Learning Predictive Representations from a History

Eric Wiewiora WIEWIORA@CS.UCSD.EDU
Computer Science and Engineering,
University of California, San Diego

Abstract

Predictive State Representations (PSRs) have shown a great deal of promise as an alternative to Markov models. However, learning a PSR from a single stream of data generated from an environment remains a challenge. In this work, we present a formalism of PSRs and the domains they model. This formalization suggests an algorithm for learning PSRs that will (almost surely) converge to a globally optimal model given sufficient training data.

1. Introduction

Modeling systems that generate sequential observations is a fundamental problem in many fields. The standard approach is to assume that the system can be represented as a Markov model. These models assume that the history of all previous observations is condensed into a sufficient statistic, known as the Markov state. If the state of the process is known, then the probability of any future sequence of observations can be calculated.

While Markov models have been successful in many applications, they suffer many shortcomings. Markov models where the state is observable tend to scale poorly with the complexity of the system to be modeled. Markov models with a latent state tend to be difficult to learn from data.

Predictive state representations (PSRs), and closely related Observable Operator Models (OOMs), address many of these issues (Littman et al., 2001; Jaeger, 2000b). Unlike Markov models with observable state, PSRs can represent a wide class of stochastic processes succinctly. Unlike latent state Markov models, PSRs use only information that is present in the observables generated from the stochastic process.

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

There has been much previous work on learning a PSR from data. Singh, *et al.* provide an algorithm that updates the parameters of a PSR online, but do not address the issue of adapting the model's representational power (2003). James & Singh give an offline algorithm that learns an arbitrarily complex PSR, but it requires the system to have a "reset", which reverts the system to some previous state (2004).

In this work, we provide a formalism of PSRs and the policies that interact with them. We also provide a theoretical analysis of the data generated from systems that can be modeled as a PSR. From this analysis, we develop an algorithm for learning a PSR from a single stream of data. Given sufficient data, the learned model will be a near optimal representation.

1.1. Stochastic Processes

We define a stochastic process (also referred to as a dynamical system) as a source of sequential observations that occur on fixed time steps. We will assume that observations are drawn from a finite set $\mathcal{O}$. Some processes require an action to be provided by an outside source. This action is from a finite set $\mathcal{A}$. When a stochastic process requires an action, we call it a controlled process; otherwise it is an autonomous process. In this paper we focus on controlled processes, but all analysis can be carried over to the autonomous case.

The distribution on the next observation generated by a stochastic process is a function of all observations the system has generated and all actions provided to the system. The entire string of previous observations and actions is known as a history.

A policy is a mapping from a history to a distribution over the current action that will be supplied to the system. When a policy is specified along with a controlled process, and the entire system can be modeled as an autonomous process. This coupled system generates observations from a finite set $\mathcal{U} = \{\mathcal{A} \times \mathcal{O}\}$.

1.2. Events and Tests

Predictive state representations use explicit predictions about future actions and observations to model the state of the system. We specify the target of these predictions using *events* and *tests*.

An event is a set of action-observation sequences, where each sequence contains l observations and l actions. An event is said to have occurred when the last l action-observation pairs in the history match one of the sequences in the event.

A test consists of two events. The *conditioning* event for test g, called g_c, is assumed to occur over the next l time steps. The *conditioned* event g_u may or may not occur. The conditioned event must a subset of the conditioning event. At any point in the history of a stochastic process, we define the probability of a test g succeeding as the probability that the conditioned event occurs in the next l observations, divided by the probability that the conditioning event occurs:

$$Pr(g|h) \equiv \frac{Pr(hg_u|h)}{Pr(hg_c|h)} = Pr(hg_u|hg_c).$$

We give special attention to a class of tests called s-tests. These tests have a single sequence of actions and observations as a conditioned event, and a conditioning event consisting of the same sequence of actions, but with any sequence of observations. Thus, an s-test g of length l can be written as

$$\begin{aligned} g_u &= a_1 o_1 a_2 o_2 \ldots a_l o_l; \\ g_c &= \{a_1 \mathcal{O} a_2 \mathcal{O} \ldots a_l \mathcal{O}\}. \end{aligned}$$

Because the actions that are in g_u are in all sequences in g_c, the influence of the policy has been factored out of the probability that an s-test will succeed. We define s-tests for autonomous processes analogously by assuming there is an implicit action between every observation.

