solving linear equations via inversion, as needed in the learning algorithm presented next.

The discovery part of DL(n) terminates when the estimated rank remains the same for two iterations, and returns a set of core tests and histories that may be incorrect and so we label them $\hat{Q}_\mathcal{T}$ and $\hat{Q}_\mathcal{H}$. The learning part proceeds just as in ADL: n samples each are used to compute estimates $\hat{P}(\hat{Q}_\mathcal{T}|\hat{Q}_\mathcal{H})$, $\hat{P}(ao|\hat{Q}_\mathcal{H})$ and $\hat{P}(aot|\hat{Q}_\mathcal{H})$ and then calculate $\hat{m}_{ao} = \hat{P}^{-1}(\hat{Q}_\mathcal{T}|\hat{Q}_\mathcal{H})\hat{P}(ao|\hat{Q}_\mathcal{H})$, and $\hat{m}_{aot} = \hat{P}^{-1}(\hat{Q}_\mathcal{T}|\hat{Q}_\mathcal{H})\hat{P}(aot|\hat{Q}_\mathcal{H})$.

4. Empirical Results

As mentioned earlier, the only previous learning algorithm for PSRs was a myopic gradient-based algorithm (Singh et al., 2003) and it obtained mixed results on a set of dynamical systems taken from Cassandra's web-page (Cassandra, 1999) on POMDPs, indicating that these systems collectively offer a range of difficulty. Hence we chose to test our algorithms on these same simulated systems (listed in the first column of Table 1) — note, however, that we added a reset action to each system that when executed takes the system back to an initial configuration. Thus, the error rates for reset learning and myopic learning cannot be compared directly, but the rates for myopic learning are given for reference.

4.1. On Analytical Discovery

Given the uncertainty as to whether the analytical discovery algorithm's stopping condition can lead to premature termination (with correct but incomplete core tests and histories), we tested it out on the systems in the first column of Table 1. We could implement analytical discovery because we could compute $p(t|h)$ for any test t and history h given the POMDP models of the systems. As the column labeled analytical discovery states, for each system a full set of correct core tests and histories were found. Jaeger's (2003) algorithms for discovery of interpretable OOMs and IO-OOMs use the same stopping condition and his ex-

[2]When removing histories, we bias this computation by the probability of the history, because we wish to avoid rare histories that would not be sampled often. Also, when considering matrices that are approximately equally well-conditioned, we chose the matrix that makes best use of existing samples.

Table 1. Summary table of empirical results. See text for details.

System	*Core Tests*	*Num. Act.*	*Num. Obs.*	*Reset Learning*	*Myopic Learning*	*Analytical Discovery*	*ADL*	
							Num Tests	*Error*
Tiger	2	3	2	3.5E-7	4.3E-6	Yes	2	5.0E-5
Paint	2	4	2	2.7E-7	1E-5	Yes	2	9.4E-6
Float-reset	5	2	2	3.7E-8	1E-4	Yes	3	4.4E-4
Cheese Maze	11	4	7	3.8E-6	3.7E-4	Yes	11	5.3E-6
Network	7	4	2	3.2E-6	8.3E-4	Yes	3	2.8E-4
Bridge Repair	5	12	5	5.1E-6	3.4E-3	Yes	5	5.6E-4
Shuttle	7	3	5	2.2E-5	2.7E-2	Yes	7	5.5E-5
4x3 Maze	10	4	6	6.4E-5	6.6E-2	Yes	10	8.8E-4

perimental work also supports a positive conjecture for the stopping condition.

However, subsequent to this experiment we were able to prove that the discovery algorithm's stopping condition is in fact not guaranteed to always converge to a full set of core tests and histories. Despite our negative result using a carefully constructed counterexample (sketched briefly in the appendix), the stopping condition seems to work as desired in practice.

4.2. On Reset Learning

In this section, we tested the learning part of DL assuming perfect discovery (via analytical discovery as in the previous section). The goal here was to see if we could get significantly lower error than obtained by Singh et al. using their myopic learning algorithm (which also assumed knowledge of the full set of core tests). We ran the learning part of DL(n) for increasing values of n and measured error for the resulting model parameters. The error function we used was the one used by Singh et al; the average one-step prediction error per time step on a long test run with uniformly random actions. The error for a run is $\frac{1}{L}\sum_{t=1}^{L}\sum_{o\in\mathcal{O}}[P(o|h_t) - \hat{P}(o|h_t)]^2$, where $P(o|h_t)$ is the true probability of observation o in history h_t and $\hat{P}(o|h_t)$ is the corresponding estimated probability that uses our estimated model parameters to maintain the prediction vectors (as in Equation 2), and to compute the one-step predictions (as in the denominator of Equation 2). For this experiment, we used $L = 10,000$. Column 5 in Table 1 (labeled Reset Learning) shows that for each system we were able to get significantly lower error than myopic learning (shown in column 6).

4.3. Discovery and Learning

We also measured the performance of our algorithm for simultaneous discovery and learning, $DL(n = 1,000)$, in two ways: by the number of core tests found, and by testing the learned model parameters using the same error function ($L = 100,000$) as in the previous section. The last two columns show the number of core tests found and the error of the approximate PSR-model learned by DL(1000). In all cases the error was low and in all systems except Network and Float-reset all the core tests were correctly found. In Network, only 3 out of 7 core tests were found and in Float-reset only 3 out of 5 core tests were found, but regardless the error in the model was low in both cases.

