Learning Predictive State Representations Using Non-Blind Policies

Michael Bowling BOWLING@CS.UALBERTA.CA
Peter McCracken PETERM@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8 Canada

Michael James MICHAEL.R.JAMES@GMAIL.COM
AI and Robotics Group, Technical Research Dept., Toyota Technical Center, Ann Arbor, Michigan, USA

James Neufeld NEUFELD@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8 Canada

Dana Wilkinson D3WILKIN@UWATERLOO.CA
School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada

Abstract

Predictive state representations (PSRs) are powerful models of non-Markovian decision processes that differ from traditional models (e.g., HMMs, POMDPs) by representing state using only observable quantities. Because of this, PSRs can be learned solely using data from interaction with the process. The majority of existing techniques, though, explicitly or implicitly require that this data be gathered using a blind policy, where actions are selected independently of preceding observations. This is a severe limitation for practical learning of PSRs. We present two methods for fixing this limitation in most of the existing PSR algorithms: one when the policy is known and one when it is not. We then present an efficient optimization for computing good exploration policies to be used when learning a PSR. The exploration policies, which are not blind, significantly lower the amount of data needed to build an accurate model, thus demonstrating the importance of non-blind policies.

1. Introduction

Predictive state representations (PSRs) are a method of representing the state of a controlled, discrete-time, stochastic dynamical system by maintaining predictions about the effect of future actions on observations (Littman et al., 2002). PSRs have several advantages over other methods of state representation, such as POMDPs and k-Markov models. It has been shown that PSRs are able to compactly model any system representable by a POMDP (Singh et al., 2004) or k-Markov model (Littman et al., 2002). Also, unlike POMDPs which use a postulated set of underlying, nominal states, PSRs are completely specified using observable quantities like actions and observations. This allows them to be learned solely using data from interaction with the system.

There have been numerous algorithms proposed for building PSR models from data (Singh et al., 2003; James & Singh, 2004; Rosencrantz et al., 2004; Wolfe et al., 2005; Wiewiora, 2005; McCracken & Bowling, 2006). Many of these algorithms specify a particular policy to be used when gathering their data. Those that do not are usually evaluated using the uniform-random policy. In either case the policy is blind, ignoring preceding observations when selecting actions. To date all of the policies used in PSR empirical evaluation have been blind. Generally speaking, though, non-blind policies are the norm not the special case. Selecting actions based on past observations is key for any policy seeking to maximize reward or achieve some goal. Even a smart exploration strategy will make decisions based on past observations. The ability to build accurate models from data gathered while purposefully exploring or acting to maximize reward is fundamental.

In this paper we examine the construction of PSR models from data gathered while following non-blind policies. In particular, we conclude that most existing algorithms fail to build a correct model when data is gathered with such a policy. We start with a brief introducion to PSRs in Section 2. In Section 3 we prove that the majority of existing

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006.

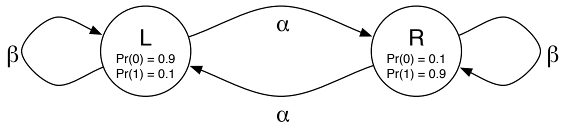

Figure 1. A dynamical system with two nominal states, action set $\mathcal{A} = \{\alpha, \beta\}$, and observation set $\mathcal{O} = \{0, 1\}$. Action α switches state, and action β preserves state. The initial state of the system is state L.

Table 1. Two policies for the system in Figure 1. Policy 1 is dependent on the previous observation, and Policy 2 is blind. '$*0$' and '$*1$' represent any history that ends in observation 0 and 1, respectively.

	Policy 1		Policy 2	
h	$\Pr(\alpha\|h)$	$\Pr(\beta\|h)$	$\Pr(\alpha\|h)$	$\Pr(\beta\|h)$
ϕ	0.5	0.5	0.5	0.5
$*0$	0.9	0.1	0.5	0.5
$*1$	0.1	0.9	0.5	0.5

algorithms are only correct if data is gathered with a blind policy. We then show in Section 4 how existing algorithms can be fixed to handle data from any policy, known or unknown. In Section 5 we present a novel method for finding smart exploration policies which are not blind. We then show results in Section 6 when learning with these policies, demonstrating the correctness and usefulness of our remedy.

2. Background

A controlled, discrete-time, finite dynamical system generates observations from a set $\mathcal{O}$ in response to actions from a set $\mathcal{A}$. The probability of a generated observation, though, can depend on the complete interaction to date. An example dynamical system is shown in Figure 1. This system has two nominal states $\{L, R\}$, actions $\mathcal{A} = \{\alpha, \beta\}$ and observations $\mathcal{O} = \{0, 1\}$. An interaction with a dynamical system can be represented as a stream of action-observation pairs, of the form $a_1 o_1 a_2 o_2 \ldots$, that begins at time step zero. In such a sequence, the first action taken is a_1, after which the observation o_1 is observed, followed by action a_2 and observation o_2, etc., until the end of the interaction. A sequence of action-observation pairs that has already occurred is referred to as a *history*. The special history, ϕ, is the zero-length history where no actions have been taken.