We include one special case in the class of s-tests. The ϵ-test is the only test consisting of events of length zero. This test succeeds with probability 1. Colloquially, the ϵ-test represents the event that anything happens in the future.

2. Linear PSRs

A predictive state representation is a method for modeling a stochastic process using a set of tests called the *core tests*. We say that a set of k tests $q_1, \ldots, q_k$ are core tests of a PSR if given any history h, the success probability of any future s-test g is a function of the probability that each of the core tests will succeed:

$$Pr(g|h) = f_g\Big(Pr(q_1|h), Pr(q_2|h), \ldots, Pr(q_k|h)\Big).$$

We call the vector of probabilities that the core tests succeed, given the history, as $\mathbf{q}(h)$. This vector is used much like the belief state vector of a hidden Markov model.

Linear PSRs assume that all core tests are s-tests and f_g is a linear function of $\mathbf{q}(h)$:

$$Pr(g|h) = \mathbf{q}(h)^\top m_g.$$

We say a stochastic process has rank k if it can be modeled as a linear PSR with k core s-tests.

In order to use this model, we need accurate core test probabilities. We do this inductively. Given a history h, followed by action a and observation o, we update the probabilities that the core tests using two sets of vectors.

First, we need the vector m_{ao} such that

$$Pr(hao|ha\mathcal{O})) = \mathbf{q}(h)^\top m_{ao},$$

for each possible action a and observation o. These are called the one step tests.

Also, for each core test q_i consisting of conditioning event c_i and conditioned event u_i, we find the vector m_{aoi} such that

$$Pr(haou_i|ha\mathcal{O}c_i) = \mathbf{q}^\top m_{aoi}.$$

We call these the core extension tests.

The future core test probabilities are updated using m_{ao} and m_{aoi}. When a new action a is taken and observation o received, the new success probability of core test q_i is given by:

$$Pr(q_i|hao) = \frac{\mathbf{q}(h)^\top m_{aoi}}{\mathbf{q}(h)^\top m_{ao}}.$$

For convenience, we arrange the core extension test vectors for ao into a matrix M_{ao}, where the ith column in this matrix corresponds to the extension test for core test q_i.

Any s-test's success probability can be calculated using M_{ao} and m_{ao}. For s-test g, where $g_u = a_1 o_1 a_2 o_2 \ldots a_n o_n$, and a core test probability vector $\mathbf{q}(h)$, we can calculate the probability g succeeds as

$$Pr(g|h) = \mathbf{q}(h)^\top M_{a_1 o_1} \times \ldots \times M_{a_{n-1} o_{n-1}} m_{a_n o_n} \quad (1)$$

(Littman et al., 2001).

2.1. Regular Form PSRs

Any rank k stochastic process can be modeled by many rank k PSRs. Any set of k s-tests where each test cannot be predicted by the others will constitute a core test set (Singh et al., 2004). We would like a guideline for choosing a core tests that leads to as simple a representation as possible.

Definition 1 *A PSR is in regular form if the set of core tests is minimal and each core test is either the ϵ-test, or an extension of a core test.*

An algorithm for producing a (nearly) regular form PSR from a Markov model has been provided in (Littman et al., 2001). We extend these results to all systems that can be modeled as PSRs.

Theorem 1 *Any k test PSR can be converted to a regular form PSR with no more than k tests.*

Proof: Assume PSR $\mathbf{P} = (q_1 \ldots q_k, \{m_{ao}\}, \{M_{ao}\})$ is not in regular form. We show how to incrementally convert it to a regular form PSR by replacing core tests that do not meet the regular form defiition.

