Journal of Machine Learning Research 15 (2014) 3575-3619 Submitted 12/13; Revised 7/14; Published 11/14

Efficient Learning and Planning with Compressed Predictive States

William Hamilton WILLIAM.HAMILTON2@MAIL.MCGILL.CA
Mahdi Milani Fard MMILANI1@CS.MCGILL.CA
Joelle Pineau JPINEAU@CS.MCGILL.CA
School of Computer Science
McGill University
Montreal, QC, Canada

Editor: Jan Peters

Abstract

Predictive state representations (PSRs) offer an expressive framework for modelling partially observable systems. By compactly representing systems as functions of observable quantities, the PSR learning approach avoids using local-minima prone expectation-maximization and instead employs a globally optimal moment-based algorithm. Moreover, since PSRs do not require a predetermined latent state structure as an input, they offer an attractive framework for model-based reinforcement learning when agents must plan without a priori access to a system model. Unfortunately, the expressiveness of PSRs comes with significant computational cost, and this cost is a major factor inhibiting the use of PSRs in applications. In order to alleviate this shortcoming, we introduce the notion of compressed PSRs (CPSRs). The CPSR learning approach combines recent advancements in dimensionality reduction, incremental matrix decomposition, and compressed sensing. We show how this approach provides a principled avenue for learning accurate approximations of PSRs, drastically reducing the computational costs associated with learning while also providing effective regularization. Going further, we propose a planning framework which exploits these learned models. And we show that this approach facilitates model-learning and planning in large complex partially observable domains, a task that is infeasible without the principled use of compression.[1]

Keywords: predictive state representation, reinforcement learning, dimensionality reduction, random projections

1. Introduction

In the reinforcement learning (RL) paradigm, an agent in a system acts, observes, and receives feedback in the form of numerical signals (Sutton and Barto, 1998). Given this experience, the agent determines an optimal policy (i.e., a guide for its future actions) via value-function based dynamic programming or parameterized policy search. This is conceptually analogous to the 'operant conditioning' postulated to underlie certain forms of

1. An earlier version of this work appeared as: W.L. Hamilton, M. M. Fard, and J. Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In *Proceedings of the Thirtieth International Conference on Machine Learning*, 2013.

animal (and human) learning. Organisms learn to repeat actions that give positive feedback and avoid those with negative results.

1.1 Fully to Partially Observable Domains

In the standard formulation, an RL agent is given prior knowledge of a domain in the form of a state-space, transition probabilities, and an observation (i.e., sensor) model. Formally, the system is described by a Markov decision process (MDP), and given the MDP description, a variety of optimization algorithms may then be used to solve the problem of determining an optimal action policy (Sutton and Barto, 1998). In general, approximate solutions are determined for domains exhibiting large, or even moderate, dimensionality (Gordon, 1999).

The situation is further complicated in domains exhibiting partial observability, where observations are aliased and do not fully determine an agent's state in a system. For example, an agent's sensors may indicate the presence of nearby objects but not the agent's global position within an environment. To accommodate this uncertainty, the MDP framework is extended as partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). Here, the true state is not known with certainty, and optimization algorithms must act upon belief states (i.e., probability distributions over the state-space).

1.2 Model-Learning Before Planning

The POMDP extension introduces a measure of uncertainty in the reinforcement learning paradigm. Nevertheless, an agent learning a policy via the POMDP framework has access to considerable a priori knowledge: Most centrally, the agent (which necessarily and implicitly contains the POMDP solver) has access to a description of the system in the form of an explicit state-space representation. Moreover, in a majority of instances, the agent knows the probabilities governing the transitions between states, the observation functions governing the emission of observable quantities from these states, and the reward function specifying some empirical measure of "goodness" for each state (Kaelbling et al., 1998).

Access to such knowledge allows for the construction of optimal (or near-optimal) plans and is useful for real-world applications where considerable domain-specific knowledge is available. However, the converse situation, where a (near)-complete system model is not known a priori, is both important and lags behind in terms of research results. In such a setting, an agent must learn a system model prior to (or while simultaneously) learning an action policy.

At an application level, there are many situations in which expert knowledge is sparse, and it is possible that even application domains with domain-knowledge could benefit from the use of algorithms that learn system models prior to planning and that are thus free from unintended biases introduced via expert-specified system models. At a more theoretical level, the development of general agents that both learn system models and plan using such models is fundamental in the pursuit of creating truly intelligent artificial agents that can learn and succeed independent of prior domain knowledge.

1.3 Learning a Model-based Predictive Agent

In this work we outline an algorithm for constructing a learning and planning agent for sequential decision-making under partial state observability. At a high-level, the algorithm is model-based, specifying an agent that builds a model of its environment through experience and then plans using this learned model. Such a model-based approach is necessary in complicated partially observable domains, where single observations are far from sufficient statistics for the state of the system (Kaelbling et al., 1998). At its core, the algorithm relies on the powerful and expressive model class of predictive state representations (PSRs) (Littman et al., 2002). PSRs (described in detail in Section 2) are an ideal candidate for the construction of an agent that both learns a system model and plans using this model, as they do not require a predetermined state-space as an input.

PSRs have been used as the basis of model-based reinforcement learning agents in a number of recent works (Boots et al., 2010; Rosencrantz et al., 2004; Ong et al., 2012; Izadi and Precup, 2008; James and Singh, 2004). However, for these previous approaches, the time and space complexities of learning scale super-linearly in the maximum length of the trajectories used (see Section 3). In this work we use an approach that simultaneously ameliorates the efficiency concerns related to constructing PSRs and alleviates the need for domain-specific feature construction. The model-learning algorithm, termed compressed predictive state representation (CPSR), uses random projections in order to efficiently learn accurate approximations of PSRs in sparse systems. In addition, the approach makes use of recent advancements in incrementally learning transformed PSRs (TPSRs), providing further optimization (Boots and Gordon, 2011). The details of the model-learning algorithm are provided in Section 3.2. Section 4 presents theoretical results pertaining to the accuracy of the approximate learned model and elucidates how our approach regularizes the learned model, trading off reduced variance for controlled bias.

The planning algorithm used is an extension of the fitted-Q function approximation-based planning algorithm for fully observable systems (Ernst et al., 2005). This approach has been applied to PSRs previously with some success (Ong et al., 2012) and provides a strong alternative to point-based value iteration methods (Izadi and Precup, 2008). The algorithm simply substitutes a predictive state for the observable MDP state in a fitted-Q learning algorithm, and a function approximator is used to learn an approximation of the Q-function for the system (i.e., the function mapping predictive states and actions to expected rewards). The details of the planning approach are outlined in Section 5. The main empirical contribution of this work is the application of this approach to domains and sample-sizes of complexity not previously feasible for PSRs. Section 6 will highlight empirical results demonstrating the performance of the algorithm on some synthetic robot navigation domains and a difficult real-world application task based upon the ecological management of migratory bird species.

This work builds upon the algorithm presented in Hamilton et al. (2013), extending it in a number of ways. Specifically, this work (1) permits a broader class of projection matrices, (2) includes optional compression of both histories and tests, (3) combines compressed sensing with incremental matrix decomposition to facilitate incremental/online learning, (4) provides a more detailed theoretical analysis of the model-learning algorithm, (5) explicitly includes a planning framework, which exploits the learned CPSR models in a principled

manner, and (6) provides extensive empirical results pertaining to both model-learning and planning, including results on a difficult real-world problem.

2. Predictive State Representations

Predictive state representations (PSRs) offer an expressive and powerful framework for modelling dynamical systems and thus provide a suitable foundation for a model-based reinforcement learning agent. In the PSR framework, a predictive model is constructed directly from execution traces, utilizing minimal prior information about the domain (Littman et al., 2002; Singh et al., 2004). Unlike latent state based approaches, such as hidden Markov models or POMDPs, PSR states are defined only via observable quantities. This not only makes PSRs more general, as they do not require a predetermined state-space, but it also increases their expressive power relative to latent state based approaches (Littman et al., 2002). In fact, the PSR paradigm subsumes POMDPs as a special case (Littman et al., 2002). In addition, PSRs facilitate model-learning without the use of local-minima prone expectation-maximization (EM) and allow for the efficient construction of globally optimal models via a method-of-moments based algorithm (James and Singh, 2004). The following section outlines the foundations of the PSR approach and sets the stage for the presentation of compressed predictive state representations in Section 3 and our efficient learning algorithm in Section 3.2. Much of the PSR background material (e.g., the derivation of the PSR model in Sections 2.2 and 2.3) expands upon the presentation in Boots et al. (2010) and uses important results from that work.

2.1 Notation

This section outlines the notation that will be used throughout the paper.

2.1.1 Matrix Algebra Notation

Bold letters denote vectors $\mathbf{v} \in \mathbb{R}^d$ and matrices $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. Given a matrix $\mathbf{M}$, $\|\mathbf{M}\|$ denotes its Frobenius norm. $\mathbf{M}^\dagger$ is used to denote the Moore–Penrose pseudoinverse of $\mathbf{M}$. Sometimes names are given to the columns and rows of a matrix using ordered index sets $\mathcal{I}$ and $\mathcal{J}$. In this case, $\mathbf{M} \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$ denotes a matrix of size $|\mathcal{I}| \times |\mathcal{J}|$ with rows indexed by $\mathcal{I}$ and columns indexed by $\mathcal{J}$. We then specify entries in a matrix (or tensor) using these indices and the bracket notation; e.g., $[\mathbf{M}]_{i,j}$ corresponds to the entry in the row indexed by $i \in \mathcal{I}$ and the column indexed $j \in \mathcal{J}$. Rows or columns of a matrix are specified using this index notation and the $*$ symbol; e.g., $[\mathbf{M}]_{i,*}$, denotes the ith row of $\mathbf{M}$. Finally, given $\mathcal{I}' \subset \mathcal{I}$ and $\mathcal{J}' \subset \mathcal{J}$ we define $[\mathbf{M}]_{\mathcal{I}',\mathcal{J}'}$ as the submatrix of $\mathbf{M}$ with rows and columns specified by the indices in $\mathcal{I}'$ and $\mathcal{J}'$, respectively.

2.1.2 Probability Notation

We denote the probability of an event by $\mathbb{P}(\cdot)$ and use $|$ to denote the usual probabilistic conditioning. To avoid excessive notation, when the $\mathbb{P}(\cdot)$ operator is applied to a vector of events, it is understood as returning a vector of probabilities unless otherwise indicated (i.e., a single operator is used for single events and vectors of events).

For clarity, we use || to denote conditioning upon an agents policy (i.e., plan). That is, || denotes that we are conditioning upon the knowledge that the agent will "intervene" in a system by executing the specified actions.

2.2 Technical Foundations

A PSR model represents a partially observable system's state as a probability distribution over future events. More formally, we maintain a probability distribution over different sequences of possible future action-observation pairs. Such sequences of possible future action-observations are termed *tests* and denoted τ. For example, we could construct a test $\tau_i = [o_{t+1}^{k_1}, o_{t+2}^{k_2}, ..., o_{t+n}^{k_n} || a_{t+1}^{l_1}, a_{t+2}^{l_2}, ..., a_{t+n}^{l_n}]$, where notationally subscripts refer to time, superscripts identify particular actions or observations, and actions following the || symbol denote that we are conditioning upon the agent "intervening" by performing those specified actions at the specified times. We can then say that such a test is *executed* if the agent intervenes and takes the specified actions, and we say the test *succeeded* if the observations received by the agent match those specified by the test. Going further, we can define the probability of success for test τ_i as

$$\mathbb{P}(\tau_i) = \mathbb{P}(o_{t+1}^{k_1}, o_{t+2}^{k_2}, ..., o_{t+n}^{k_n} || a_{t+1}^{l_1}, a_{t+2}^{l_2}, ..., a_{t+n}^{l_n}).$$

Of course, we want to know more than just the unconditioned probabilities of success for each test. A complete model of a dynamical system also requires knowing the success probabilities for each test conditioned on the agent's previous experience, or *history*. We denote such a history $h_j = [a_0^{l_0} o_0^{k_0}, a_1^{l_1} o_1^{k_1} ... a_t^{l_t} o_t^{k_t}]$, where again subscripts denote time and superscripts identify particular actions or observations. Importantly, the || symbol for intervention is absent from the definition of history, as the sequence of actions specified in a history are assumed to have already been executed.

Finally, given that an agent has performed some actions and received some observations, defining some history h_j, we compute

$$\mathbb{P}(\tau_i^{\mathcal{O}} | h_j || \tau_i^{\mathcal{A}}),$$

the probability of τ_i succeeding conditioned upon the agent's current history in the system, where $\tau_i^{\mathcal{A}}$ and $\tau_i^{\mathcal{O}}$ denote the ordered lists of actions and observations, respectively, defined in τ_i.

It is not difficult to see that a partially observable system is completely described by the conditional success probabilities of all tests given all histories. That is, if we have $\mathbb{P}(\tau_i^{\mathcal{O}} | h_j || \tau_i^{\mathcal{A}}) \; \forall i \; \forall j$ then we trivially have all necessary information to characterize the dynamics of a system. Of course, maintaining all such probabilities directly is infeasible, as there is a potentially infinite number of tests and histories (and at the very least an exorbitant number for any system of even moderate complexity) (Littman et al., 2002).