At its essence a system is a probability distribution over observations, conditioned on the history and next action. For example, the system in Figure 1 can be described as,

$$\Pr(0|ha) = \begin{cases} 0.9 & \text{if } s(ha) = 0 \bmod 2 \\ 0.1 & \text{if } s(ha) = 1 \bmod 2 \end{cases},$$

where $s(h)$ is the number of α, or switching, actions in h. POMDPs (or HMMs in the uncontrolled case), k-Markov models, and PSRs are simply compact models of this infinite set of probability distributions.

A *policy* is a probability distribution over actions, conditioned on history. Example policies are shown in Table 1. A policy can rely on previous observations, like Policy 1 in Table 1, or it can be blind to previous observations, like Policy 2. A policy is said to be *blind* if each policy action is conditionally independent of previous observations, given previous actions. It should be noted that blind policy actions are not necessarily independent of previous observations. If action a_2 depends on action a_1, and observation o_1 depends on a_1, then a_2 and o_1 may not be (statistically) independent.

A policy and a system together define a complete probability distribution on histories. The following convenient notation summarizes the contributions of the policy and the system for a particular history,

$$\pi(a_1 o_1 \ldots a_n o_n) \equiv \prod_{i=1}^{n} \Pr(a_i | a_1 o_1 \ldots o_{i-1})$$
$$p(a_1 o_1 \ldots a_n o_n) \equiv \prod_{i=1}^{n} \Pr(o_i | a_1 o_1 \ldots a_i).$$

Using this notation the probability of any history is $\Pr(h) = \pi(h)p(h)$. We will also use the notation,

$$\pi(t|h) \equiv \frac{\pi(ht)}{\pi(h)} = \prod_{i=1}^{n} \Pr(a_i | h a_1 o_1 \ldots o_{i-1})$$
$$p(t|h) \equiv \frac{p(ht)}{p(h)} = \prod_{i=1}^{n} \Pr(o_i | h a_1 o_1 \ldots a_i),$$

where $t = a_1 o_1 \ldots a_n o_n$.

2.1. Predictive State Representations

A *test* is a sequence of action-observation pairs that may occur in the future. A test $t = a_1 o_1 \ldots a_n o_n$ succeeds if the observation sequence of the test, $o_1, \ldots, o_n$, is observed when the action sequence, $a_1, \ldots, a_n$, is taken. The null test, ε, is the zero length test that, by definition, always succeeds. The *prediction* for a test t from a particular history h is the probability that the test will succeed, which we define to be $p(t|h)$.[1]

[1] We use a mathematical definition of prediction that differs from the literature. Traditionally $p(t|h)$ is written as $\Pr(o_1 \ldots o_n | h a_1 \ldots a_n)$, where the actions of the test appear to be random variables in the conditional probability. But they are not. They are best thought of as parameters of the joint distribution. In this work we are interested in actions as random variables and so use the more complicated but unambiguous definition.

Suppose there exists a finite set of tests, Q, such that the prediction for any test can be written as a linear combination of predictions of tests in Q. Then we can model the system compactly as a *predictive state representation*. Formally, let $p(Q|h)$ be the row vector of predictions for the tests in Q at history h. So, for all tests t, there exists a column vector m_t such that $\forall h\ p(t|h) = p(Q|h)m_t$. We will call such a Q, a set of *core tests* for the system.

A PSR summarizes histories (i.e., represents state) using predictions of the core tests. In other words, the vector $p(Q|h)$ is the PSR's state representation. After taking action a and receiving observation o, we need to update this state vector of core test predictions. Notice that for $q \in Q$,

$$\begin{aligned} p(q|hao) &= \frac{p(haoq)}{p(hao)} \\ &= \frac{p(aoq|h)p(h)}{p(ao|h)p(h)} = \frac{p(aoq|h)}{p(ao|h)}. \end{aligned}$$

Using the fact that all predictions are linear combinations of core predictions, we get,

$$p(q|hao) = \frac{p(Q|h)m_{aoq}}{p(Q|h)m_{ao}}.$$

So we can compute the new prediction of any core test from the previous core tests' predictions. Hence, a PSR consists of a finite set of core tests Q, an inital prediction vector $p_0 \equiv p(Q|\phi)$, and weight vectors m_{aot} for all $t \in Q \cup \{\varepsilon\}$. The size of a PSR is linear in the number of actions, observations, and rank of the system (i.e., $|Q|$) and is at least as compact as the smallest POMDP (Singh et al., 2004). or k-Markov representation of the same system (Littman et al., 2002).