Consider when the ϵ-test is not a core test in $\mathbf{P}$. By our definition of a PSR, there is some m_ϵ such that $\mathbf{q}(h)^\top m_\epsilon = 1$ for all histories h. We find some test q_i where the ith entry in m_ϵ is nonzero. Such a test must exist since the probability of the ϵ test is nonzero. We can calculate the probability of q_i using the other core tests and the ϵ-test:

$$\begin{aligned} 1 &= \sum_j Pr(q_j|h)m_\epsilon[j] \\ Pr(q_i|h)m_\epsilon[i] &= 1 - \sum_{j\neq i} Pr(q_j|h)m_\epsilon[j] \\ Pr(q_i|h) &= Pr(\epsilon|h)\frac{1}{m_\epsilon[i]} \\ &\quad + \sum_{j\neq i} Pr(q_j|h)\frac{-m_\epsilon[j]}{m_\epsilon[i]}. \end{aligned}$$

Because we can calculate $Pr(q_i|h)$ for any h, this core test can be replaced with the ϵ-test.

Now consider the case where there is some core test q_i which is not an extension of any other core test. Also, assume there is some extension of a shorter core test q_j, which we call g_{aoj}, that is not a core test, and q_i has a nonzero entry in m_{aoj}. We use the same argument made for the ϵ-test to show that we can replace q_i with g_{aoj}.

Suppose, however that we cannot find a core test extension of any shorter q_j, where q_i influences the success probability. If q_i also does not influence the success probabilities of extensions no shorter than q_i, then this test is redundant. We show this by producing an m_i that predicts the probability of q_i without using the value of q_i. Using equation 1, we have $m_i = M_{a_1^i o_1^i} \times \ldots \times M_{a_l^i o_l^i} m_\epsilon$, where we have multiplied core extension matrices corresponding to the l actions and observations in the q_i event. Because q_i does not affect the probabilities of extensions of any other test, all of the M matrices have zero entries in the off-diagonal rows corresponding to i. If the ϵ event is present, m_ϵ must also have a zero entry in the ith position. Thus, the product of these matrices and m_ϵ will have a zero in the ith position.

Finally, we must consider the case where q_i is needed to predict some extension of a longer core test. In this case, we use arguments similar to those above to show that either there is a long core event that can be replaced with a shorter event, or all core events longer than q_i are redundant. $\square$

A regular form PSR has fewer parameters than a general PSR. Because we have included the ϵ-test, all one step tests are also core extension tests. Also, many core extension tests are also core tests. A general k test PSR has $|\mathcal{U}| * k$ entries in one step test vectors, and $|\mathcal{U}| * k^2$ entries in core test extension matrices. A regular form PSR has only $|\mathcal{U}| * (k^2 - k)$ unique entries.

The regular form PSR proof also guarantees that we will not have to search indefinitely for a set of core tests.

Corollary 1 *Every rank k stochastic process has a predictive state representation where each core test has length less than k.*

Proof: Because every core test is an extension of some other core test, the maximal length of any test is the maximum number of times a test can be extended. This is $(k-1)$ times. $\square$

3. Linear Policies

We treat policies much like we have treated the processes that policies control. Like a controlled process, the next action a policy outputs can be an arbitrary function of the entire history of previous observations and actions. This is an infinite set of possible policies. In order to analyze the long-term behavior of a policy, we need a measure of its complexity similar to the rank of a PSR.

We will do this by introducing the a-test. An a-test is equivalent to an s-test, but with the role of actions and observations reversed:

$$\begin{aligned} p_u &= o_1 a_1 o_2 a_2 \ldots o_k a_k; \\ p_c &= \{o_1 \mathcal{A} o_2 \mathcal{A} \ldots o_k \mathcal{A}\}. \end{aligned}$$

The ϵ-test is also included in the set of a-tests.

A policy of rank has rank f if the probability that any sequence of actions, conditioned on the sequence of observations, is a linear combination of the probability of a set of f a-tests. After any history h, we call the probability vector of the a-tests $\mathbf{b}(h)$.

Like a PSR, we will need to define functions to maintain the probabilities of the success of the a-tests. The one-step tests calculate the probability of the test $Pr(oa|o)$ for all a and o. This is a linear function parameterized with a weight vector r_{oa}. Also, we need to calculate the vectors for the a-test extensions. We arrange the weight vectors for the core test extensions of oa into matrix R_{oa}. Like PSRs, a regular policy can be expressed in a regular form.