Fortunately, it has been shown that it suffices to remember only the conditional probabilities for a (potentially) small *core set* of tests, and the conditional probabilities for all other tests may be defined as linear functions of the conditional probabilities for the tests in this core set (Littman et al., 2002).[2] More formally, we define the system dynamics ma-

2. In this work, the shortened phrase *core set* is always to be interpreted as *core set of tests*; that is, such sets always correspond to a set of tests.

trix, $\mathbf{H}$, as the (potentially infinite size) matrix, where each row corresponds to a particular test (under some lexicographic ordering), each column to a particular history (under some lexicographic ordering), and a particular $[\mathbf{H}]_{i,j}$ entry to $\mathbb{P}(\tau_i^{\mathcal{O}}|h_j||\tau_i^{\mathcal{A}})$. $\mathbf{H}$ simply organizes $\mathbb{P}(\tau_i^{\mathcal{O}}|h_j||\tau_i^{\mathcal{A}})$, $\forall i \forall j$ in a matrix structure. In Littman et al. (2002) and Singh et al. (2004) it is shown that if $\mathbf{H}$ has rank k then (1) k corresponds to the rank of the partially observable system, as defined by Jaeger (2000) and (2) there exists a *minimal core set* of size k (i.e., the smallest core set of tests is of size k, though there may be larger core sets). Thus, if $\mathbf{H}$ has rank k, it suffices to remember conditional probabilities for only k tests (those that are a part of the minimal core set), and the conditional probabilities for all other tests may be defined as *linear* functions of the conditional probabilities for these tests.

The rank of $\mathbf{H}$ thus describes the complexity of a system. For example, a system with $\text{rank}(\mathbf{H}) = k$ can not be modelled by a POMDP with less than k states; though it may require more than k POMDP states (Singh et al., 2004). In contrast, a PSR can always (exactly) model a system with $\text{rank}(\mathbf{H}) = k$ using a minimal core set of exactly size k (Singh et al., 2004). This demonstrates how PSRs can be more compact than POMDPS.

Thus, for a PSR, given a minimal core set $\mathcal{Q}$ (i.e., $|\mathcal{Q}| = \text{rank}(\mathbf{H})$), we can compute the conditional probability of some test $\tau_i \notin \mathcal{Q}$ as

$$\mathbb{P}(\tau_i^{\mathcal{O}}|h_j||\tau_i^{\mathcal{A}}) = \mathbf{r}_{\tau_i}^{\top}\mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_j||\mathcal{Q}^{\mathcal{A}}),$$

where $\mathbf{r}_{\tau_i}$ is a vector of weights and $\mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_j||\mathcal{Q}^{\mathcal{A}})$ an ordered vector of conditional probabilities for each test in the minimal core set $q_i \in \mathcal{Q}$. Integral to this approach is the fact that restricting the model to linear functions of tests in the minimal core set does not preclude the modelling of non-linear systems, as the dynamics implicit in the probabilities may specify non-linear behaviours (Littman et al., 2002).

Thus, given the functions mapping tests in the core set to all other tests, it suffices to maintain, at time t, only the vector $\mathbf{m}_t = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_t||\mathcal{Q}^{\mathcal{A}})$, where h_t is the history of the system at time t. That is, it suffices to maintain only the vector of conditional probabilities for the tests in a core set (which is usually assumed to be minimal) .

2.3 The PSR Model

Formally, a PSR model of a system is defined by $\langle \mathcal{O}, \mathcal{A}, \mathcal{Q}, \mathcal{F}, \mathbf{m}_0 \rangle$, where $\mathcal{O}$ and $\mathcal{A}$ define the possible observations and actions respectively, $\mathcal{Q}$ is a minimal core set of tests, $\mathcal{F}$ defines a set of linear functions mapping success probabilities for tests in the minimal core set to the probabilities for all tests, and $\mathbf{m}_0$ defines the initial state of the system (i.e., $\mathbf{m}_0 = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}||\mathcal{Q}^{\mathcal{A}})$). Since $\mathcal{F}$ contains only linear functions, its elements can be specified as vectors of weights. These vectors, in turn, are specified using a finite set of linear operators (i.e., matrices). Specifically, we define a linear operator $\mathbf{M}_{a^l o^k}$ for each action-observation pair such that

$$\begin{aligned}\mathbb{P}(o_{t+1}^k|h_t||a_{t+1}^l) &= \mathbf{m}_{\infty}^{\top}\mathbf{M}_{a^l o^k}\mathbb{P}(Q^{\mathcal{O}}|h_t||Q^{\mathcal{A}}) \\ &= \mathbf{m}_{\infty}^{\top}\mathbf{M}_{a^l o^k}\mathbf{m}_t,\end{aligned}$$

where $\mathbf{m}_{\infty}$ is a constant normalizer such that $\mathbf{m}_{\infty}^{\top}\mathbf{m}_t = 1$, $\forall t$.

These operators map probabilities of tests in the specified minimal core set to the probabilities for single action-observation pairs and may be recursively combined to generate the full set of linear functions in $\mathcal{F}$. For instance, for the test $\tau_i = [o^{k_1}_{t+1}, o^{k_2}_{t+2}, ..., o^{k_n}_{t+n} || a^{l_1}_{t+1}, a^{l_2}_{t+2}, ..., a^{l_n}_{t+n}]$, we compute

$$\begin{aligned}\mathbb{P}(\tau_i^{\mathcal{O}}|h_t||\tau_i^{\mathcal{A}}) &= \mathbf{r}_{\tau_i}^\top \mathbb{P}(Q^{\mathcal{O}}|h_t||Q^{\mathcal{A}}) \\ &= \mathbf{m}_\infty^\top \mathbf{M}_{a^{l_n}o^{k_n}} \cdots \mathbf{M}_{a^{l_2}o^{k_2}} \mathbf{M}_{a^{l_1}o^{k_1}} \mathbf{m}_t. \end{aligned} \tag{1}$$

These operators can also be used to produce n-step predictions (i.e., the probability $\mathbb{P}(o^k_{t+n}|h_t||a^l_{t+n})$ of seeing an observation, o^k, after taking action, a^l, n-steps in the future) by

$$\mathbb{P}(o^k_{t+n}|h_t||a^l_{t+n}) = \mathbf{m}_\infty^\top \mathbf{M}_{a^l o^k} (\mathbf{M}_\star)^{n-1} \mathbf{m}_t,$$

where $\mathbf{M}_\star = \sum_{a^l o^k \in \mathcal{A}\times\mathcal{O}} \mathbf{M}_{a^l o^k}$ is a matrix that can be computed once and stored as a parameter for quick computation (Wiewiora, 2007).

Lastly, the operators provide a convenient method for updating the predictive state, defined by the prediction vector $\mathbf{m}_t$, as an agent tracks through a system and receives observations. The prediction vector $\mathbf{m}_t$ is updated to $\mathbf{m}_{t+1}$ after an agent takes an action a^l and receives observation o^k using

$$\begin{aligned}\mathbf{m}_{t+1} &= \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_{t+1}||\mathcal{Q}^{\mathcal{A}}) \\ &= \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_t a^l o^k||\mathcal{Q}^{\mathcal{A}}) \\ &= \frac{\mathbf{M}_{a^l o^k}\mathbf{m}_t}{\mathbf{m}_\infty^\top \mathbf{M}_{a^l o^k}\mathbf{m}_t}. \end{aligned} \tag{2}$$

Together, the elements of $\langle \mathcal{O}, \mathcal{A}, \mathcal{Q}, \mathcal{F}, \mathbf{m}_0 \rangle$ (where $\mathcal{F}$ is understood to contain the linear operators described above and the normalizer) thus provide a succinct model of a system, which allows for the efficient computation of event probabilities and also facilitates conditioning upon observed histories.

2.4 Learning PSRs

There is a considerable amount of literature describing different approaches to learning PSRs. We provide an overview of the standard approaches, as Section 3.2 describes, in detail, the efficient compressed learning approach we propose.[3]

In general, PSR learning approaches may be divided into two distinct classes: discovery-based and subspace-based. In the discovery-based approach, a form of combinatorial search is used to discover the (minimal) core set of tests, and the PSR model is then computed in a straightforward manner given the explicit knowledge of $\mathcal{Q}$ (James and Singh, 2004; James et al., 2005). This method generates an exact PSR model. However, the combinatorial search required to find $\mathcal{Q}$ precludes the use of this approach in domains of even moderate cardinality.

Unlike the discovery-based approaches, subspace-based approaches obviate the need for determining $\mathcal{Q}$ exactly (Hsu et al., 2008; Boots et al., 2010; Rosencrantz et al., 2004).

3. For a slightly more detailed discussion of existing PSR learning approaches see Wiewiora (2007).

Instead, subspace-identification techniques (e.g., spectral methods) are used in order to find a subspace that is a linear transformation of the subspace defined by $\mathcal{Q}$ (Rosencrantz et al., 2004). The linear nature of the PSR model allows the use of this transformed PSR model in place of the exact PSR model without detriment. Specifically, it can be shown that the probabilities obtained via such a transformed model are consistent with those obtained via the true model (Boots et al., 2010).

Formally, one first specifies a large (non-minimal) core set of tests $\mathcal{T}$ and a set of histories $\mathcal{H}$. Next, one defines two *observable matrices* $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$, $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$, and $|\mathcal{A}| \times |\mathcal{O}|$ *observable matrices* $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$ (one for each action-observation pair). $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$ is a $|\mathcal{T}| \times |\mathcal{H}|$ matrix which contains the joint probabilities of all specified tests and histories. $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$ is a $|\mathcal{H}| \times 1$ vector containing the marginal probabilities of each possible history. And each $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$ is a $|\mathcal{T}| \times |\mathcal{H}|$ matrix containing the the joint probabilities of all specified tests and histories where a particular action-observation pair (indicated by the subscript) is appended to the history (Boots et al., 2010). These observable matrices can be viewed as submatrices of $\mathbf{H}$, the system dynamics matrix (e.g., $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}} = [\mathbf{H}]_{\mathcal{T},\mathcal{H}}$). We also define matrices $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}}$ $\forall ao \in \mathcal{A} \times \mathcal{O}$ analogously but with $\mathcal{Q}$ replacing $\mathcal{T}$ (e.g., $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}} = [\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}]_{\mathcal{Q},*}$).

Under the assumption that the empty history occurs first in the lexicographic ordering of $\mathcal{H}$, the discovery-based approach builds a PSR model by

$$\mathbf{m}_0 = [\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}]_{*,1} \tag{3}$$

$$\mathbf{m}_\infty^\top = \boldsymbol{\mathcal{P}}_{\mathcal{H}}^\top (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger, \tag{4}$$

$$\mathbf{M}_{ao} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}} (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger, \tag{5}$$

while the subspace-based approach builds a model by

$$\boldsymbol{\beta}_0 = [\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}]_{*,1} \tag{6}$$

$$\boldsymbol{\beta}_\infty^\top = \boldsymbol{\mathcal{P}}_{\mathcal{H}}^\top (\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}})^\dagger, \tag{7}$$

$$\mathbf{B}_{ao} = \mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}} (\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}})^\dagger, \tag{8}$$

where $\mathbf{Z}$ is the projection matrix defining the subspace used for learning, which satisfies certain conditions. The conditions upon $\mathbf{Z}$ and the standard selection criterion for choosing it are elucidated in Section 2.5 below.

From these equations we see that PSR learning, in both the subspace and discovery paradigms, corresponds to a set of regression problems. The pseudoinverses in (3)-(8) corresponding to solutions to a set regression problems. For example, in the learning of $\mathbf{m}_\infty$ the columns of $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}$ correspond to samples in the regression (i.e., each history is a sample), the rows to features (i.e., each test is a feature), and the regression targets are the entries of $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$ (i.e., the marginal history vector).

In general, the complexity of the discovery-based learning approach is dominated by the combinatorial search for the set of core tests. In the worst case this search has time-complexity $O((|\mathcal{A}||\mathcal{O}|)^L)$, where L is the max-length of a trajectory (i.e., execution trace) used to learn the model. If the minimal core set of tests is provided as input, the discovery-based method has complexity $O(|\mathcal{H}||\mathcal{Q}|^2)$; however, the assumption that the minimal core set of tests is known is not realistic in practice. In contrast, the subspace-based approach has time-complexity $O(|\mathcal{H}||\mathcal{T}|d_{\mathbf{Z}})$, where $d_{\mathbf{Z}}$ is the column-dimension of $\mathbf{Z}$. If the size of the minimal core set of tests is known (an unrealistic assumption) then $d_{\mathbf{Z}} = |\mathcal{Q}|$.

2.5 Transformed Representations

PSR models learned via the subspace method are often referred to as transformed PSRs (TPSRs), since they learn a model that is an invertible transform of a standard PSR model. More formally, given the set of linear parameters defining a PSR model and an invertible matrix $\mathbf{J}$, we can construct a TPSR by applying $\mathbf{J}$ as a linear operator to each parameter. That is, we set $\boldsymbol{\beta}_0 = \mathbf{J}\mathbf{m}_0$, $\boldsymbol{\beta}_\infty^\top = \mathbf{m}_\infty^\top \mathbf{J}^{-1}$, and $\mathbf{B}_{ao} = \mathbf{J}\mathbf{M}_{ao}\mathbf{J}^{-1}\ \forall ao \in \mathcal{A} \times \mathcal{O}$, and these new transformed matrices constitute the TPSR model (Boots and Gordon, 2011). It is easy to see that the $\mathbf{J}$'s cancel out in the prediction equation (1) and update equation (2). Intuitively, TPSRs can be thought of as maintaining a predictive state upon an invertible linear transform of the state defined by the tests in the minimal core set.