2.2. Discovery and Learning

One of the advantages of the PSR model is that the representation is strictly in terms of observable quantities, i.e., tests consisting of actions and observations. This feature allows PSRs to be learned solely using data from interaction with the system. Extracting a PSR model is often divided into two subproblems. The problem of finding a set of core tests is called *discovery*, while finding the initial prediction and weight update vectors given a set of core tests is usually termed *learning*. A number of algorithms have been successfully shown to perform both discovery and learning (Singh et al., 2003; James & Singh, 2004; Rosencrantz et al., 2004; Wolfe et al., 2005; Wiewiora, 2005; McCracken & Bowling, 2006). The algorithms differ in both method and assumptions (e.g., some algorithms assume the system can always be reset to the null history while gathering data.)

A common component in most of the existing algorithms is Monte Carlo prediction estimation. Estimates of $p(t|h)$ are collected through interaction with the system and then used to choose core tests and approximate initial prediction and weight vectors. Many approaches proceed in an iterative fashion: alternating between gathering data for Monte Carlo estimates and using the estimates to find core tests and update vectors. This paper focuses on this broad class of algorithms, mainly concentrating on how prediction estimates are made from data.

Some of the Monte Carlo approaches explicitly specify a policy used for data collection (James & Singh, 2004). Other approaches only show results with uncontrolled systems where there are no actions (Rosencrantz et al., 2004). Others make no mention of how data is to be gathered (Wolfe et al., 2005; Wiewiora, 2005) using a uniform-random or other equally blind policy in the experiments. Regardless of how data is collected, all of the approaches use the following Monte Carlo estimator for a prediction,

$$\hat{p}_\bullet(a_1o_1\ldots a_no_n|h) \equiv \frac{\#ha_1o_1\ldots a_no_n}{\#ha_1a_2\ldots a_n},$$

where $\#h$ are counts of the number of times a particular history is observed and $\#ha_1a_2\ldots a_n$ is the number of times the particular sequence of actions is observed after reaching history h regardless of the interleaved observations.[2] In the next section we examine this estimator closely showing that it does not always converge to $p(t|h)$.

3. Non-Blind Policies

To date all of the experimental results for Monte Carlo approaches have involved blind policies. This is either explicitly part of the algorithm or a choice of the experimental setup. However, blind policies are a very narrow class of policies. It is interesting to consider if these algorithms are correct outside of this special case.

Theorem 1 *$\hat{p}_\bullet(t|h)$ is* **not***, in general, an unbiased estimator of $p(t|h)$.*

Proof. Let $t = a_1o_1\ldots a_no_n$. Then,

$$\begin{aligned} &E\left[\frac{\#ha_1o_1\ldots a_no_n}{\#ha_1a_2\ldots a_n}\right] \\ &\quad= E\left[E\left[\frac{\#ha_1o_1\ldots a_no_n}{\#ha_1a_2\ldots a_n}\middle|\#ha_1a_2\ldots a_n\right]\right] \\ &\quad= E\left[\frac{E\left[\#ha_1o_1\ldots a_no_n|\#ha_1a_2\ldots a_n\right]}{\#ha_1a_2\ldots a_n}\right] \\ &\quad= E\left[\frac{\#ha_1a_2\ldots a_n \Pr(o_1\ldots o_n|ha_1\ldots a_n)}{\#ha_1a_2\ldots a_n}\right] \quad (1) \\ &\quad= \Pr(o_1\ldots o_n|ha_1\ldots a_n) \end{aligned}$$

[2] For algorithms that don't require the ability to reset the system, h more precisely refers to a set of histories or contexts.

$$
\begin{aligned}
&= \frac{\Pr(a_1 o_1 \ldots a_n o_n | h)}{\Pr(a_1 \ldots a_n | h)} = \frac{p(t|h)\pi(t|h)}{\Pr(a_1 \ldots a_n | h)} \\
&= p(t|h) \frac{\prod_{i=1}^{n} \Pr(a_i | h a_1 o_1 \ldots o_{i-1})}{\prod_{i=1}^{n} \Pr(a_i | h a_1 \ldots a_{i-1})}, \qquad (2)
\end{aligned}
$$

where 1 follows from the expectation of a binomial and the rest from basic probability and our notation. The estimator is only unbiased when the ratio in Equation 2 is one. This is true only if the policy from history h is conditionally independent of observations given actions, i.e., is blind. If the policy is not blind from history h the estimator is biased. □

As an example, consider the system in Figure 1 where $t = \alpha 0 \alpha 0$ and $h = \phi$.

$$
\begin{aligned}
E\left[\hat{p}_{\bullet}(t|h)\right] &= \Pr(0|\alpha)\Pr(0|\alpha 0 \alpha) \frac{\Pr(\alpha|\phi)\Pr(\alpha|\alpha 0)}{\Pr(\alpha|\phi)\Pr(\alpha|\alpha)} \\
&= \frac{\Pr(0|\alpha)\Pr(0|\alpha 0 \alpha)\Pr(\alpha|\alpha 0)}{\Pr(0|\alpha)\Pr(\alpha|\alpha 0) + \Pr(1|\alpha)\Pr(\alpha|\alpha 1)}
\end{aligned}
$$