In order to gain intuition on proper policies, consider the class of reactive policies. A reactive policy selects the next action only based on the previous observation. This policy has rank 1. The reason for this is that the probability that any action is taken is completely determined by the one-step tests. The probability these tests succeed remains constant throughout the interaction of the policy with the environment. Because of this, $\mathbf{b}(h)$ contains only one entry corresponding to the ϵ-test. In general, a kth order reactive policy (one that selects the next action according to the previous k observation-action pairs) will have rank $|\mathcal{U}|^k$. Regular policies are also capable of representing policies encoded as options (Sutton et al., 1999), finite state automata (Hansen, 1997), and many other forms.

Now we examine the behavior of a linear PSR coupled with a proper policy. It is easy to show that an n state Markov decision process that is controlled by a m state controller will become a $m*n$ state Markov chain. We show a similar result for PSRs.

Theorem 2 *A rank k process controlled by a rank f regular policy is an autonomous process with rank no greater than $k*f$.*

Proof: We treat the combination of a policy and a controlled process as a coupled system that outputs an action-observation pair every time step. A similar analysis is possible if one considers the system outputting observation-action pairs, or a system that periodically outputs a single observation or action.

The coupled system is parameterized by the initial probabilities of the core s-tests of the decision process $\mathbf{q}(h)$. We also know the initial probabilities the core a-tests, $\mathbf{b}(h-)$. We use the term $(h-)$ to reflect the fact that the a-tests in $\mathbf{b}$ are defined on histories ending in an action, not an observation. Also, because the first output of the coupled system will be an action, we specify special tests r_{-a} and R_{-a} for all actions a. These tests respectively determine the probability of the first action, and the value of $\mathbf{b}(ha)$ multiplied by the probability that a is the first action.

The probability that some ao occurs immediately is

$$\begin{aligned} Pr(ao|\epsilon) &= Pr(a|h)Pr(ao|h,a) \\ &= \mathbf{q}(h)^\top m_{ao} \cdot \mathbf{b}(h-)^\top r_{-a}. \end{aligned}$$

We calculate the probability of longer sequences inductively. Assume that any sequence s_k of length k can be written as a product of $\mathbf{q}(h)^\top m_{s_k}$ and $\mathbf{b}(h-)^\top r_{s_k}$.

We now show how to calculate probability of a length $k+1$ sequence s_{k+1}:

$$\begin{aligned} Pr(s_k a_{k+1} o_{k+1}|h) &= Pr(a_1 o_1 \ldots a_k o_k|h) \\ &\quad \cdot Pr(a_{k+1}|h a_1 o_1 \ldots a_k o_k) \\ &\quad \cdot Pr(o_{k+1}|h a_1 o_1 \ldots a_k o_k a_{k+1}) \\ &= \mathbf{q}(h)^\top m_{s_k} \cdot \mathbf{b}(h-)^\top r_{s_k} \\ &\quad \cdot \frac{\mathbf{b}(h-)^\top R_{s_k}}{\mathbf{b}(h-)^\top r_{s_k}} r_{o_k a_{k+1}} \\ &\quad \cdot \frac{\mathbf{q}(h)^\top M_{s_k}}{\mathbf{q}(h)^\top m_{s_k}} m_{a_{k+1} o_{k+1}} \\ &= \mathbf{m}(h)^\top m_{s_{k+1}} \cdot \mathbf{b}(h-)^\top r_{s_{k+1}}. \end{aligned}$$

Here R_{s_k} and M_{s_k} are Matrices calculated according to equation 1. Thus, the probability any sequence of action-observation pairs occurs can be described as the product of two dot products. We rewrite this as the single dot product:

$$\begin{aligned} \mathbf{q}(h)^\top m_s \cdot \mathbf{b}(h-)^\top r_s &= m_s^\top \Big(\mathbf{q}(h) \cdot \mathbf{b}(h-)^\top\Big) r_s \\ &= m_s^\top C(h) r_s \\ &= \sum_{i,j} C(h)[i,j] m_s[i] r_s[j] \\ &= \mathbf{c}(h)^\top \omega_s, \end{aligned}$$

where $\mathbf{c}(h)$ is the vectorized outer product of $\mathbf{q}(h)$ and $\mathbf{b}(h)$, and ω_s is the vectorized outer product of m_s and r_s. Because any sequence probability is a dot product of a $(k*f)$-dimensional vector, we will only be able to find up to $k*f$ linearly independent sequence probabilities. □

It's important to note that the core tests of the coupled system are not a-tests or s-tests, but rather distinct sequences. In fact, the core tests of the coupled system need not have any relation to s-tests or a-tests of the component systems.