In practice, the matrix $\mathbf{J}$ is determined by the projection matrix $\mathbf{Z}$, which is used during learning in the subspace-based paradigm. To make the relationship between $\mathbf{J}$ and $\mathbf{Z}$ explicit, we define the following matrices: $\mathbf{R} = (\mathbf{r}_{\tau_i}, \mathbf{r}_{\tau_2}, ..., \mathbf{r}_{\tau_{|\mathcal{T}|}})^\top \in \mathbb{R}^{\mathcal{T} \times \mathcal{Q}}$, with each row i corresponding to the linear function mapping the probabilities of tests in the minimal core set to the probability of test τ_i (i.e., the $\mathbf{r}_{\tau_i}$ as defined in Equation 1); $\mathbf{N} = \text{diag}(\boldsymbol{\mathcal{P}}_\mathcal{H}) \in \mathbb{R}^{\mathcal{H} \times \mathcal{H}}$, with the marginal history probabilities along the diagonal; and, $\mathbf{Q} \in \mathbb{R}^{\mathcal{Q} \times \mathcal{H}}$, with each column j equal to the expected probability vector for the tests in the minimal core set given that history h_j has been observed (i.e., $[\mathbf{Q}]_{*,j} = \mathbf{M}_{h_j}\mathbf{m}_0$). These matrices can then be used to define a factorization of the observable matrices. In particular, Boots et al. (2010) show that

$$\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}} = \mathbf{R}\mathbf{Q}\mathbf{N} \tag{9}$$

and that

$$\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}} = \mathbf{R}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N} \tag{10}$$

holds for all $ao \in \mathcal{A} \times \mathcal{O}$.

Examining the equations for the different learning methods (i.e., Equations 3 and 6) and using the factorizations given in (9) and (10), we see first that for the discovery-based method, which learns a true untransformed PSR, we have that

$$\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}} = \mathbf{I}\mathbf{Q}\mathbf{N},$$

where $\mathbf{I}$ is the identity. In this case the set of tests in $\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}}$ is the minimal core set, and thus the core set mapping operator $\mathbf{R}$ is replaced by the identity. Similarly, we have

$$\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}} = \mathbf{I}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N}.$$

Thus for the discovery method

$$\begin{aligned}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}}(\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger &= \mathbf{M}_{ao}\mathbf{Q}\mathbf{N}(\mathbf{Q}\mathbf{N})^\dagger \\ &= \mathbf{M}_{ao},\end{aligned}$$

where we used the fact that $\mathbf{QN}$ is full column-rank by definition. By contrast, for the subspace learning algorithm we have, assuming that $\mathbf{ZR}$ has full row-rank,

$$\begin{aligned}\mathbf{B}_{ao} &= \mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}(\mathbf{Z}\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}})^\dagger \\ &= \mathbf{Z}\mathbf{R}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N}(\mathbf{Z}\mathbf{R}\mathbf{Q}\mathbf{N})^\dagger \\ &= \mathbf{Z}\mathbf{R}\mathbf{M}_{ao}\mathbf{Q}\mathbf{N}(\mathbf{Q}\mathbf{N})^\dagger(\mathbf{Z}\mathbf{R})^\dagger \\ &= \mathbf{Z}\mathbf{R}\mathbf{M}_{ao}(\mathbf{Z}\mathbf{R})^\dagger,\end{aligned} \tag{11}$$

where we again used the fact that $\mathbf{QN}$ has full column-rank. If we further assume that $\mathbf{ZR}$ is invertible (i.e., is square in addition to being full row rank) then (11) simplifies to

$$\mathbf{ZRM}_{ao}(\mathbf{ZR})^{\dagger} = \mathbf{ZRM}_{ao}(\mathbf{ZR})^{-1}.$$

Similar results hold for $\boldsymbol{\beta}_{\infty}$ and $\boldsymbol{\beta}_0$, showing that the subspace learning method does, in fact, return TPSRs in the case where $\mathbf{ZR}$ is invertible, and in this case we have a transformed representation with $\mathbf{J} := \mathbf{ZR}$.

The final piece of a TPSR is the specification of $\mathbf{Z}$, the projection matrix defining the subspace used during learning (and implicitly defining the transformation matrix $\mathbf{J}$). We know from the above derivations that $\mathbf{Z}$ must be chosen such that $\mathbf{ZR}$ is invertible. The standard method for guaranteeing this is by choosing $\mathbf{Z}$ via spectral techniques; that is, $\mathbf{Z}$ is set to be $\mathbf{U}^{\top}$, the transpose of the matrix of right singular vectors (from the thin-SVD of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$) (Boots et al., 2010).

The TPSR approach can also be extended to work with features of tests and histories (Boots et al., 2010; Boots and Gordon, 2011) and/or kernelized to work in continuous domains (Boots and Gordon, 2013). This is useful in cases where the observation space is too complex for standard tests to be used (i.e., when the observation space is structured or continuous). When features of tests and histories are used, however, they are usually specified in a domain-specific manner (Boots et al., 2010). Some authors have also used randomized Fourier methods to efficiently approximate kernel-based feature selection (Boots and Gordon, 2011). These methods are quite successful in continuous domains (Boots et al., 2010; Boots and Gordon, 2011, 2013).

In contrast, the benefit of the algorithm presented in Section 3.2 is that it implicitly performs general purpose feature selection (for discrete-domains) using random compression. And this is especially useful in cases where it is difficult to know a sufficient set of features prior to training (e.g., in the case where the model is being learned incrementally). Moreover, the motivation between the compression performed in this work and the above-mentioned feature-based techniques are disjoint in that the goal of this work is to provide compression for efficient learning whereas the above-mentioned feature-based learning strategies are motivated by the need to cope with continuous or structured observation spaces. See Section 7.2 for further discussion on the relationship between this work and these alternative feature-based approaches.

3. Compressed Predictive State Representations

In this section, we describe our extension of PSRs, compressed predictive state representations (CPSRs). The CPSR approach, at its core, combines the state-of-the-art in subspace PSR learning with recent advancements in compressed sensing. This marriage provides an extremely efficient and principled approach for learning accurate transformed approximations of PSRs in complex systems, where learning a full PSR is simply intractable. Section 3.1 motivates the use of compressed sensing techniques in a PSR learning algorithm, and Section 3.2 describes the efficient CPSR learning approach we propose.

3.1 Foundations: Compressed Estimation

Despite the fact that non-compressed subspace-based algorithms, such as TPSR, can specify a small dimension for a transformed space (e.g., by removing the least important singular vectors of $\mathbf{U}$ as in done in Rosencrantz et al. (2004) and analyzed in Kulesza et al. (2014)), there are still a number of computational limitations. To begin, TPSRs require that the $|\mathcal{T}| \times |\mathcal{H}|$ matrix, $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$, be estimated in its entirety, and that the $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$ matrices be partially estimated as well. Moreover, since the naive TPSR approach must compute a spectral decomposition of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$ it has computational complexity $O(|\mathcal{H}||\mathcal{T}|^2)$, in the batch (and incremental mini-batch) setting, assuming the observable matrices are given as input. Thus in domains that require many (possibly long) trajectories for learning or that have large observation spaces, such as those described in Section 6, the naive TPSR approach becomes intractable, since $|\mathcal{H}|$ and $|\mathcal{T}|$ both scale as $O(L|Z|)$, where L is the max length of a trajectory in a training set Z of size $|Z|$.[4][5] In order to circumvent these computational constraints (and provide a form of regularization), the CPSR learning algorithm we propose (in the next section) performs *compressed estimation*.

This method is borrowed from the field of compressed sensing and works by projecting matrices down to low-dimensional spaces determined via randomly generated bases. More formally, a $m \times n$ matrix $\mathbf{Y}$ is compressed to a $d \times n$ matrix $\mathbf{X}$ (where $d < m$) by

$$\mathbf{X} = \boldsymbol{\Phi}\mathbf{Y}, \tag{12}$$

where $\boldsymbol{\Phi}$ is a $d \times m$ *Johnson-Lindenstrauss matrix* (i.e., a matrix satisfying the Johnson-Lindenstrauss lemma) (Baraniuk and Wakin, 2009). Intuitively, a Johnson-Lindenstrauss matrix is a random matrix defining a low-dimensional embedding which approximately preserves Euclidean distances between projected points (i.e., the projection preserves the dot-product between vectors). Different choices for $\boldsymbol{\Phi}$ are discussed in Section 6. It is worth noting that in our case, the matrix multiplication in (12) is in fact performed "online", and the matrices corresponding to $\mathbf{X}$ and $\boldsymbol{\Phi}$ are never explicitly held in memory (details in Section 3.2).

The fidelity of this technique depends what is called the *sparsity* of the matrix $\mathbf{Y}$. Sparsity in this context refers to the maximum number of non-zero entries which occur in any column of $\mathbf{Y}$. Formally, if we denote a column vector of $\mathbf{Y}$ by $\mathbf{y}_i$, we say that a matrix is k-sparse if

$$k \geq ||\mathbf{y}_i||_0 \ \ \forall \mathbf{y}_i \in \mathbf{Y},$$

where $||\cdot||_0$ denotes Donoho's zero "norm" (which simply counts the number of non-zero entries in a vector).

The technique is very well suited for application to PSRs. Informally, the sparsity condition is the requirement that for every history h_j, only a subset of all tests have non-zero probabilities (a more formal definition appears in the theory section below). This

4. Note that $|\mathcal{H}|$ and $|\mathcal{T}|$ scale linearly with the number of *observed* test/histories. The $O(L|Z|)$ bound is thus pessimistic in that it assumes each training instance is unique.
5. It is worth noting that no explicit bounds on the sample complexity of PSR learning have been elucidated. However, the sample complexity bounds of Hsu et al. (2008) provide results for a special case of TPSR learning (i.e., no actions and only single length tests and histories). In general, PSR approaches are consistent estimators but cannot be assumed to be data efficient (thus emphasizing the need to accommodate large sample sizes).

seems realistic in many domains. For example, in the PocMan domain described below, we empirically found the average column sparsity of the matrices to be roughly 0.018% (i.e., approximately 0.018% of entries in a column were non-zero). Moreover, as we will demonstrate empirically in Section 6, certain noisy observation models induce sparsity that can be exploited by this approach.

3.2 Efficiently Learning CPSRs

In this section, we present our novel compressed predictive state representation (CPSR) learning algorithm. The algorithm builds upon the work of Hamilton et al. (2013), extending their algorithm in a number of important ways. Specifically, the algorithm presented here (1) permits a broad class of compression matrices (any full-rank projection matrix satisfying the JL lemma), (2) includes optional compression of both histories and tests, and (3) combines compressed sensing with spectral methods in order to provide numerical stability and facilitate incremental (and even online) model-learning. Section 3.2.1 describes the foundational batch-learning algorithm. Section 3.2.2 describes how to incrementally update a learned model with new data efficiently for deployment in online settings.

3.2.1 Batch Learning of CPSRs

To begin, we define two injective functions: $\phi_{\mathcal{T}} : \mathcal{T} \rightarrow \mathbb{R}^{d_{\mathcal{T}}}$ and $\phi_{\mathcal{H}} : \mathcal{H} \rightarrow \mathbb{R}^{d_{\mathcal{H}}}$. These functions are independent mappings from tests and histories, respectively, to columns of independent random full-rank Johnson-Lindenstrauss (JL) projection matrices $\boldsymbol{\Phi}_{\mathcal{T}} \in \mathbb{R}^{d_{\mathcal{T}} \times \mathcal{T}}$ and $\boldsymbol{\Phi}_{\mathcal{H}} \in \mathbb{R}^{d_{\mathcal{H}} \times \mathcal{H}}$, respectively. The matrices are defined via these functions since the full sets $\mathcal{T}$ and $\mathcal{H}$ may not be known a priori, and we can get away with this "lazy" specification since the columns of JL projection matrices are determined by independent random variables.

Next, given a training trajectory z of action-observation pairs of any length, let $\mathbb{I}_{h_j}(z)$ be an indicator function taking a value of 1 if the action-observations pairs in z correspond to h_j. Similarly define $|\cdot|$ as the length of a sequence (e.g., of action-observation pairs) and let $\mathbb{I}_{h_j,\tau_i}(z)$ be an indicator function taking a value of 1 if z can be partitioned such that, starting from some index k within the sequence, there are $|h_j|$ action-observation pairs corresponding to those in $h_j \in \mathcal{H}$ and the next $|\tau_i|$ pairs correspond to those in $\tau_i \in \mathcal{T}$.[6]

Given a batch of of training trajectories Z we compute compressed estimates

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{\mathcal{H}} &= \boldsymbol{\Phi}_{\mathcal{H}} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{H}} \\ &= \sum_{z \in Z} \sum_{h_j \in \mathcal{H}} \mathbb{I}_{h_j}(z) \phi_{\mathcal{H}}(h_j) \end{aligned} \tag{13}$$

and

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}} &= \boldsymbol{\Phi}_{\mathcal{T}} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}} \boldsymbol{\Phi}_{\mathcal{H}}^{\top} \\ &= \sum_{z \in Z} \sum_{t_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j,t_i}(z) \left[\phi_{\mathcal{T}}(t_i) \oplus \phi_{\mathcal{H}}(h_j) \right] \end{aligned} \tag{14}$$

6. In this work we use $k = 0$. That is we do not use the suffix history estimation algorithm (Wolfe et al., 2005), where k is varied in the range $[0, |z|)$. Using $k = 0$ minimizes dependencies between estimation errors as the same samples are not used to get estimates for multiple histories.

of the observable matrices $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$, respectively, where $\oplus$ denotes the tensor (outer) product of two vectors.[7]

Next, we compute the $\hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^\top$ rank-d' thin SVD of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$:

$$(\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}) = \text{SVD}(\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}). \tag{15}$$

Given these matrices we can construct

$$\mathbf{c}_1 = \hat{\mathbf{S}}\hat{\mathbf{V}}^\top \mathbf{e} \tag{16}$$

and

$$\mathbf{c}_\infty^\top = \hat{\boldsymbol{\Sigma}}_{\mathcal{H}}^\top \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}, \tag{17}$$

the compressed and transformed estimates of $\mathbf{m}_1$ and $\mathbf{m}_\infty$, respectively, where $\mathbf{e}$ is a vector such that $\boldsymbol{\Phi}_{\mathcal{H}}\mathbf{e} = (1, 0, 0, ..., 0)^\top$. In practice this can be guaranteed by defining a modified history map $\phi'_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^{d+1}$ such that that for the null history, $\emptyset$, $\phi'_{\mathcal{H}}(\emptyset) = (1, 0, 0, ..., 0)^\top$ and that $\phi'_{\mathcal{H}}(h_j) = [0 \;\; \phi_{\mathcal{H}}(h_j)]$ for all $h_j \neq \emptyset$. This specification of $\mathbf{e}$ assumes that all $z \in Z$ are starting from a unique start state. If this is not the case, then we set $\mathbf{e}$ such that $\boldsymbol{\Phi}_{\mathcal{H}}\mathbf{e} = (1, 1, 1, ..., 1)^\top$, which again can be guaranteed without cost but in this case by simply adding a constant "dummy" column to the front of $\boldsymbol{\Phi}_{\mathcal{H}}$. In this latter scenario, we would, in fact, not be learning $\mathbf{c}_1$ exactly and instead would learn $\mathbf{c}_*$, an arbitrary feasible state as our start state. The uncertainty in our state estimate should decrease, however, as we update and track through our system and the process mixes (Boots et al., 2010). And indeed, the majority of domains without well-defined start-states are those for which there is significant mixing over time, so this technique should introduce only a small amount of error in practice.