Using Policy 1 from Table 1, we have,

$$
E\left[\hat{p}_{\bullet}(t|h)\right] = \frac{(0.1)(0.9)(0.9)}{(0.1)(0.9) + (0.9)(0.1)} = 0.45 \neq p(t|h).
$$

Using Policy 2 from Table 1, we have,

$$
E\left[\hat{p}_{\bullet}(t|h)\right] = \frac{(0.1)(0.9)(0.5)}{(0.1)(0.5) + (0.9)(0.5)} = 0.09 = p(t|h).
$$

The Monte Carlo estimator is in fact only correct when the data is gathered with a blind policy. As motivated in the introduction, non-blind policies arise in many important learning scenarios, e.g., learning while taking actions to achieve some goal or observation guided exploration. Therefore, this is a serious limitation on the majority of existing PSR discovery and learning algorithms.[3] In the next section we present unbiased Monte Carlo estimators for both the case where the policy is known and the case where it is unknown. We then show how non-blind exploration policies can be found and used to speed learning.

4. Corrected Monte Carlo Estimators

In order to construct a PSR from data gathered by a non-blind policy we need an unbiased Monte Carlo estimate of predictions. These estimates can then be used in any of the Monte Carlo discovery and learning algorithms to handle data from an arbitrary policy. We will construct our estimates from sample trajectories of interaction with the system. As above, we use counts, $\#h$, of the number of times a history occurred in the sample trajectories.

Depending on the circumstance, we may have other knowledge as well. In particular, the non-blind policy used to gather the data may be known, either because we are given the policy or because we are actually choosing the policy. In this case, $\pi(h)$ is known or computable for any history. It is also possible that the policy is not known. For example, the data may have been generated by another agent (e.g., a human) interacting with the system. We look at each case in turn.

4.1. Policy is Known

Let us suppose the policy is known. Let,

$$
\hat{p}_{\pi}(t|h) = \frac{\#ht}{\#h} \frac{1}{\pi(t|h)}. \qquad (3)
$$

Theorem 2 *$\hat{p}_{\pi}(t|h)$ is an unbiased estimator of $p(t|h)$.*

Proof.

$$
\begin{aligned}
E\left[\frac{\#ht}{\#h}\frac{1}{\pi(t|h)}\right] &= E\left[E\left[\frac{\#ht}{\#h}\frac{1}{\pi(t|h)}\middle| \#h\right]\right] \\
&= E\left[\frac{E\left[\#ht|\#h\right]}{\#h\,\pi(t|h)}\right] \\
&= E\left[\frac{\#h\ \Pr(ht)/\Pr(h)}{\#h\,\pi(t|h)}\right] \qquad (4) \\
&= \frac{\Pr(ht)/\Pr(h)}{\pi(t|h)} \\
&= \frac{p(ht)\pi(ht)}{p(h)\pi(h)\pi(t|h)} = p(t|h),
\end{aligned}
$$

where 4 follows from the expectation of a binomial and the rest from basic probability and our notation. □

Not only is $\hat{p}_{\pi}(t|h)$ unbiased, we can also compute its variance conditioned on observing n trajectories reaching history h.[4]

$$
\begin{aligned}
&V\left[\hat{p}_{\pi}(t|h)\middle| \#h = n\right] \\
&\quad = \frac{V\left[\#ht|\#h = n\right]}{n^2\pi(t|h)^2} \\
&\quad = \frac{n\Pr(ht)/\Pr(h)(1 - \Pr(ht)/\Pr(h))}{n^2\pi(t|h)^2} \\
&\quad = \frac{\pi(t|h)p(t|h)(1 - \pi(t|h)p(t|h))}{n\pi(t|h)^2} \\
&\quad = \frac{p(t|h)(1 - \pi(t|h)p(t|h))}{n\pi(t|h)} \\
&\quad = \frac{p(t|h)}{n\pi(t|h)} - \frac{p(t|h)^2}{n}.
\end{aligned}
$$

[3] The myopic gradient algorithm (Singh et al., 2003), which does not address discovery, and the constrained gradient algorithm (McCracken & Bowling, 2006) are the two exceptions. Neither uses Monte Carlo estimates.

[4] Since there is some non-zero probability that no trajectory will reach history h, the unconditional variance is not defined.

Since $p(t|h)$ is not generally known we may want to find a bound on the variance. In particular, the value of $p(t|h)$ that maximizes the variance is $(2\pi(t|h))^{-1}$. Substituting we get the following bound,

$$\begin{aligned} V\left[\hat{p}_\pi(t|h)|\#h=n\right] &= \frac{p(t|h)}{n\pi(t|h)} - \frac{p(t|h)^2}{n} \\ &\leq \frac{1}{2n\pi(t|h)^2} - \frac{1}{4n\pi(t|h)^2} \\ &= \frac{1}{4n\pi(t|h)^2}. \end{aligned} \tag{5}$$

We will use this bound in Section 6.