4. A Limit Property

Given an autonomous process, we would like to know its long-term behavior. In OOM theory, it is generally assumed that the process is stationary and ergodic (Kretzschmar, 2003). We take a more careful look at what conditions are sufficient to build learning algorithms, and when these conditions are met.

We analyze long-term behavior in terms of the *empirical frequencies* of sequences a process generates. Given history h_{T+j} we define the empirical frequency of sequence g of length j as

$$\bar{g} = \frac{1}{T}\sum_{i=1}^{T} \mathrm{I}(h_{i+j} = h_i g),$$

where the function I() is the indicator function. Given a sufficiently long history, we hope that the empirical frequency of a sequence converges to its average success probability:

$$\hat{g} = \frac{1}{T}\sum_{i=1}^{T} Pr(g|h_i) \text{ as } T \to \infty.$$

Theorem 3 *As T becomes large, for all q_i that constitute a minimal set of core tests for an autonomous process, $\hat{q}_i$ converges almost surely.*

Proof:(sketch) At any point in the evolution of the stochastic process, the system's future is characterized by the (unkown) vector $\mathbf{q}(h_t)$.

We track the expectation of $\mathbf{q}(h_{t+1})$ using the matrix

$$M = \sum_{u \in \mathcal{U}} M_u.$$

This matrix calculates the expected value of the core test vector on the next time step. The M matrix shares many properties of a transition matrix for a Markov chain. It can be shown that M has spectral value one, with at least one eigenvector for eigenvalue one. All Jordan blocks for eigenvalue one must have rank one. If the spectral value is greater than one, then either some entry in $\mathbf{q}(h)$ will not be a valid probability, or all valid $\mathbf{q}$ vectors are orthogonal to all eigenvectors with eigenvalue greater than one. This additional linear constraint on the core test probabilities contradicts the assumption that $\mathbf{q}$ was minimal. If the spectral value of M is less than 1, the ϵ event will be expected to have probability less than 1 in the future. More properties that the M matrix must display can be derived from results in the OOM literature (Jaeger, 2000a).

Define M^* as $\sum_{i=1}^{N} \frac{1}{N} M^i$, as $N \to \infty$. This matrix determines the expected value of $\hat{\mathbf{q}}$, given the current value of $\mathbf{q}(h)$. Because of the spectral properties of M we have outlined above, we know that this limiting matrix exists. Since M^* converges, it must satisfy $M^* = MM^*$.

Note that for any possible $\mathbf{q}(h_i)$,

$$\mathbf{q}(h_i)^\top M^* = \mathbf{q}(h_i)^\top M M^* = E[\mathbf{q}(h_{i+1})^\top M^*].$$

Thus, $\mathbf{q}(\cdot)^\top M^*$ is a martingale (Grimmett & Stirzaker, 1982). Also, note that $\mathbf{q}(\cdot)^\top M^*$ has bounded variance, due to the fact that the entries in $\mathbf{q}$ always range between 0 an 1. Using these two results, and standard martingale theory, we can show that the random variable $\mathbf{q}(\cdot)^\top M^* = E[\hat{\mathbf{q}}]$ converges to a fixed point almost surely (Grimmett & Stirzaker, 1982). □

Corollary 2 *If the average success probabilities of all core tests converge, then the average success probabilities of all sequences converge.*

Proof: At each time step, the probability that a sequence g will occur over the next observations is determined by the probabilities of the core tests at that time:

$$Pr(g|h) = \mathbf{q}(h)^\top m_g.$$

The long term average success probability can likewise be written as a linear function of the average success probabilities of the core events:

$$\begin{aligned} \hat{g} &\approx \frac{1}{T}\sum_{i=1}^{T} \mathbf{q}(h_i)^\top m_g \\ &= \Big(\frac{1}{T}\sum_{i=1}^{T} \mathbf{q}(h_i)^\top\Big) m_g \\ &\approx \hat{\mathbf{q}}^\top m_g. \end{aligned}$$