Given the SVD of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$, we can also estimate the $\mathbf{C}_{ao}$ matrices, the compressed and transformed versions of the $\mathbf{M}_{ao}$ matrices, directly via a second pass over the data. First, however, we must define a third class of indicator functions on $z \in Z$: $\mathbb{I}_{h_j,ao,\tau_i}(z)$ takes value 1 if and only if the training sequence z can be partitioned such that, starting from some index k within the sequence, there are $|h_j| + 1$ action-observation pairs corresponding to h_j appended with a particular $ao \in \mathcal{A} \times \mathcal{O}$ and the next $|\tau_i|$ correspond to those in τ_i. In other words, $\mathbb{I}_{h_j,ao,\tau_i}(z)$ is equivalent to $\mathbb{I}_{h'_j,\tau_i}(z)$, where a particular $ao \in \mathcal{A} \times \mathcal{O}$ is appended to the history h'_j. Using these indicators and the SVD matrices of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$, we compute, for each $ao \in \mathcal{A} \times \mathcal{O}$,

$$\mathbf{C}_{ao} = \sum_{z \in Z} \sum_{t_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j,ao,t_i}(z) \left[\left(\hat{\mathbf{U}}^\top \phi_{\mathcal{T}}(t_i) \right) \oplus \left(\hat{\mathbf{S}}^{-1} \hat{\mathbf{V}}^\top \phi_{\mathcal{H}}(h_j) \right) \right]. \tag{18}$$

Thus, in two passes over the data, we are able to efficiently construct our CPSR model parameters. The primary computational savings engendered by this approach is in the computation of the pseudoinverse of $\hat{\boldsymbol{\Sigma}}_{\mathcal{T},\mathcal{H}}$, which we implicitly compute via an SVD. Since we are performing pseudoinversion (i.e., SVD) on a compressed matrix, the computational

7. We do not normalize our probability estimates in the estimation equations since the normalization constants cancel out during learning.

complexity is uncoupled from the number of tests and histories in the set of observed trajectories Z. Recalling that L denotes the max length of a trajectory in Z and letting $|Z|$ denote the number of trajectories in the set Z, this approach has a computational complexity of

$$O\left(L|Z|d_{\mathcal{H}}d_{\mathcal{T}} + d_{\mathcal{T}}^2 d_{\mathcal{H}}\right) = O\left(L|Z|\right) \tag{19}$$

since $d_{\mathcal{H}}$ and $d_{\mathcal{T}}$ are a user-specified constants (assuming the standard cubic computational cost for the SVD).[8] Without compression (i.e., with naive TPSR), a computational cost of

$$O\left(L|Z| + |\mathcal{H}||\mathcal{T}|^2\right) = O\left(L^3|Z|^3\right) \tag{20}$$

is incurred.

In addition to these computational savings, the above approach has the added benefit of not requiring that $\mathcal{T}$ and $\mathcal{H}$ be known in entirety prior to learning. This is especially important in the case where we want to alternate model-learning and planning/exploration phases using incremental updates (described below), as it is very unlikely that all possible tests and histories are observed in the first round of exploration. Performing SVD on the compressed matrices also induces a form of regularization (similar to L_2 regularization) on the learned model, where variance is reduced at the cost of a controlled bias (details in Section 4).

3.2.2 Incremental Updates to the Model

In addition to straightforward batch learning, it is also possible to incrementally update a learned model, given new training data, Z' (Boots and Gordon, 2011). This is especially useful in that it facilitates alternating exploration and exploitation phases. Of course, if such a non-blind alternating approach is used then the distribution of the training data changes (i.e., it becomes non-stationary), and the sampled trajectories can no longer be assumed to be i.i.d.. Despite this theoretical drawback, Ong et al. (2012) show that non-blind sampling approaches can lead to better planning results in a small sample setting.[9]

Briefly, we obtain a new $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ estimate and update our $\hat{\mathbf{\Sigma}}_{\mathcal{H}}$ estimate using using (14) and (13) with Z'. Next, we update our SVD matrices, given our additive update to $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$, using the methods of (Brand, 2002). The $\mathbf{c}_1$ and $\mathbf{c}_\infty$ vectors are then re-computed exactly as in equations (16) and (17).

To obtain our $\mathbf{C}_{ao}^{\text{new}}$ matrices, we compute

$$\begin{aligned}\mathbf{C}_{ao}^{\text{new}} = &\sum_{z \in Z'} \sum_{t_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j, ao, t_i}(z) \left[\left(\hat{\mathbf{U}}_{\text{new}}^\top \phi_{\mathcal{T}}(t_i)\right) \oplus \left(\hat{\mathbf{S}}_{\text{new}}^{-1} \hat{\mathbf{V}}_{\text{new}}^\top \phi_{\mathcal{H}}(h_j)\right)\right] \\ &+ \hat{\mathbf{U}}_{\text{new}}^\top \hat{\mathbf{U}}_{\text{old}} \mathbf{C}_{ao}^{\text{old}} \hat{\mathbf{S}}_{\text{old}} \hat{\mathbf{V}}_{\text{old}}^\top \hat{\mathbf{V}}_{\text{new}} \hat{\mathbf{S}}_{\text{new}}^{-1}.\end{aligned} \tag{21}$$

The first term in (21) corresponds to estimating the contribution to the new $\mathbf{C}_{ao}$ matrix from the new data, and the second term is the projection of the old $\mathbf{C}_{ao}$ matrix onto the new basis. Using the results of Brand (2002), the complexity of this update is

$$O(L'|Z'|(d_{\mathcal{T}}d_{\mathcal{H}} + (d')^3 + d'd_{\mathcal{T}}) + d_{\mathcal{T}}d'd_{\mathcal{H}}), \tag{22}$$

8. Section 4 describes how the choice of these constants affects the accuracy of the learned model.
9. In this work, where larger sample sizes were used, we did not find a significant benefit to goal-directed sampling and in fact saw detrimental effects in terms of planning ability and numerical stability during learning. See Section 7 for details.

where L' denotes the maximum length of a trajectory in Z'.

4. Theoretical Analysis of the Learning Algorithm

In the following section, we describe theoretical properties of the CPSR learning approach. Our analysis proceeds in two stages. First, we show that the learned model is consistent in the case where $d_{\mathcal{T}} \geq |\mathcal{Q}|$ and $d_{\mathcal{H}} \geq |\mathcal{Q}|$ (i.e., when no real compression occurs). Following this, we outline results bounding the induced approximation error (bias) and decrease in estimation error (variance) due to learning a compressed model.

The analysis included in this section is intended as a means to justify the compression technique and study the overall consistency of our algorithm. It also provides guidance for the choosing of a theoretically sound range of values for the projection size used in the algorithm.

4.1 Consistency of the Learning Approach

The following adapts the results of Boots et al. (2010) and shows the consistency of our learning approach when the random projection dimension is greater than or equal to the true underling dimension of the system (i.e., the size of the minimal core set of tests, $|\mathcal{Q}|$). We then describe the implications of this result for the case where we are in fact projecting down to a dimension smaller than $|\mathcal{Q}|$.

4.1.1 Consistency in the Non-Compressed Setting

We begin by noting a fundamental result from the TPSR literature. Recall the matrix $\mathbf{R} = (\mathbf{r}_{\tau_1}, \mathbf{r}_{\tau_2}, ..., \mathbf{r}_{\tau_{|\mathcal{T}|}})^\top \in \mathbb{R}^{\mathcal{T} \times \mathcal{Q}}$ where each row, $\mathbf{r}_i$, specifies the linear map

$$\mathbf{r}_{\tau_i}^\top \mathbb{P}(Q^{\mathcal{O}} | h_t || Q^{\mathcal{A}}) = \mathbb{P}(\tau_i^{\mathcal{O}} | h_t || \tau_i^{\mathcal{A}}).$$

Supposing that $d_{\mathcal{T}} \geq |\mathcal{Q}|$ and $d_{\mathcal{H}} \geq |\mathcal{Q}|$ and with $\mathbf{U}$ coming from the SVD of $\mathbf{\Sigma}_{\mathcal{T},\mathcal{H}}$, we have

$$\mathbf{c}_0 = (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R}) \mathbf{m}_0, \tag{23}$$

$$\mathbf{c}_\infty^\top = \mathbf{m}_\infty^\top (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})^{-1}, \tag{24}$$

$$\mathbf{C}_{ao} = (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R}) \mathbf{M}_{ao} (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})^{-1}. \tag{25}$$

That is, we simply recover a TPSR where $\mathbf{J} = (\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})$, and it has been shown that the above implies a *consistent* learning algorithm (Boots et al., 2010; Boots and Gordon, 2011). We note that $\mathbf{\Phi}_{\mathcal{T}}$ appears in these consistency equations, while $\mathbf{\Phi}_{\mathcal{H}}$ does not, emphasizing the different roles these two matrices occupy. This difference will play an important role in the theoretical analysis below.

4.1.2 Extension to the Compressed Case

In the case where $d_{\mathcal{T}} < |\mathcal{Q}|$ and/or $d_{\mathcal{H}} < |\mathcal{Q}|$ things are not as straightforward. Specifically, equations (23)-(25) no longer hold as $(\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{T}} \mathbf{R})$ is no longer invertible (it is in fact, no longer square), since the SVD is taken on $\mathbf{\Sigma}_{\mathcal{T},\mathcal{H}}$ which has rank less than $|\mathcal{Q}|$ when $d_{\mathcal{T}} < |\mathcal{Q}|$ and/or $d_{\mathcal{H}} < |\mathcal{Q}|$ while the column dimension of $\mathbf{R}$ is $|\mathcal{Q}|$. The primary focus of our theoretical

analysis is the effect of this fact, i.e. $(\mathbf{U}^\top \mathbf{\Phi}_\mathcal{T} \mathbf{R})$ not being invertible. We show how we can view $\mathbf{\Phi}_\mathcal{T}$ as inducing a form of compressed linear regression, and we provide bounds on the excess risk of learning within a compressed space.

There is, however, the additional complication of $\mathbf{\Phi}_\mathcal{H}$ when $d_\mathcal{H} < |\mathcal{Q}|$, as in that setting it is no longer possible to remove $\mathbf{\Phi}_\mathcal{H}$ from the consistency equations (23)-(25). From the perspective of regression, $\mathbf{\Phi}_\mathcal{H}$ can be viewed as compressing the number of samples, while $\mathbf{\Phi}_\mathcal{T}$ can be viewed as compressing the features. In this work, we focus on the effect of compressing tests and provide detailed analysis of how compressing tests (i.e., features) affects the implicit linear regression performed. Zhou et al. (2007) discuss the effect of compressing samples during regression, a result that follows naturally from the Johnson-Lindenstrauss lemma, and in Section 7, we discuss these results and their relationship to this work. For completeness, Section 6 also provides an empirical analysis of the effects of compressing histories and tests versus compressing tests alone.

4.2 Effects of Compression

In what follows, we analyse the effects of compression by viewing $\mathbf{\Phi}_\mathcal{T}$ as inducing a form of compressed linear regression, where both the input data and targets are compressed.

4.2.1 Preliminaries

This approach is justified by noting that, as discussed in Section 2.4, in equations (17) and (18) of our learning algorithm we are in fact performing implicit linear regression. That is, for $(\hat{\mathbf{U}}, \hat{\mathbf{S}}, \hat{\mathbf{V}}) = \text{SVD}(\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}})$,

$$\begin{aligned} \hat{\mathbf{V}}\hat{\mathbf{S}}^{-1} &= (\hat{\mathbf{U}}^\top \hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}})^\dagger \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}} \mathbf{\Phi}_\mathcal{H}^\top)^\dagger. \end{aligned} \tag{26}$$

In other words, $\hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}$ is the Moore-Penrose pseudoinverse of $\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_\mathcal{H} \mathbf{\Phi}_\mathcal{H}^\top$, and multiplication by $\hat{\mathbf{V}}\hat{\mathbf{S}}^{-1}$ is thus equivalent to performing least-squares linear regression.