4.2. Policy is Not Known

Suppose the policy generating the data is not known. Let,

$$\hat{p}_{\not\pi}(t|h) = \prod_{i=1}^{n} \frac{\#ha_1o_1\ldots a_io_i}{\#ha_1o_1\ldots a_i}, \tag{6}$$

where $t = a_1o_1\ldots a_no_n$.

Theorem 3 $\hat{p}_{\not\pi}(t|h)$ *is an unbiased estimator of* $p(t|h)$.

Proof. We will prove by induction on n. If $n=1$, then the policy is essentially blind for this test since there are no preceding observations to condition actions upon. Formally,

$$\begin{aligned} E\left[\frac{\#ha_1o_1}{\#h}\right] &= E\left[E\left[\frac{\#ha_1o_1}{\#ha_1}\middle|\#ha_1\right]\right] \\ &= E\left[\frac{\#ha_1\Pr(o_1|ha_1)}{\#ha_1}\right] \\ &= \Pr(o_1|ha_1) = p(a_1o_1|h). \end{aligned}$$

Now, suppose $\hat{p}_{\not\pi}$ is an unbiased estimator for all $n-1$ length tests. Consider a test t of length n.

$$\begin{aligned} & E\left[\prod_{i=1}^{n}\frac{\#ha_1o_1\ldots a_io_i}{\#ha_1o_1\ldots a_i}\right] \\ &= E\left[E\left[\prod_{i=1}^{n}\frac{\#ha_1o_1\ldots a_io_i}{\#ha_1o_1\ldots a_i}\middle| \begin{array}{c}\forall j\in\{1,\ldots n\} \\ \#ha_1o_1\ldots a_j \\ \#ha_1o_1\ldots o_{j-1}\end{array}\right]\right] \end{aligned}$$

We're conditioning on all of the random variables in the product but $\#ha_1o_1\ldots a_no_n$. So all but this term can be pulled outside the inner expectation. The remaining term is conditionally independent of all of the conditioning variables but one.

$$= E\left[\frac{\left(\prod_{i=1}^{n-1}\frac{\#ha_1o_1\ldots a_io_i}{\#ha_1o_1\ldots a_i}\right) E[\#ha_1o_1\ldots a_no_n|\#ha_1o_1\ldots a_n]}{\#ha_1o_1\ldots a_n}\right]$$

$$\begin{aligned} &= E\left[\frac{\left(\prod_{i=1}^{n-1}\frac{\#ha_1o_1\ldots a_io_i}{\#ha_1o_1\ldots a_i}\right)\#ha_1o_1\ldots a_n\Pr(o_n|ha_1o_1\ldots a_n)}{\#ha_1o_1\ldots a_n}\right] \\ &= E\left[\prod_{i=1}^{n-1}\frac{\#ha_1o_1\ldots a_io_i}{\#ha_1o_1\ldots a_i}\right]\Pr(o_n|ha_1o_1\ldots a_n) \end{aligned}$$

By the induction hypothesis, the expectation becomes,

$$\begin{aligned} &= \left(\prod_{i=1}^{n-1}\Pr(o_i|ha_1o_1\ldots a_i)\right)\Pr(o_n|ha_1o_1\ldots a_n) \\ &= \prod_{i=1}^{n}\Pr(o_i|ha_1o_1\ldots a_i) = p(t|h), \end{aligned}$$

thus concluding the proof. □

5. Exploration in PSRs

With a correct estimate of predictions we can now correctly incorporate data from non-blind policies when constructing a PSR. We will use this feature to develop a novel exploration mechanism to speed discovery and learning of PSR models. Our approach to exploration assumes that data is gathered in iterations. For each iteration, we will construct a suitable, non-blind exploration policy to be executed to gather data. The new estimates will then be combined with the estimates from the previous iterations. In this section we describe how to find a suitable exploration policy. In the next section, we show results of learning using non-blind exploration policies with the James and Singh reset algorithm (James & Singh, 2004).

5.1. Combining Estimators

Before we consider how to construct an exploration policy for the next iteration, we examine how to combine estimates from multiple iterations. Since each estimate involves a variable amount of data using a different policy, it is not a simple matter of averaging estimates.[5]

Suppose X_1 and X_2 are unbiased estimates of some unknown value. We want to find the minimum variance weighted combination of these two estimates. In other words, find an α that minimizes,

$$\begin{aligned} & V\left[\alpha X_1 + (1-\alpha)X_2\right] \\ &= \alpha^2 V[X_1] + (1-\alpha)^2 V[X_2] \\ &= \alpha^2(V[X_1]+V[X_2]) - 2\alpha V[X_2] + V[X_2]. \end{aligned} \tag{7}$$