As T becomes large, both $\hat{\mathbf{q}}$ and $\hat{g}$ will be arbitrarily close to some fixed point with probability 1. □

4.1. Empirical Test Frequencies

Given empirical frequencies of sequences, we can derive estimates of the average success probabilities of tests. We adopt the estimator used in James and Singh (2004). For some test $t = Pr(t_u|h, t_c)$ we calculate $\bar{t}$ as:

$$\bar{t} = \frac{\bar{t}_u}{\bar{t}_c},$$

where we calculate $\bar{t}_i$ as the sum of the empirical frequencies of sequences that compose event t_i.

In order for us to use this estimate, the process should have nonzero probability of generating t_c for any history. Thus, in order to learn s-tests, we should use a policy that chooses all actions with nonzero probability, given any history.

4.2. Context Tests

Our estimates of empirical test probabilities suggest a method for learning the parameters of a PSR. We define a set of context tests $\varphi_1, \ldots, \varphi_f$. Given a context test φ and some test g, we find the empirical frequency of test φg, where

$$Pr(\varphi g|h) = Pr(h\{\varphi_u g_u\}|h\{\varphi_c g_c\}).$$

We call these context extension tests.

When context tests are s-tests, the following nice property holds. Given the core tests $\mathbf{q}$, a context test φ and an arbitrary test g, we have

$$\begin{aligned} \hat{\varphi g} &= \hat{\mathbf{q}}^\top m_{\varphi g} \\ &= \hat{\mathbf{q}}^\top M_\varphi m_g \\ &= \hat{\varphi \mathbf{q}}^\top m_g, \end{aligned}$$

where M_φ is a matrix built according to equation 1. Thus, if we can find context tests that produce k linearly independent $\hat{\varphi \mathbf{q}}$, we can derive an estimate of m_g. The restriction that context tests are s-tests can be loosened to include any test. In general, when we have used a finite rank policy to collect data, $\hat{\varphi g} = c_\varphi * (\hat{\varphi' \mathbf{q}}^\top m_g)$, where φ' is a linear combination of s tests and c_φ is a constant that depends on the policy that is used.

The context test has several precursors in the literature. In OOM theory, the role of the context test are assumed by indicative events (Jaeger, 2000b). A set of indicative events partition all length m sequences into k sets. In both Rosencrantz et al. (2003) and James and Singh (2004), context sequences consist of a single history, starting from time zero.

5. Learning and Discovery

The *discovery* problem is to find a set of core tests, and the *learning* problem is to find appropriate $\{m_{ao}\}$, and $\{M_{ao}\}$ parameters for these tests (Littman et al., 2001). Given a sufficient number of empirical test frequencies, we can solve these problems.

We do this by accumulating empirical test frequencies into a matrix $\mathbf{F}$. The $[i, j]$th entry of $\mathbf{F}$ corresponds to the value of $\bar{\varphi_i g_j}$, for arbitrary context test φ_i and s-test g_j.

If we impose an upper bound on the complexity of the system to be modeled, we can limit the size of $\mathbf{F}$ that will be required to find a set of core tests.

Theorem 4 *If a stochastic process has rank k, the $\mathbf{F}$ matrix consisting of context tests of all sequences of length $(k-1)$ and less will be sufficient to find core tests that model the process's future observation probabilities.*

Proof:(sketch) The argument is similar in detail to the proof of theorem 1. If we have not found k linearly independent rows in the $\mathbf{F}$ matrix consisting of context tests of length $m < k$ and less, then there must be at least one column in the $\mathbf{F}$ matrix where length $m+1$ context tests are also included. If this is not the case, then all larger $\mathbf{F}$ matrices cannot contain additional independent rows. This last condition may arise if the process has transient characteristics that will not affect probabilities after some point in time. $\square$

Using theorems 1 and 5, we see that in order to learn k core tests, we need an $\mathbf{F}$ matrix with no more than $O(|\mathcal{U}|^{k-1})$ rows and columns. This is a worst-case bound; it is very likely that far fewer than $|\mathcal{U}|^{k-1}$ rows will be required to find k good core tests.