Following the discussion in the previous section and to avoid unnecessary complication, we assume $\mathbf{\Phi}_\mathcal{H}$ has orthonormal columns (i.e., is not compressive) while analyzing the effects of compressing the tests. In the case where $\mathbf{\Phi}_\mathcal{H}$ has orthonormal columns, we define $\mathbf{\Sigma}_{\mathcal{T},ao,\mathcal{H}}$ as the compressed analogue of $\boldsymbol{\mathcal{P}}_{\mathcal{T},ao,\mathcal{H}}$, and see that (18) can be rewritten as

$$\begin{aligned} \mathbf{C}_{ao} &= (\hat{\mathbf{U}}^\top \hat{\mathbf{\Sigma}}_{\mathcal{T},ao,\mathcal{H}})(\hat{\mathbf{U}}^\top \hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}})^\dagger \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},ao,\mathcal{H}} \mathbf{\Phi}_\mathcal{H}^\top)(\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}} \mathbf{\Phi}_\mathcal{H}^\top)^\dagger \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},ao,\mathcal{H}}) \mathbf{\Phi}_\mathcal{H}^\top (\mathbf{\Phi}_\mathcal{H}^\top)^\dagger (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}})^\dagger \qquad (27) \\ &= (\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},ao,\mathcal{H}})(\hat{\mathbf{U}}^\top \mathbf{\Phi}_\mathcal{T} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{T},\mathcal{H}})^\dagger, \qquad (28) \end{aligned}$$

where (27)-(28) holds since $\mathbf{\Phi}_\mathcal{H}$ is assumed to have orthonormal columns. An analogous result holds for $\mathbf{c}_\infty$ and thus, $\mathbf{\Phi}_\mathcal{H}$ can, indeed, be omitted in our analysis (under the assumption that $\mathbf{\Phi}_\mathcal{H}^\top \mathbf{\Phi}_\mathcal{H} = \mathbf{I}$).

Moreover, we ignore the $\hat{\mathbf{U}}^\top$ term in what follows, which is justified in the case where $d' = d_\mathcal{T}$ (i.e., when the truncated SVD dimension is equal to the test compression dimension). This $d' = d_\mathcal{T}$ condition is very mild in the sense that the use of SVD during learning is

primarily motivated by the need to efficiently compute pseudoinverses, which facilitates the efficient batch and incremental model-learning algorithms. That is, the SVD is not used as a dimensionality reduction technique, as random projections are used in that role.[10] Thus, under the assumption that $d' = d_{\mathcal{T}}$, we have that

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow \hat{\mathbf{U}}^\top \mathbf{A}\mathbf{x} = \hat{\mathbf{U}}^\top \mathbf{b}$$

holds, since $\hat{\mathbf{U}}^\top$ is orthonormal for $d' = d_{\mathcal{T}}$. Thus, the appearance of $\hat{\mathbf{U}}$ in the pseudoinverse is inconsequential in an analysis of the effect of compressing prior to regression.

To simplify the analysis one step further, we assume that our test set is a minimal core set $\mathcal{Q}$. Therefore, random projections are applied on $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$ and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$ matrices. The projections from over-complete test sets with rank bigger than $|\mathcal{Q}|$ down to $d_{\mathcal{T}}$ dimensions can be achieved by first projecting to size $|\mathcal{Q}|$ and then projecting from $|\mathcal{Q}|$ to $d_{\mathcal{T}}$. By the results of Section 4.1, this first projection leads to a consistent model, i.e. a model that is a linear transform of the model learned directly from $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$ and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$ matrices, since $\hat{\mathbf{U}}^\top \boldsymbol{\Phi}_{\mathcal{T}} \mathbf{R}$ is invertible with probability 1 when the projected dimension is equal to $|\mathcal{Q}|$ (Boots et al., 2010). The assumption that we work with the $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$ and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$ matrices directly (as apposed to invertible transforms of them) simplifies the analysis below in that we can elucidate our sparsity assumptions etc. directly in terms of the minimal core set of tests instead of random linear functions of tests in the minimal core set. This assumption is mild in that we could work with these random invertible linear transforms and discuss the discrepancy between a "random" TPSR (i.e., a TPSR defined via a random linear transform) and a compressed version of this "random" TPSR, and this discussion would be analogous to that which is provided below, albeit with more cumbersome and unnecessarily complex derivations. The assumption that we work with the minimal core set of tests simply allows for a more interpretable and less cluttered analysis.

Now, we define

$$\mathbf{B}_{ao} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,\mathcal{H}} (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger, \quad \boldsymbol{\beta}_\infty = (\boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}})^\dagger \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{H}}.$$

Since $\mathcal{Q}$ is a minimal core set of tests, the above is a TPSR representation (Boots et al., 2010; Rosencrantz et al., 2004). Assume we have enough histories in $\mathcal{H}$ such that matrices are full rank. Defining $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ to be the vectors containing the joint probabilities of all tests in the minimal core set and a fixed history h, we have that (by the linearity of PSRs)

$$\forall h : \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} = \mathbf{B}_{ao} \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}, \quad \boldsymbol{\mathcal{P}}_h = \boldsymbol{\beta}_\infty^\top \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}.$$

One can thus think of finding the $\mathbf{B}_{ao}$ and $\boldsymbol{\beta}_\infty$ parameters as regression problems, having the estimates of $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$s as noisy input features. We also have noisy observations of the outputs $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ and $\boldsymbol{\mathcal{P}}_h$. Since the sample set suffers from the error in variables problem (i.e., is noisy both on the input and output values) direct regression in the original space might result in large estimation error. Therefore, we apply random projections, reducing the

10. As noted in Section 7.1.3 it is sometimes beneficial to use $d' < d_{\mathcal{T}}$ and/or discard very small singular values in order to improve the numerical stability of computing inverses during learning. However, this issue of numerical stability is orthogonal to the analysis presented in this section.

estimation error (variance) at the cost of a controlled approximation error (bias). And we get the added benefit that working in the compressed space also helps with the computation complexity of the algorithm.

Note that there is an inherent difference between our work and the TPSR framework. In TPSR, one seeks to find concise linear transformations of the observation matrices, whereas CPSR seeks to find good approximations in a compressed space (which cannot be linearly transformed to the original model). That said, approximate variants of the TPSR learning algorithm have been analyzed from the perspective of compressed regression (albeit without appealing to the compressed sensing framework we employ) (Kulesza et al., 2014; Boots and Gordon, 2010). For example, Kulesza et al. (2014) analyze low-rank TPSR models where the rank of the learned model is made less than $|\mathcal{Q}|$ by removing the least significant singular vectors of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$. We reiterate, however, that these analyses are distinct from the analysis presented in this work, as we analyze low-rank models where the rank is reduced via random projection-based compression (not by removing least-significant singular vectors). The following sections provide an analysis of the error induced by this compression and how the error propagates through the application of several compressed operators.

4.2.2 Error of One Step Regression

When the size of the projections is smaller than the size of the minimal core set, we have the implicit regression performed on a compressed representation. The update operators are thus the result of compressed ordinary least-squares regression (COLS). There are several bounds on the excess risk of regression in compressed spaces (Maillard and Munos, 2009, 2012; Fard et al., 2012, 2013). In this section, we assume the existence of a generic upper bound for the error of COLS.

Assume we have a target function $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b(\mathbf{x})$ where $\mathbf{x}$ is in a k-sparse D-dimensional space, and $b(\cdot)$ is the bias of the linear fit. We observe an i.i.d. sample set $\{(\mathbf{x}_i, f(\mathbf{x}_i) + \eta_i)\}_{i=1}^n$, where η_i's are independent zero-mean noise terms for which the maximum variance is bounded by σ_η^2, and $\mathbf{x}_i$'s are sampled from a distribution ρ. Let $\hat{f}_d(\mathbf{x})$ be the compressed least-squares solution on this sample with a random projection of size d. That is, $\hat{f}_d(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\Phi}^\top \hat{\mathbf{w}}_d$ with

$$\hat{\mathbf{w}}_d = (\boldsymbol{\Phi}\mathbf{X}^\top \mathbf{X}\boldsymbol{\Phi}^\top)^{-1}(\boldsymbol{\Phi}\mathbf{X}^\top)\mathbf{y} \in \mathbb{R}^d,$$

where $\mathbf{X} \in \mathbb{R}^{n \times D}$ is a design matrix, $\mathbf{y} \in \mathbb{R}^n$ is a vector of training targets, and $\boldsymbol{\Phi} \in \mathbb{R}^{d \times D}$ is a random projection matrix. Define $\|g(\mathbf{x})\|_{\rho(\mathbf{x})} = \sqrt{\mathbb{E}_{\mathbf{x}\sim\rho}(g(\mathbf{x}))^2}$ to be the weighted L^2 norm under the sampling distribution. We assume the existence of a generic upper bound function ϵ, such that with probability no less than $1 - \delta$

$$\|f(\mathbf{x}) - \hat{f}_d(\mathbf{x})\|_{\rho(\mathbf{x})} \leq \epsilon(n, D, d, \|\mathbf{w}\|^2, \|\mathbf{x}\|^2_{\rho(\mathbf{x})}, \|b(\mathbf{x})\|^2_{\rho(\mathbf{x})}, \sigma_\eta^2, \delta). \tag{29}$$

The effectiveness of the compressed regression is largely dependent on how the $\|\mathbf{w}\|\|\mathbf{x}\|_{\rho(\mathbf{x})}$ term behaves compared to the norm of the target values. We refer the reader to the discussions in Maillard and Munos (2009) and Maillard and Munos (2012) on the $\|\mathbf{w}\|\|\mathbf{x}\|_{\rho(\mathbf{x})}$ term. In the case of working with PSRs, we have that the probability of the tests are often highly correlated. Using this property, we will show that $\|\mathbf{w}\|^2$ can be bounded well below its size.

In order to use these bounds, we need to consider the sparsity assumptions in our compressed PSR framework. We formalize the inherent sparsity, discussed in previous sections, as follows: For all h, $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ and $\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ are k-sparse. Given that the empirical estimates of zero elements in these vectors are not noisy, for $\Delta_x = \hat{\mathbf{P}}_{\mathcal{Q},h} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ we have that Δ_x is k-sparse (with a similar argument for $\Delta_y = \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,h} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$).

To simplify the analysis, in this section we define our $\mathbf{C}_{ao}$ matrices to be slightly different from the ones used in the described algorithm. By forcing the diagonal entries to be 0, we avoid using the ith feature for the ith regression. This removes any dependence between the projection and the target weights and simplifies the discussion. Since we are working with random compressed features as input, all of the features have similar correlation with the output, and thus removing one of them changes the error of the regression by a factor of $O(1/d)$. We can nevertheless change the algorithm to use this modified version of the regression so that the analysis stays sound.

The following theorem bounds the error of a one step update using the compressed operators. We use i.i.d. normal random projection for simplicity. The error bounds for other types of random projections should be similar.[11] Let $[\mathbf{A}]_{-i,*}$ be matrix $\mathbf{A}$ with the ith row removed. We have the following:

Theorem 1 *Let $\mathcal{H}$ be a large collection of sampled histories according to ρ, and let $\boldsymbol{\Phi}^{d\times|\mathcal{Q}|}$ be an i.i.d. normal random projection: $\boldsymbol{\Phi}_{ij} \sim \mathcal{N}(0, 1/d)$. We observe noisy estimate $\hat{\mathbf{P}}_{\mathcal{Q},h} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},h} + \Delta_x$ of input and $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,h} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + \Delta_y$ of the output, where elements of Δ_x and Δ_y are independent zero-mean random variables with maximum variance σ_x^2 and σ_y^2 respectively. Let $\sigma_1^2 \ldots \sigma_{|\mathcal{Q}|}^2$ be the decreasing eigenvalues of $E_{\rho(h)}[\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}^\top]$. Choose $1 \le m \le |\mathcal{Q}|$ such that $\sigma_m^2 \le 1$ and define $\nu = \sum_{i=m+1}^{|\mathcal{Q}|} \sigma_i^2$. For $1 \le i \le d$, define*

$$\mathbf{u}_i = \boldsymbol{\Phi}_i \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}} (\boldsymbol{\Phi}_{-i} \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}})^\dagger.$$

Define $\mathbf{C}_{ao}$ to be a $d \times d$ matrix such that

$$(\mathbf{C}_{ao})_i = [\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,i-1}, 0, \mathbf{u}_{i,i}, \mathbf{u}_{i,i+1}, \ldots, \mathbf{u}_{i,d-1}].$$

Then with probability no less than $1 - \delta$ we have

$$\|\mathbf{C}_{ao}(\boldsymbol{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - \boldsymbol{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\|_{\rho(h)} \le \sqrt{d}\epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/4d), \tag{30}$$

where

$$w^2 = \|\mathbf{B}_{ao}\|^2 (m + 4\sqrt{m}\ln(4d/\delta)), \tag{31}$$

$$x^2 = \|\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}\|_{\rho(h)}^2, \tag{32}$$

$$b^2 = \nu + 4\sqrt{\nu}\ln(4d/\delta), \tag{33}$$

$$\sigma_\eta^2 = \frac{4k\ln(4|\mathcal{Q}|/\delta)}{d}\sigma_y^2 + w^2\sigma_x^2. \tag{34}$$

11. The core modifications necessary are analogous to those used made in Achlioptas (2001) to adapt the Johnson-Lindenstrauss lemma to more general random matrices.

The proof is included in the appendix. The main idea of the theorem is to use the dependence and sparsity of the features to tighten the bound on the error of compressed regression. When most of the variation in the PSR state can be explained using m linear observations, we can substitute the $\mathbf{\Phi}_i \mathbf{B}_{ao}$ target weights having norm $O(\sqrt{|\mathcal{Q}|})$, with a linear approximation having much smaller norm $O(\sqrt{m})$, at the expense of a small bias b. The theorem also describes the overall noise combining the effects of Δ_x and Δ_y.