Taking the derivative with respect to α and setting to zero gives us,

$$\alpha = \frac{V[X_2]}{V[X_1]+V[X_2]} \tag{8}$$

[5]Although any weighted combination of unbiased estimates is unbiased, we want our weighted combination to have small variance.

$$
\begin{aligned}
&= \frac{V[X_1]^{-1}}{(V[X_1]+V[X_2])/V[X_1]V[X_2]} \\
&= \frac{V[X_1]^{-1}}{V[X_1]^{-1}+V[X_2]^{-1}},
\end{aligned}
$$

and substituting 8 into 7 gives,

$$
\begin{aligned}
V\left[\alpha X_1 + (1-\alpha)X_2\right] &= V[X_2] - \frac{V[X_2]^2}{V[X_1]+V[X_2]} \\
&= \frac{V[X_1]V[X_2]}{V[X_1]+V[X_2]} \\
&= \left(V[X_1]^{-1}+V[X_2]^{-1}\right)^{-1}.
\end{aligned}
$$

Basically, estimates should be weighted proportionally to their inverse variance. The inverse variance of the resulting estimator is just the sum of the individual inverse variances.

5.2. Variance Minimizing Exploration

Monte Carlo based algorithms compute estimates of $p(t|h)$ for specific sets of tests and histories. Since our estimator is unbiased, we reduce error in the estimate by simply reducing the variance of the estimates. The problem of exploration for discovery and learning can be seen as simply finding a policy that minimizes variance. Or equivalently, we want a policy that maximizes inverted variance.

We will assume data is gathered in iterations. Let $v_i(h,t)$ be the variance of our estimate $\hat{p}(t|h)$ at iteration i, and k_i the number of trajectories to be sampled for iteration i. Then,

$$
\begin{aligned}
&E\left[v_i(h,t)^{-1}\right] \\
&\quad= E\left[v_{i-1}(h,t)^{-1} + V\left[\frac{\#ht}{\#h}\frac{1}{\pi(t|h)}\middle|\#h\right]^{-1}\right] \\
&\quad\geq v_{i-1}(h,t)^{-1} + E\left[4\#h\;\pi(t|h)^2\right] \\
&\quad= v_{i-1}(h,t)^{-1} + 4k_i\;p(h)\pi(h)\pi(t|h)^2.
\end{aligned}
$$

We can further bound this term to get,

$$
\begin{aligned}
E\left[v_i(h,t)^{-1}\right] &\geq v_{i-1}(h,t)^{-1} + 4k_i\;p(h)\pi(h)^2\pi(t|h)^2 \\
&= v_{i-1}(h,t)^{-1} + 4k_i\;p(h)\pi(ht)^2.
\end{aligned}
$$

For convenience we seek to maximize the square root of the expected inverted variance above, which is equivalent since square root is a monotonic function. We can then use the fact that square root is a concave function [6] to arrive at the following final bound,

$$
\begin{aligned}
&\sqrt{E\left[v_i(h,t)^{-1}\right]} \\
&\quad\geq \sqrt{v_{i-1}(h,t)^{-1} + 4k_i\;p(h)\pi(ht)^2} \\
&\quad= \sqrt{2}\sqrt{\frac{v_{i-1}(h,t)^{-1} + 4k_i\;p(h)\pi(ht)^2}{2}} \\
&\quad\geq \frac{\sqrt{2}}{2}\left(\sqrt{v_{i-1}(h,t)^{-1}} + \sqrt{4k_i\;p(h)\pi(ht)^2}\right) \\
&\quad= \frac{\sqrt{2}}{2}\left(\sqrt{v_{i-1}(h,t)^{-1}} + 2\sqrt{k_i\;p(h)}\pi(ht)\right).
\end{aligned}
$$

[6]If f is a concave function $f\left(\frac{a}{2}+\frac{b}{2}\right) \geq \frac{f(a)}{2}+\frac{f(b)}{2}$.

The Objective. We can now define an optimization problem to find a policy that will minimize the variance of our estimates. We cannot simutaneously minimize the variance of all estimates, since our policy must explicitly trade off exploring some tests and histories for others. Therefore, we attempt to minimize the expected variance of the highest-variance estimate,

$$\operatorname*{argmin}_{\pi} \max_{h,t} E\left[v_i(h,t)\right].$$

As we noted above, this is equivalent to maximizing the smallest root expected inverse variance.

$$\operatorname*{argmax}_{\pi} \min_{h,t} \sqrt{E\left[v_i(h,t)^{-1}\right]}.$$

Root expected inverse variance is well-defined (as opposed to expected variance), but it is inconvenient from an optimization perspective. So we will instead maximize our lower bound, pushing up root expected inverse variance in the process. We get the following optimization,

$$\operatorname*{argmax}_{\pi} \min_{h,t} \left(\sqrt{v_{i-1}(h,t)^{-1}} + 2\sqrt{k_i\;p(h)}\pi(ht)\right), \quad (9)$$

where the constant factor on the root expected inverse variance falls out of the optimization. Notice that this problem is linear in the policy variables $\pi(h)$, which are essentially a sequence form representation of the policy (Koller et al., 1996).