We also explored the dependence of the performance of DL(n) on the number of samples n. We ran DL(n) for various values of n and computed the number of core tests found as a function of n and the test-error as a function of n. Figure 2 show the results for all the problems in Table 1; in all cases error dropped significantly with a few hundred samples per history-test pair. In all cases except Network and Float-reset, the core tests were discovered with a few hundred samples. As above, in Network and Float-reset the error becomes small without the discovery of all the core tests. These final results show that DL can discover and learn in these small systems fairly rapidly.

5. Conclusion

Replacing the state-test prediction matrix U used in the original PSR paper with the history-prediction matrix $\mathcal{Z}$ opens up new possibilities for learning and discovery algorithms. In this paper we focused on the class of controlled dynamical systems with reset and presented a discovery algorithm (a first for PSRs on a general class of systems) as well as a new learning algorithm. Our empirical results on the learning subproblem show that our algorithm was able to find good model parameters on the dynamical systems where the previous myopic algorithm had difficulty. Our empirical results on our discovery and learning algorithm (DL(n)), showed that discovery and learning can be fairly rapid. As future work we intend to continue em-

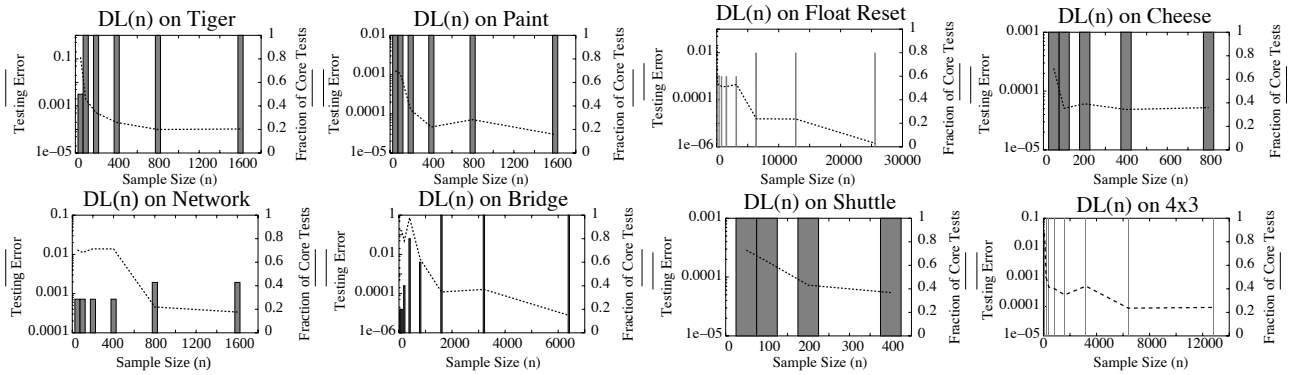

Figure 2. Learning curves for simultaneous discovery and learning in DL(n). The horizontal axis in each plot is the number of samples n. At each iteration of the algorithm the number of samples was doubled (we started with an initial sample size of $n = 50$). In each plot, the fraction of core tests found is plotted as bars against the right vertical axis, and the test error is plotted as a dashed line against the left vertical axis. See text for additional details.

pirical investigation of our algorithms on larger systems as well as to develop new algorithms that do not require a reset.

6. Appendix

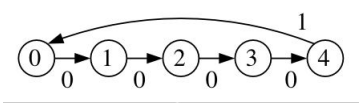

Figure 3. A state-based representation of the counterexample to the stopping condition in the analytical discover algorithm. There is a single action (hence it is also an HMM), and the labels on the arcs are observations.

The deterministic uncontrolled system in Figure 3 always produces observation 0 except in the transition from state 4 to state 0. The null history, ϕ, starts the system in state 0. Consider $\mathcal{Z}_1$ ($\mathcal{Z}_2$) the submatrix of $\mathcal{Z}$ for this system that contains all rows for histories up to length 1 (2) and all columns for tests up to length 1 (2). Each column of both $\mathcal{Z}_1$ and $\mathcal{Z}_2$ will either be all 0's or all 1's. Therefore the rank of both $\mathcal{Z}_1$ and $\mathcal{Z}_2$ is 1 and so analytical discovery will stop after iteration 2 and only find one core test and one core history. In this example, there are 5 core histories and tests. (Jaeger's 2003 OOM discovery algorithm is subject to a similar difficulty with this problem.) A detailed version of this proof is available in Singh et al. (2004).

References

Cassandra, A. (1999). Tony's pomdp page. http://www.cs.brown.edu/research/ai/pomdp/index.html.

Golub, G., & VanLoan, C. (1996). *Matrix computations*. The Johns Hopkins University Press. 3rd edition.

Jaeger, H. (2000). Observable operator processes and conditioned continuation representations. *Neural Computation*, *12*, 1371–1398.

Jaeger, H. (2003). Discrete-time, discrete-valued observable operator models: a tutorial.

Littman, M. L. (1996). *Algorithms for sequential decision making* (Technical Report CS-96-09). Ph.D thesis, Department of Computer Science, Brown University.

Littman, M. L., Sutton, R. S., & Singh, S. (2001). Predictive representations of state. *Advances In Neural Information Processing Systems 14*.

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, *28*, 47–65.

Rivest, R. L., & Schapire, R. E. (1994). Diversity-based inference of finite automata. *Journal of the ACM*, *41*, 555–589.

Shatkay, H., & Kaelbling, L. P. (1997). Learning topological maps with weak local odometric information. *Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)* (pp. 920–929).

Singh, S., James, M., & Rudary, M. (2004). Predictive state representation: A new theory for modeling dynamical systems. Submitted.

Singh, S., Littman, M. L., Jong, N. K., Pardoe, D., & Stone, P. (2003). Learning predictive state representations. *The Twentieth International Conference on Machine Learning (ICML-2003)*.