Theorem 1 has three main implications. One is that the complexity of the compressed regression depends on how fast the eigenvalues drop for the minimal core set covariance matrix. If the eigenvalues drop exponentially fast, as is observed empirically in our experiments, we can guarantee a smaller regression error. Second, if the projection size is of order $O(k \ln |\mathcal{Q}|)$ we can control the variance of the combined noise term. Third, if we use the sparse COLS bound of Fard et al. (2012, 2013), we can can show that regression of size $O(k \ln |\mathcal{Q}|)$ should be enough to decrease the overall estimation error at the expense of a controlled bias.

The following corollary follows immediately from Theorem 1 by union bounding over all action-observation pairs.

Corollary 2 *Using the assumptions of Theorem 1, with probability no less than $1-\delta$ we have, for all $a \in \mathcal{A}$ and $o \in \mathcal{O}$,*

$$\|\mathbf{C}_{ao}(\mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - \mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\|_{\rho(h)} \leq \sqrt{d}\epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/(4d|\mathcal{A}||\mathcal{O}|)),$$

where $w^2 = \max_{ao} \|\mathbf{B}_{ao}\|^2(m + 4\sqrt{m}\ln(4d/\delta))$, and other factors are as defined in Theorem 1.

4.2.3 Error of the Compressed Normalizer

The $\mathbf{c}_\infty$ operator is the normalization operator for the compressed space. Therefore, for any history h, $\mathbf{c}_\infty^T \mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ should equal $\boldsymbol{\mathcal{P}}_h$. The following theorem provides a bound over the error of such a prediction:

Theorem 3 *Let $\mathcal{H}$ be a large collection of sampled histories according to ρ. We observe noisy estimate $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}} = \boldsymbol{\mathcal{P}}_{\mathcal{Q},\mathcal{H}} + \Delta_x$ of input and $\boldsymbol{\mathcal{P}}_{\mathcal{H}} = \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{H}} + \Delta_z$ of the output, where elements of Δ_x and Δ_z are independent zero-mean random variables with maximum variance σ_x^2 and σ_z^2 respectively. Define $\mathbf{c}_\infty = (\mathbf{\Phi}_i \hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}})^\dagger \boldsymbol{\mathcal{P}}_{\mathcal{H}}$. Then with probability no less than $1-\delta$ we have*

$$\left\| \mathbf{c}_\infty^\top (\mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - \boldsymbol{\mathcal{P}}_h \right\|_{\rho(h)} \leq \epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, \|\boldsymbol{\beta}_\infty\|^2, \|\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}\|^2_{\rho(\mathbf{h})}, 0, \sigma_\infty^2, \delta),$$

where we define effective noise variance $\sigma_\infty^2 = \sigma_z^2 + \sigma_x^2 \|\boldsymbol{\beta}_\infty\|^2$.

The proof is included in the appendix.

4.2.4 Error Propagation

Once we have the one step errors of compressed operators, we can analyse the propagation of errors as we concatenate the operators. Define $o_{1:n} = o_1 o_2 \dots o_n$ (and similarly for $a_{1:n}$

and $[ao]_{1:n}$). We would like to bound the error between $\mathbb{P}(o_{1:n}||a_{1:n})$ and our prediction $\mathbf{c}_\infty \mathbf{C}_{a_n o_n} \mathbf{C}_{a_{n-1} o_{n-1}} \dots \mathbf{C}_{a_1 o_1} \mathbf{c}_1$.

Since the theorems in the previous sections were in terms of a fixed measure ρ, we have to make distributional assumptions to simplify the derivations. Assume that we fit our model using samples $h \sim \rho$, imposing a distribution $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h} \sim \mu$. Note that as we increase the size of a history h, the norm of $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ becomes smaller. We make the assumption that for all $1 \leq t \leq n$, for a history $[ao]_{1:t} \sim \rho_t$, the implied $\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:t}}$ is sampled from a scaled version of μ (i.e., $\frac{1}{s_t}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:t}} \sim \mu$). Therefore $\|f(\boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho_t(h)} = \|f(s_t \boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho(h)}$.

Theorem 4 *Let ϵ and ϵ_∞ be the bounds of Corollary 2 and Theorem 3 respectively, for a sample $\mathcal{H}$ according to ρ and failure probability $\delta/2$. Let ρ_n and its marginals $\rho_{n-1} \dots \rho_1$, be distributions over histories of size $n, n-1, \dots 1$ respectively, such that $\|f(\boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho_t(h)} = \|f(s_t \boldsymbol{\mathcal{P}}_{\mathcal{Q},h})\|_{\rho(h)}$ for all measurable f. With probability $1 - \delta$*

$$\left\|\mathbf{c}_\infty \mathbf{C}_{a_n o_n} \mathbf{C}_{a_{n-1} o_{n-1}} \dots \mathbf{C}_{a_1 o_1} \mathbf{c}_1 - \mathbb{P}(o_{1:n}||a_{1:n})\right\|_{\rho_n} \leq \epsilon_\infty s_n + \|\mathbf{c}_\infty\| \epsilon \sum_{t=1}^{n-1} s_t c^{n-t-1},$$

where $c = \max_{a,o} \|\mathbf{C}_{ao}\|$.

The proof is included in the appendix. Note that s_t is exponentially decreasing in t (because longer tests are less probable). The norm of the update operators are expected to be less than 1 (as they shrink the vector of test probabilities). Combining these two, we expect the summation in the bound of Theorem 4 to be over a small exponential function of n.

5. Planning with CPSRs

The learning algorithm presented in Section 3.2 facilitates the construction of accurate predictive models in large complex partially observable domains. In this section, we outline how to plan (near)-optimal sequences of actions using such a learned model. The planning approach we employ was first proposed by Ong et al. (2012). In essence, the approach substitutes a predictive state in place of an observable state in the standard fitted-Q learning algorithm of Ernst et al. (2005).

Unlike point-based value-iteration PSR (PBVI-PSR) planning algorithms, the theoretical convergence of the fitted-Q algorithm does not require that the PSR correspond to a finite-dimensional POMDP. That is, existing error-bounds for PBVI-PSR require that the PSRs used in planning correspond to some finite-dimensional POMDP (Izadi and Precup, 2008), whereas in general PSRs may have no corresponding finite-dimensional POMDP (Denis and Esposito, 2008).[12] In contrast, the fitted-Q approach only requires that the input state-space be sufficient to describe the system, and PSRs satisfy this requirement, meaning that the convergence results for fitted-Q carry over to the PSR setting (when an

12. It is worth noting, however, that the PSR-PBVI error bounds could possibly be modified to alleviate this issue and that PBVI-PSR algorithms have been employed with considerable empirical success (Boots et al., 2010; Izadi and Precup, 2008).

exact PSR model is used) (Ernst et al., 2005).[13] Moreover, the fitted-Q approach does not explicitly require learning a model of rewards prior to the application of the planning algorithm (i.e., the reward model is captured only through the Q-function). We found this to be preferable to explicitly modelling the immediate rewards as a function of the CPSR states prior to planning, as such an explicit model introduces an extra (and unnecessary) level of approximation. In what follows, we briefly review the fitted-Q approach and provide a high-level description of our planning algorithm.

5.1 Fitted-Q with CPSRs

Algorithm 1: Fitted-Q with CPSR

Inputs: A set $\mathcal{D}$ of tuples of the form $(\mathbf{c}_t, a_t, r_t, \mathbf{c}_{t+1})$ constructed using a CPSR model, where r_t is a numerical reward; $\mathcal{R}$, a regression algorithm; γ, a discount factor; and T, a stopping condition

Outputs: A policy π

1: $k \leftarrow 0$
2: Set $\hat{Q}_k(\mathbf{c}_t, a) = 0 \ \forall a \in \mathcal{A}$ and all possible $\mathbf{c}_t$
3: **repeat**
4: $\quad k \leftarrow k + 1$
5: $\quad$ Build training set, $\mathbb{T} = \{(y^l, i^l), l = 1, ..., |\mathcal{D}|\}$ where: $i^l = (\mathbf{c}_t^l, a_t^l)$ and $y^l = r_t^l + \gamma \max_a \hat{Q}_{k-1}(\mathbf{c}_{t+1}^l, a)$
6: $\quad$ Apply $\mathcal{R}$ to approximate $\hat{Q}_k$ from $\mathbb{T}$
7: **until** T is met

output π, where $\pi(\mathbf{c}_t) = \text{argmax}_a\{\hat{Q}_k(\mathbf{c}_t, a)\}$

As stated above, fitted-Q with PSRs is analogous to the MDP case, with the predictive state taking the place of the MDP state in Algorithm 1. The algorithm iteratively builds more and more accurate approximations of the Q-function, which in our case maps predictive states and actions to expected returns. In this work, the *Extra-Trees* algorithm is used as the base regression algorithm (Geurts et al., 2006), as it is a non-linear function approximator for which the fitted-Q convergence results hold (Ernst et al., 2005). For T, the termination condition, we use an iteration limit (instead of an ϵ convergence condition), as this allows for more accurate predictions of runtimes.

Letting $\Psi(T)$ be the expected number of iterations under stopping condition T and assuming that the splitting procedure for nodes in the Extra-Trees algorithm takes constant time, the computational complexity of this fitted-Q approach is (recalling the definitions of Section 3.2)

$$O\left(\Psi(T) \times L|Z|log\left(L|Z|\right)\right), \tag{35}$$

13. The error bounds for PSR-PBVI also require that an exact model is known. In general, current theoretical results on PSR planning ignore the impact of estimation and/or approximation errors incurred during model-learning, though empirical analyses (e.g., the work of Boots et al. (2010) and Section 6 of this paper) suggest that the impact of such errors is small.

which is a factor $\Psi(T) \times log(L|Z|)$ greater than the complexity for the model-learning algorithm of Section 3.2. In practice, we found Algorithm 1 to be several orders of magnitude slower than the CPSR learning algorithm.

5.2 Combined Learning and Planning

Algorithm 2 specifies how CPSR model-learning and the fitted-Q planning algorithm are combined at a high level. This general specification permits a variety of sampling and Q-function approximation strategies. Specifically, it permits pure unbiased random sampling, interleaving exploration and exploitation phases, or even the drawing of samples from some arbitrary (e.g., expert) policy. Of course, if non-blind policies are used then the sample distribution becomes biased (i.e., the samples are no longer i.i.d.), and the analysis of Section 4 no longer holds.

Also note that the number of iterations used by the learner and planner need not be identical. More specifically, more samples may be used to learn the CPSR model than are used in planning. This is a pragmatic specification, as the CPSR learning algorithm can efficiently accommodate orders of magnitude larger sample sets than the fitted-Q planner (by Equations 19 and 35).

Algorithm 2: Combined learning and planning

Inputs: π_s, a sampling policy; N, the number of sampling iterations; I_m, the number of trajectories to use in learning; and I_p, the number of trajectories to use in planning ($I_m \geq I_p$)

Outputs: A CPSR model, $\mathbf{C}$ and policy π

$\mathcal{D}_0 \leftarrow \emptyset$
Initialize the CPSR model, $\mathbf{C}$
for i=1 to N **do**
  Sample I_m trajectories, Z_i, using π_s
  Update $\mathbf{C}$ using Z_i
  Sub-sample I_p trajectories from Z_i and use $\mathbf{C}$ to construct a tuple-set $\mathcal{D}_i$
  $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \mathcal{D}_{i-1}$
  Apply Algorithm 1 with $\mathcal{D}_i$ to learn a policy, π_i
  [Optional] Update π_s (e.g., using π_i)
end for
output $\mathbf{C}$ and π_N

---

6. Empirical Results

We examine empirical results pertaining to both the model quality of compressed models and the proficiency of model-based planning. The goal of this analysis in the model-quality setting is to elucidate (1) the empirical cost (in terms of prediction accuracy) of performing compression (if any), (2) the compute-time reduction engendered by the use of compression, and (3) the impact of the implicit regularization induced by performing compression. We also provide model-quality results explicitly comparing prediction performance when histo-

ries are compressed versus uncompressed, showing that history compression has a negligible effect empirically (and justifying the simplifying assumption that $\mathbf{\Phi}_{\mathcal{H}}^{\top}\mathbf{\Phi}_{\mathcal{H}} = \mathbf{I}$ in Section 4).

In the planning setting, we again seek to elucidate the empirical impact of performing compression, and we do so using three different partially observable domains. First, we use a simple synthetic robot navigation domain (identical to that used in the model-quality experiments) to compare the planning performance of agents trained with CPSR models, agents trained with uncompressed TPSR models, and memoryless (model-free) agents, which serve as a baseline. Next, we examine a massive partially observable domain that is intractable for classic POMDP-based approaches, demonstrating how the use of compression facilitates learning and planning in settings where it would be otherwise intractable. We also provide a qualitative comparison to the Monte-Carlo AIXI algorithm (Veness et al., 2011), a related model-based reinforcement learning approach, using this domain. Lastly, we apply CPSR based learning and planning to the difficult real-world task of adaptive migratory management (Nicol et al., 2013). In this adaptive migratory management problem, a sequential decision-making agent must learn a model of how a certain bird species migrates and how their migration patterns are adversely affected by rising sea-levels (and must do so without prior domain-specific knowledge). Using this learned model the agent must determine an optimal policy for protecting different locations along the birds' migratory route so as to minimize population decline (Martin et al., 2007; Nicol et al., 2013). This difficult real-world domain, which builds upon hand-crafted simulators and ecological data sets (Iwamura, 2011; Nicol et al., 2013), demonstrates both the benefits of compression (in that it is computationally intractable for uncompressed TPSR) and the stark benefits of model-based planning over memoryless (model-free) planning.