We now have enough theory to specify an algorithm that solves the discovery and learning problems. This algorithm is similar to many algorithms in the literature (Rosencrantz et al., 2003; James & Singh, 2004; Jaeger, 2000b). We diverge from existing algorithms by explicitly learning a regular form PSR and explicitly minimizing the average error of context extension tests.

Greedy Discovery/Learning Algorithm

1. Build an $\mathbf{F}$ matrix from a sequence of observations generated from a stochastic process; Choose a value for error threshold parameter θ.
2. Assign the ϵ-test as the first core test:
$$\mathbf{Q}[\cdot, 1] = \mathbf{F}[\cdot, \epsilon]$$
3. For all core test extensions i in $\mathbf{F}$:
    (a) solve $\mathbf{F}[\cdot, i] \approx \mathbf{Q} m_i$ with regression
    (b) $\mathbf{E}[\cdot, i] = \mathbf{F}[\cdot, i] - \mathbf{Q}\, m_i$
4. For $k = 2, 3, \ldots$, while $||\mathbf{E}|| > \theta$:
    (a) Search all columns in $\mathbf{E}$ that correspond to core test extensions; choose column j with

maximum norm:

$$\mathbf{Q}[\cdot, k] = \mathbf{F}[\cdot, j]$$

(b) Recompute m_i and $\mathbf{E}$ using the equations of step 4.

Below we discuss some details of the algorithm.

- The algorithm requires the specification of a parameter θ, which effectively controls the number of core tests the algorithm will discover. We can control for the number of core tests explicitly if we have some idea how many to expect. The SVD algorithm has been used in prior literature for this purpose (Rosencrantz et al., 2003).
- When we train using a short history, the raw $\mathbf{F}$ matrix is very noisy, with some entries representing a handful of occurrences of the conditioning event. In order to compensate for this, we use context tests that do not condition on the actions of the sequence. This effectively weights each row by the probability the context sequence occurs in the data.
- Our choice of context tests will bias the estimates towards reducing average error after a particular context test succeeds. This provides a potential selection criterion for context tests.
- In our experiments, we use least squared error regression to estimate the m_i parameters. Optimizing for other error measures is possible.

6. Examples

The following examples intend to demonstrate the applicability of the above algorithm to PSR problems. The experiments are ongoing, and the present results should be treated as a proof-of-concept.

We test our algorithm on several learning tasks taking place in the float-reset domain (Littman et al., 2001). This domain can be represented as a 5 state Markov model with two actions and two observations. When the reset action is taken, the system transitions to state 0. If the system is in state zero when the reset action is taken, an observation of "1" is emitted; otherwise the system outputs a "0". The float causes a transition to a neighboring state with equal probability. Thus, a float in state 3 causes a transition to state 1 or 4. Transitions to states -1 and 5 instead transition to 0 and 4, respectively. The "0" observation is always observed when the float action is taken.

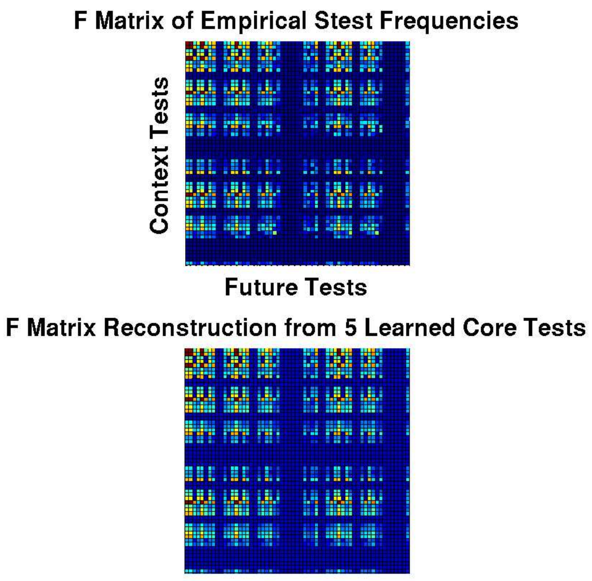

Figure 1. An frequency matrix calculated from data on the float-reset problem. Below is a reconstruction of the matrix using the columns corresponding to the core tests.