The Constraints. Using the objective from 9, we still have to constrain the variables $\pi(h)$ to conform to a valid representation of policy. In particular, the following constraints are sufficient for a policy in sequence form,

1. $\pi(\phi) = 1$,
2. $\forall h, o \in \mathcal{O} \quad \pi(h) = \sum_a \pi(hao)$, and
3. $\forall h, a \in \mathcal{A}, \{o, o'\} \subseteq \mathcal{O} \quad \pi(hao) = \pi(hao')$.

Notice that these constraints are all linear in the policy variables $\pi(h)$.

The Optimization. We can combine the above linear objective and linear constraints into a simple linear program. For each iteration of the discovery and learning algorithm we receive a list of history-test pairs, the previous estimates' inverse variances, and the number of trajectories to be sampled, k_i, using the exploration policy. The linear

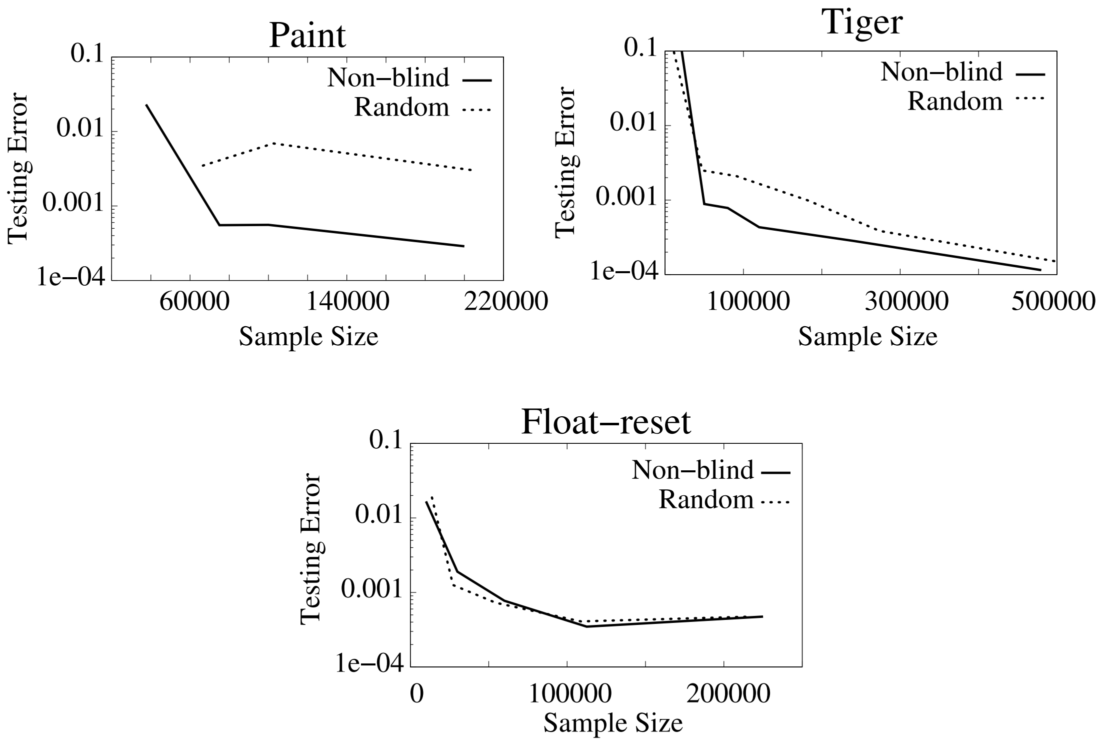

Figure 2. Results of PSR learning using non-blind policies as compared to blind random policies for three systems. Note that the error uses a logarithmic scale. See text for details.

program produces the policy variables $\pi(h)$ for every history that is a prefix of some history-test pair. The policy can then be computed,

$$\Pr(a|h) = \frac{\pi(hao)}{\pi(h)}.$$

and used to gather data for the next iteration of the algorithm.

6. Results

As mentioned earlier, existing PSR discovery and learning algorithms use blind policies, as opposed to our algorithm which computes and follows a non-blind policy. The following experiments evaluate the difference in performance between these two types of policies. The blind policy used for these experiments is the uniform-random policy, selecting each action with the same probability.

In evaluating PSR discovery and learning algorithms, a common technique (James & Singh, 2004) is to include a *reset*, which is an additional action that brings the system to a fixed, but unknown, state. Existing work (Wolfe et al., 2005) shows how to extend discovery and learning algorithms to the more general case without reset. However, since our objective is just to evaluate the effect of blind and non-blind policies, our experiments include a reset.