6.1 Projection Matrices

In this analysis, we examine three different classes of random projection matrices: spherical, Rademacher, and hashed. The spherical projection matrices contain random Gaussian distributed entries and are identical to those used in Hamilton et al. (2013). The Rademacher are a related class of random matrices where each entry is an independent Rademacher variable; these matrices also satisfy the JL lemma (Baraniuk and Wakin, 2009) and can afford additional efficiencies with low level implementations that exploit the fact that only additions and subtractions are used in the matrix multiplications (this optimization is not used here) (Achlioptas, 2001). The hashed random projection matrices induce a feature-mapping analogous to random hashing; each column of the random projection matrix has a 1 in a random position and the other entries are zero. These random hashing matrices do not directly satisfy the JL lemma, but they have been shown to preserve certain kernel-functions and perform extremely well in practice (Weinberger et al., 2009; Shi et al., 2009).

6.2 Domains

The domains used are based upon previous work on planning with PSRs and on model-based reinforcement learning in large, complex partially observable domains.

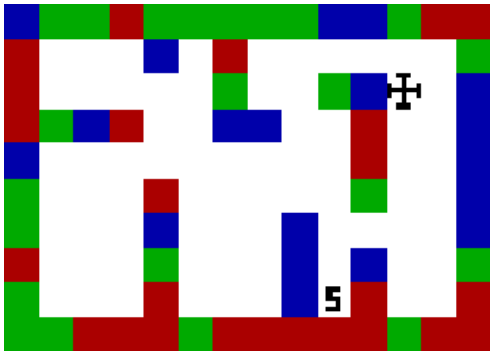


Figure 1: Graphical depiction of *ColoredGridWorld*. The **S** denotes the start position and the target denotes the goal.

6.2.1 ColoredGridWorld

The first domain, *ColoredGridWorld*, is conceptually similar to the simulated robot navigation domains commonly used in the PSR literature and is a direct extension of the *GridWorld* domain used in Hamilton et al. (2013) and Ong et al. (2012). The environment is a 47-state maze with coloured walls. The agent must navigate from a fixed start state to a fixed goal state using only aliased local observations. The action space consists of moves anywhere in the four cardinal directions (moving into walls produces no effect). To simulate noise in the agent's actuators, actions fail with probability 0.2, and if this occurs, the agent moves randomly in a direction orthogonal to that which was specified. The observation space consists of whether or not the agent can see coloured walls in any of the 4 cardinal directions (one observation per wall). There are three possible colors, so there are 3 possible observations per wall and thus 81 possible observations in total. A reward of 1 is returned at the goal state (resetting the environment), and no other states emit rewards.

Though simple, this domain is quintessentially partially observable in that it is impossible to learn how to reach the goal without incorporating memory. Moreover, the added complication of coloured walls exponentially increases the cardinality of the observation space, leading to many possible tests and histories. In essence, the agent cannot know a priori whether the colouring is pertinent to the problem, so it vastly complicates the learning problem.

6.2.2 Partially Observable PacMan

The second domain used is based upon the partially observable PacMan domain, denoted *PocMan*, first proposed by Silver and Veness (2010). It is an extremely large partially observable domain with on the order of 10^{56} states (Veness et al., 2011). The basic dynamics follow that of the video-game PacMan: an agent must navigate a maze-like environment starting from a fixed start-point, collecting food and avoiding coming in contact with any of four ghosts.

In this work, we examine two versions of the domain. The first version is a replica of the *PocMan* domain used by Veness et al. (2011) in their work on a Monte Carlo AIXI

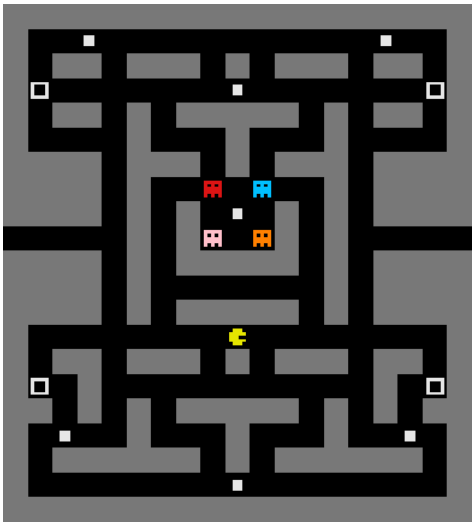


Figure 2: Graphical depiction of *S-PocMan*. The white dots denote food and the white annuli denote power-pills. The yellow PacMan figure denotes the fixed starting position

approximation. In the second version, which we call *S-PocMan*, we further complicate the environment by dropping the parts of the observation vector that allow the agent to sense in what direction food lies, and we sparsify the amount of food in the environment. In the original domain food was placed in each position with probability $\frac{1}{2}$; in *S-PocMan* there are only 7 pieces of food in total, each in a fixed position. The reason for examining this more difficult version of the domain is that, as summarized in Section 6.4, we found that a memoryless controller was able to perform extremely well on the original *PocMan*, achieving results approaching that of the AIXI algorithm. In other words, simply treating the original *PocMan* domain as if it were fully observable led to very good results. This seems to be due to the fact that the food rewards were plentiful and fully observable. In *S-PocMan* we make the problem more partially observable in order to demonstrate the usefulness of a model-based approach.

6.2.3 Adaptive Migratory Management

The last domain we examine is based upon the ecological task of *adaptive migratory management* (*AMM*). The specific goal of *AMM* is to use intervention to protect certain regions in a bird-species' migratory route. In this work, we focus on the Lesser Sand Plover, which is one of many species that uses the East-Asian-Australasian (EAA) migratory route. While migrating, the Lesser Sand Plover stop at staging sites where they feed on invertebrates and gather energy (Martin et al., 2007). These staging sites are located at intertidal mudflats that are especially susceptible to rising sea levels (Iwamura, 2011). The sites can be protected via intervention, but limited resources within the conservation community means that protection can only be implemented at a limited number of sites within a particular year.

By phrasing the task of protecting these intertidal areas as a sequential decision-making problem, the hope is to learn an optimal strategy for intervention.

In Nicol et al. (2013) the *AMM* problem is formalized, and a simulator based dataset is provided (for a number of species including the Lesser Sand Plover). At its core the simulator uses a network-flow model for the migratory routes augmented with handcrafted models for sea-level rises, population declines, and other relevant elements. See Nicol et al. (2013) for a complete description. In this work, we use data generated from the simulator, and we attempt to both learn a succinct model of the domain and optimize decisions using this learned model (i.e., we do not assume access to information contained within the internal simulator state).[14]

Formally, at each time point (which roughly corresponds to a year) the decision-making agent receives a vector of observations, where the first entry corresponds to the population level at the breeding site/node and the next three entries correspond to the protection levels at the three intertidal sites/nodes on the Lesser Sand Plover's migratory route. There are four discretized population levels and three protection levels, corresponding to protection against three increasing states of sea-level rise. There are thus 108 unique possible observations. At each time-step the decision-making agent must increase the protection level at one of the non-breeding nodes, and thus there are three possible actions at each time-step. (If the agent opts to protect a node which is already maximally protected then the action has no effect). Internal dynamics of the underlying system-model determine how protection levels decline over time, but none of this information is available to the agent. At the beginning of a simulation (i.e., in the fixed start-state) the protection and population levels are set to their minimal discretized values.

6.3 Model Quality Results

We examined the model quality of different CPSRs and an uncompressed TPSR on the *ColoredGridWorld* domain. Sample trajectories were generated using a simple ϵ-greedy exploration policy, where the non-random actions were determined by a policy learned via a memoryless controller. All models were set with $d' = 5$, where d' is final model dimension (from Section 3.2) set after performing SVD; however, singular values below a tolerance of 10^{-6} were also discarded. All tests, τ_i, with $|\tau_i| \leq 7$ were included in the estimation process (including longer length tests did not improve performance).[15] For the CPSR models, we set $d_{\mathcal{T}} = d_{\mathcal{H}}$, as preliminary experiments did not reveal any significant benefits to using $d_{\mathcal{T}} \neq d_{\mathcal{H}}$ and examined projection dimensions in the range $[25, 75]$. Only the best performing size (determined through cross-validation) is reported. All models used 10000 train trajectories (of max length 13) and were evaluated with 10000 trajectories. The PacMan-style domains and the *AMM* domain were not examined in this model-quality context as naive TPSRs

14. Note that for the benchmark results presented in Nicol et al. (2013), they use knowledge of the underlying simulator state and cast the planning problem in the POMDP framework, while in this work we solve both the learning and planning problems (rather than just the planning problem).
15. If a particular test was never encountered in the training data, however, it was discarded, as such tests lead to singularities in the observable matrices.

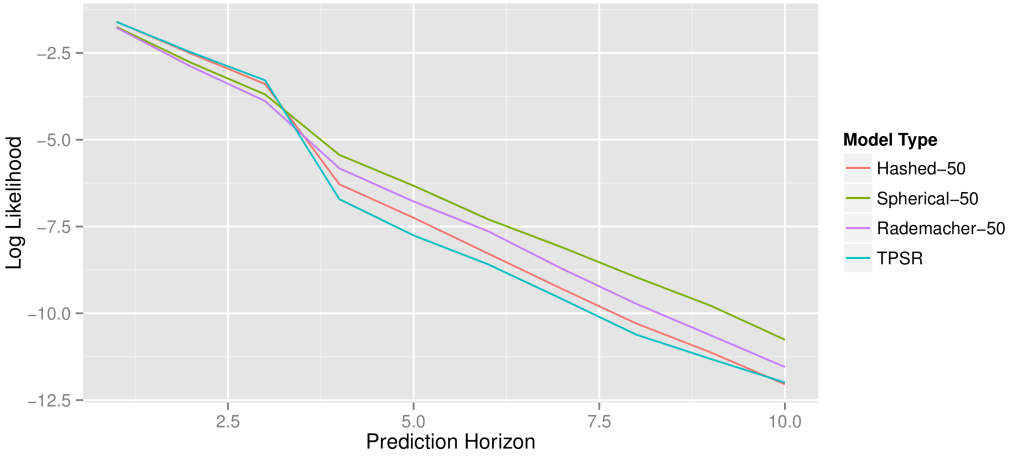


Figure 3: Model-quality results on the *ColoredGridWorld* domain. Plot shows the log-likelihood of the test data given the different models as the prediction horizon is increased. The numbers adjacent to the CPSR projection types correspond to the compressed dimension used. 95% confidence interval error bars are too small to be visible.

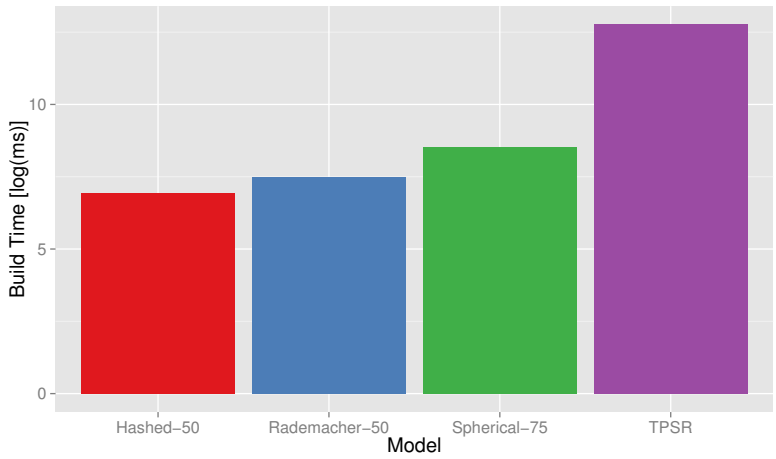


Figure 4: Model build times (on a log-scale) for the different model types on the *Colored-GridWorld* domain. Compressed dimension sizes are listed next to the model names. Times do not include time taken to build the training set. 95% confidence interval error bars are too small to be visible.

exhausted memory limits when tests of length longer than 1 were used, making a rigorous comparison is infeasible.[16]

16. Experiments were run on a machine with a 8-core 3.2 GHz Intel Xeon processor (x64 architecture) and 8GB of RAM.

Figure 3 plots the average log-likelihood of the models as the prediction horizon (i.e., length of the sequences to predict) is increased. The log-likelihood for a single sequence is computed by taking the logarithm of the probability obtained via (1), and this likelihood is averaged over all sequences in the test set. From this figure, we see that the compressed models are not only competitive with the uncompressed TPSR, they actually outperform TPSR at longer prediction horizons. We conjecture that this is due to the regularization induced by the use of random projections. Figure 4 plots the build times for the different models, showing that the compressed models can be built in a fraction of the time required to build the uncompressed TPSR.

Figure 5 shows a focused experiment examining the impact of compressing histories, compared to only compressing tests as was done in Hamilton et al. (2013). These results show $\log(\mathcal{L}(\theta)) - \log(\mathcal{L}(\theta_{HC}))$, the difference between the model-likelihood for a model where histories are not compressed (θ) and where histories are compressed (θ_{HC}). Both the predictive models are constructed using spherical projection matrices and using (identical) samples generated from the *ColoredGridWorld* domain (with the experimental set-up described above). As is evidenced in the plot, there is only a small difference in likelihood between the two models (cf. the likelihood difference seen in Figure 3), and in fact, the model with compressed histories does slightly better for the first few time steps.