Although this domain is conceptually simple, it has proven hard to learn in practice. The minimal set of core tests for this domain each have a different length. For instance, a set of core tests is $u_i = \{\epsilon, "r0", "f0r0", "f0f0r0", "f0f0f0r0"\}$. In order to learn the m_g vectors for this set of core tests, the system must spend a significant amount of time in state 4. It requires a fairly long sequence of float actions before the system will be in state 4 with high probability.

James and Singh (2004) present an experiment on learning and discovery of a PSR model for the float-reset domain. They found that using a uniformly random policy, they could not gather sufficient data to confirm that the system has rank 5. This is because the influence of the longer core tests on transition probabilities is very subtle. After 30,000 examples, the algorithm did not have sufficient evidence to warrant a fifth core test.

We present a series of experiments on the float-reset problem with various instantiations of our greedy algorithm. In all experiments, we learn from a single stream of data generate from the process.

We enumerate every s-test of length 5 or less. These s-tests are used as both context tests and as potential core tests. The $\mathbf{F}$ matrix constructed from these tests is very large, with many rows and columns that correspond to conditioned events that have rarely or never been seen in the training data. Thus, weighting

F with **W** is necessary for small sample sizes.

In our first experiment, we sample 25,000 action-observation pairs from the float-reset domain being controlled by a policy that chooses between float and reset uniformly at random. On 5 runs of the greedy discovery and learning algorithm, the shortest 4 core tests are always chosen by the discovery algorithm out of the first five core tests. On 2 runs, the first five core tests that were chosen were a core set. On 2 other runs, the fifth core test was chosen as the sixth test.

The second experiment attempts to discover the core tests more effectively by using an exploratory policy. This policy performs between 0 and 10 float operations before executing a reset action. The policy performs 0 float operations between resets with probability 1/3, and divides the probability of performing 1 to 10 floats evenly. The system performed comparably to the previous experiment. It learned the correct core tests 3 times out of five, and chose 5 core test out of 6 on another trial.

In order to show that we have successfully learned proper parameters as well as discovered the core tests, we ran an experiment to construct an **F** matrix from the core tests and core test extension vectors. The original **F** matrix is shown on the top of figure 1. After running the greedy learning and discovery algorithm, we applied the learned PSR to predicting future rows of **F**, given $\varphi_{\bar{j}}\mathbf{q}$. This is analogous to the PSR starting with a prior probability on $\mathbf{q}(h)$. Is is obvious from the figure that the experiment was successful. The entire **F** matrix was reconstructed with little error.

7. Conclusion

In this paper, we have presented a theoretical framework for analyzing Predictive State Representations and algorithms for learning them. This analysis has been put to use in developing a new learning algorithm, as well as providing a theoretical justification for some design choices made in previous work.

An important contribution of this work is the introduction of regular policies. This formalism has the promise of unifying the analysis of many classes of policies. For example, the PSR framework may be provide the proper analytic tools for justifying the benefit of temporal abstraction through options.

Acknowledgements

I would like to thank Satinder Singh for many valuable comments and suggestions. The author was funded under a grant from the NSF IGERT program.

References

Grimmett, G., & Stirzaker, D. (1982). *Probability and random processes.* Claredon Press.

Hansen, E. (1997). An improved policy iteration algorithm for partially observable MDPs. *Advances in Neural Information Processing Systems.*

Jaeger, H. (2000a). *Discrete-time, discrete-valued observable operator models: a tutorial* (Technical Report). GMD - German National Research Center for Information Technology.

Jaeger, H. (2000b). Observable operator models for discrete stochastic time series. *Neural Computation, 12.*

James, M. R., & Singh, S. (2004). Learning and discovery of predictive state representations in dynamical systems with reset. *Twenty-first International Conference on Machine Learning.*

Kretzschmar, K. (2003). *Learning symbol sequences with observable operator models* (Technical Report). GMD - German National Research Center for Information Technology.

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

Rosencrantz, M., Gordon, G., & Thrun, S. (2003). Learning low dimensional predictive representations. *Twenty-first International Conference on Machine Learning.*

Singh, S., James, M., & Rudary, M. (2004). Predictive state representations: A new theory for modelling dynamical systems. *Conference on Uncertainty in Artificial Intelligence.*

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

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence, 112.*