We selected three dynamical systems that are commonly used in evaluating PSRs and POMDPs, whose definitions are available online (Cassandra, 1999). Our experimental method used the reset-based PSR discovery and learning algorithm (James & Singh, 2004), but used the above optimization to compute non-blind policies for action selection during exploration. A typical PSR discovery and learning run involves a number of iterations, each of which will identify specific histories and tests for which additional samples would be desirable. During each iteration, these sets of histories and tests are passed into the linear program, along with estimates of $p(h)$ and $\#h$, for appropriate h. The optimization is run five times for each iteration of discovery and learning, in order to maintain a reasonably current exploration policy.

After the discovery and learning algorithm has completed (see James and Singh (2004) for details on the stopping condition), the resulting PSR model is evaluated. This is accomplished by running the model for a long sequence following the random policy, and computing the average one-step error. The error for a run is,

$$\frac{1}{L}\sum_{t=1}^{L}\sum_{o\in\mathcal{O}}[\Pr(o|h_t a_t) - \hat{\Pr}(o|h_t a_t)]^2,$$

where $\Pr(o|h_t a_t)$ is the true probability of observation o in history h_t when the randomly selected action a_t is taken, and $\hat{\Pr}(o|h_t a_t)$ is the corresponding prediction from the estimated model. For these experiments, $L = 100,000$. This procedure tests all the parameters of the PSR model, not just the parameters for one-step prediction, because the update procedure, which is used L times, will require the use of all of the parameters.

Results are presented in Figure 2, displaying data for a number of different discovery and learning runs for each system. The x-axis shows the amount of data used for a particular run, measured by the total number of sample trajectories. Each trajectory begins with the reset action and continues until the next reset is taken. The y-axis shows the error as defined above. Note that these graphs use a logarithmic scale, so small differences in the graphs correspond to order of magnitude improvements in test error.

The test error when using the exploration policy in the paint domain is dramatically lower than when using the blind random exploration policy. The improvement is less dramatic in the tiger domain, but still amounts to a factor of two reduction in the amount of data needed to learn a model with equivalent accuracy. The exploration policy performs similarly to the random policy in the float-reset domain. This is not surprising since the the first few core-tests of the domain are both very easy to learn and are enough to attain fairly accurate predictions. In addition, it should be noted that the test error is measured while following the random policy, which may bias the results in favor of random exploration.

7. Conclusion

The Monte Carlo estimator for predictions is integral to the majority of current algorithms for discovery and learning in PSRs. The problem, as proven in this paper, is that it is only an unbiased estimator when a blind policy is used to gather data. Simple examples demonstrate that this bias can be extreme depending on the policy. We provided a corrected Monte Carlo estimator that is unbiased for all policies, even when the policy is not known. We further provided a new variance-minimizing exploration algorithm and demonstrated its improvement over a blind policy exploration algorithm in common non-Markovian test domains.

Acknowledgments

We would like to thank Peter Hooper for his initial observation of the inconsistency of Monte Carlo estimates. We would also like to thank Richard Sutton for numerous discussions. This work was partially funded by NSERC, iCore, and Alberta Ingenuity through the Alberta Ingenuity Centre for Machine Learning.

References

Cassandra, A. (1999). Tony's POMDP file repository page. http://www.cs.brown.edu/research/ai/pomdp/examples/.

James, M. R., & Singh, S. (2004). Learning and discovery of predictive state representations in dynamical systems with reset. *Proceedings of the 21st International Conference on Machine Learning (ICML)* (pp. 719–726).

Koller, D., Megiddo, N., & von Stengel, B. (1996). Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, *14*, 247–259.

Littman, M., Sutton, R. S., & Singh, S. (2002). Predictive representations of state. *Advances in Neural Information Processing Systems 14 (NIPS)* (pp. 1555–1561). MIT Press.

McCracken, P., & Bowling, M. (2006). Online learning of predictive state representations. *Advances in Neural Information Processing Systems 18 (NIPS)*. MIT Press. To appear.

Rosencrantz, M., Gordon, G., & Thrun, S. (2004). Learning low dimensional predictive representations. *Proceedings of the 21st International Conference on Machine Learning (ICML)* (pp. 695–702).

Singh, S., James, M. R., & Rudary, M. R. (2004). Predictive state representations: A new theory for modeling dynamical systems. *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI)* (pp. 512–519).

Singh, S., Littman, M., Jong, N., Pardoe, D., & Stone, P. (2003). Learning predictive state representations. *Proceedings of the Twentieth International Conference on Machine Learning (ICML)* (pp. 712–719).

Wiewiora, E. (2005). Learning predictive representations from a history. *Proceedings of the 22nd International Conference on Machine Learning (ICML)* (pp. 969–976).

Wolfe, B., James, M. R., & Singh, S. (2005). Learning predictive state representations in dynamical systems without reset. *Proceedings of the 22nd International Conference on Machine Learning (ICML)* (pp. 985–992).