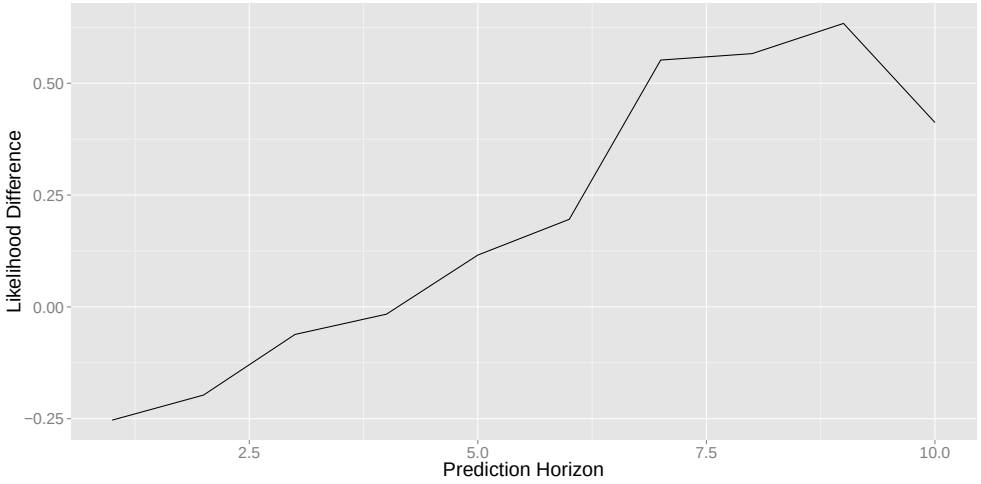


Figure 5: Difference in log-likelihood between a model where histories are not compressed and a model where histories are compressed.

6.4 Planning Results

Next, we apply the full learning and planning approach (Algorithm 2) to the domains *ColoredGridWorld, PocMan*, *S-PocMan*, and *AMM*.

In all experiments, we used 10000 random sampled trajectories to build the models and again used $d_{\mathcal{T}} = d_{\mathcal{H}}$. For planning, we used $I_p = 1000$ with $N = 1$ and a random sampling strategy; this represents the standard unbiased batch-learning setting (Section 7 discusses

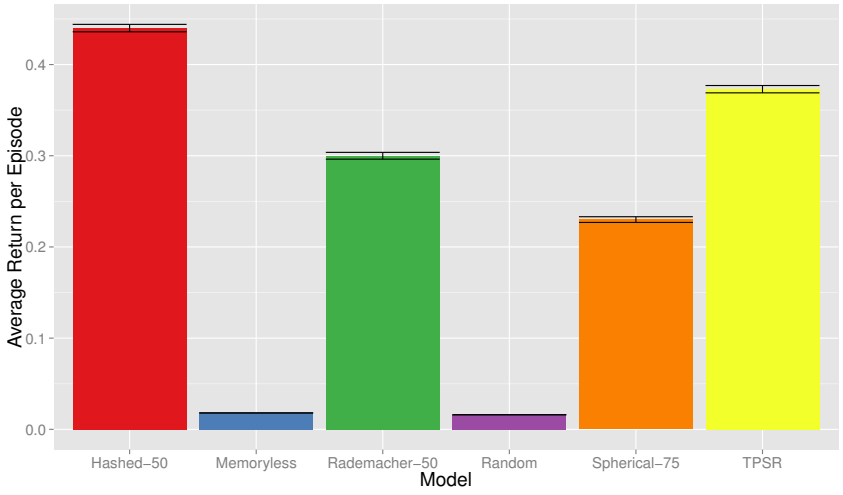


Figure 6: Average return per episode achieved in the *ColoredGridWorld* domain using different models and the baselines. Compressed dimension sizes are listed next to the model names. 95% confidence interval error bars are shown.

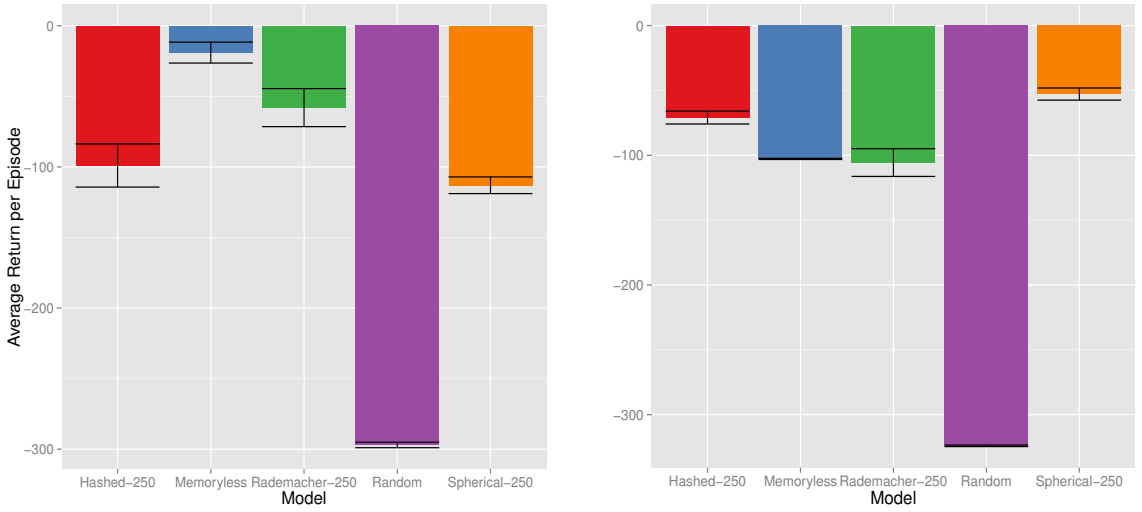
the possibility of using more complex sampling strategies). And for the fitted-Q algorithm, we used 100 fitted-Q iterations, one *Extra-Tree* ensemble of 25 trees per action, and the default settings for the *Extra-Trees* (Geurts et al., 2006). As a baseline, we examined the performance of a memoryless controller on the domains. This controller is analogous to treating the domains as fully observable and running the standard fitted-Q algorithm of Ernst et al. (2005). In order to achieve a fair comparison, the memoryless controller is permitted to use samples that would otherwise be used for model-learning in order to refine its policy (i.e., the memoryless baseline uses the same total number of samples in the experiments as the model-based methods). The use of this baseline is not arbitrary, as its success provides an empirical measure of how partially observable a domain is with respect to planning; if a domain is easily solved by the memoryless controller then it is nearly fully observable in that immediate observations are sufficient for determining near-optimal plans. We also used a simple random planner which selects actions uniformly randomly as a second baseline.

6.4.1 ColoredGridWorld

For *ColoredGridWorld*, the models examined were identical to those described in the model quality experiments above. A discount factor of $\gamma = 0.99$ was used for this domain.

Figure 6 details the performance of the different algorithms on the *ColoredGridWorld* domain. For this domain, the hashed CPSR algorithm achieved the best performance while the TPSR algorithm performed second-best. All the PSR-based approaches vastly outperformed the memoryless-controller baseline. This is expected, as the *ColoredGridWorld* problem is strongly partially observable.

6.4.2 Partially Observable PacMan



(a) *PocMan* (b) *S-PocMan*

Figure 7: Average return per episode achieved in the *PocMan* (a) and *S-PocMan* (b) domains using different models and the baselines. Compressed dimension sizes are listed next to the model names. 95% confidence interval error bars are shown.

For both *PocMan* and *S-PocMan*, we set $d' = 25$ and examined compressed dimensions in the range $[250, 500]$ (selecting only the top performer via a validation set); no TPSR models were used on these domains, as their construction exhausted the memory capacity of the machine used. Following Veness et al. (2011), for these domains we use $\gamma = 0.99999$ as a discount factor.

Figure 7 details the performance of the CPSR algorithms on the *PocMan* and *S-PocMan* domains. In these domains, we see a much smaller performance gap between the CPSR approaches and the memoryless baseline. In fact, in the *PocMan* domain, the memoryless controller is the top-performer. This demonstrates, first and foremost, that the *PocMan* domain is not strongly partially observable. Though the observations do not fully determine the agent's state, the immediate rewards available to an agent (with the exception of reward for eating the power pill and catching a ghost) are discernible through the observation vector (e.g., the agent can see locally where food is). Thus, the memoryless controller is able to formulate successful plans despite the fact that is treating the domain as if it were fully observable. Moreover, a qualitative comparison with the Monte-Carlo AIXI approximation (Veness et al., 2011) reveals that the quality of the memoryless controller's plans are actually quite good. In that work, they use a slightly different optimization criteria of optimizing for average transition reward, and with on the order of 50000 transitions they achieve an average transition reward in the range $[-1, 1]$ (depending on parameter settings). With on the order of 250000 transitions they achieve an average transition reward in the range $[1, 1.5]$. In this work, the memoryless controller achieves an average transition reward of

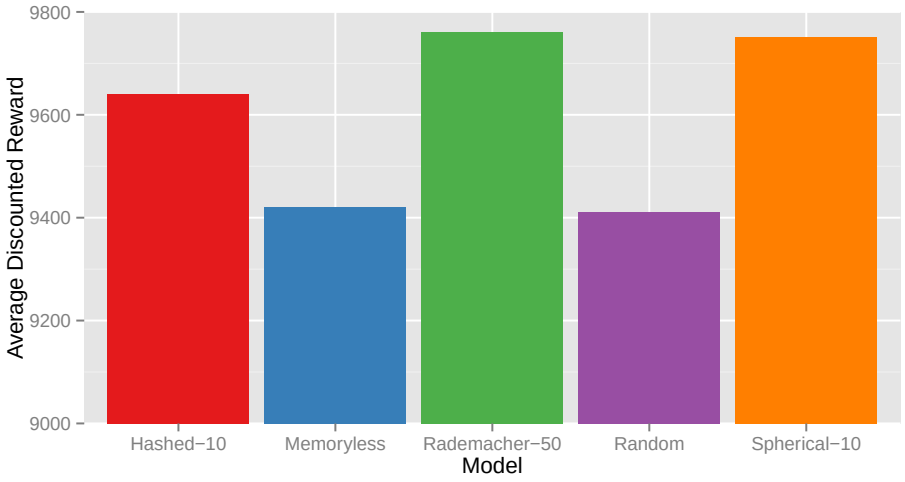


Figure 8: Average discounted reward per episode (i.e., average return per episode) achieved in the *AMM* domain using different methods over 100000 test episodes (each of length 50). The numbers beside the CPSR method names denote the projected dimension size. 95% confidence intervals are too small to be visible.

-0.2 (despite the fact that it is actually optimizing for average return per episode), and it is thus, competitive given the same magnitude of samples, as approximately 50000 transitions were used in this work. It is also important to note that PSR-type models may be combined with memoryless controllers as memory PSRs (described in Section 7.2), and so it should be possible to boost the performance of the CPSR models to match that of the memoryless controller in that way.

Importantly, in *S-Pocman* where part of the observation vector is dropped and the rewards are sparsified, we see that the top-performer is again a CPSR based model (which in this case uses spherical projections). This matches expectations since the food-rewards are no longer fully discernible from the observation vector, and thus the domain is significantly less observable. It is also worth noting that building naive TPSRs (without compression or domain-specific feature selection) is infeasible computationally in these PacMan-inspired domains, and thus the use of a PSR-based reinforcement learning agent (via the compression techniques used) in these domains is a considerable advancement.

A final observation is that the performance is quite sensitive to the choice of projection matrices in these results. For example, in the *S-PocMan* domain, the Rademacher projections perform no better than the memoryless baseline, whereas for *PocMan* the Rademacher outperforms the other projection methods. The exact cause of this performance change is unclear. Nevertheless, this highlights the importance of evaluating different projection techniques when applying this algorithm in practice.

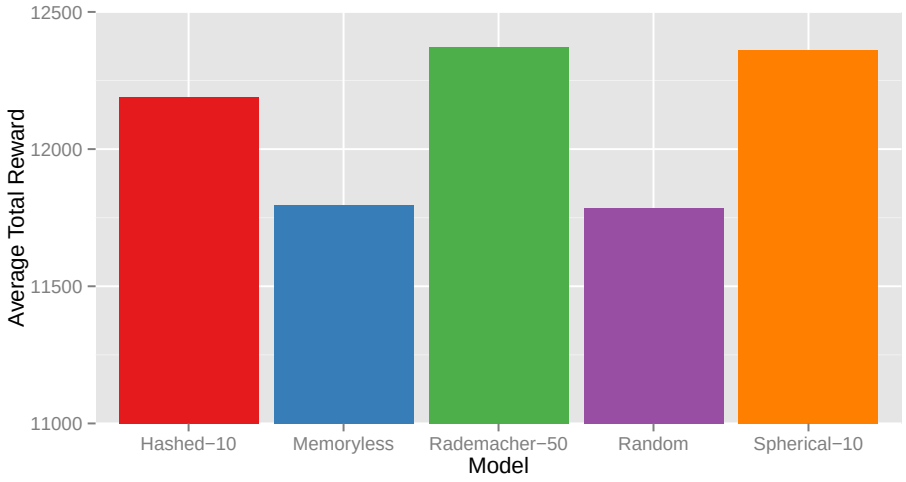


Figure 9: Average total (undiscounted) reward per episode achieved in the *AMM* domain using different methods over 100000 test episodes (each of length 50). The numbers beside the CPSR method names denote the projected dimension size. 95% confidence intervals are too small to be visible.

6.4.3 Adaptive Migratory Management

We used a discount factor of $\gamma = 0.99$ for the *AMM* domain. For model-learning, we set $d' = 10$ and examined compressed dimension in the range $[10, 100]$. The trajectories used during learning are all of maximum length 50 (the simulation may terminate earlier if all the birds perish). Note that since the *AMM* domain is non-stationary (Nicol et al., 2013), the model-learning algorithms must incorporate histories of length up to 50 (i.e., the entire trajectory) (Boots et al., 2010), making the history dimension extremely large (i.e., ≈ 100000) and making uncompressed PSR learning infeasible. Tests up to length 4 were used for this task.

The results obtained are summarized in Figures 8 and 9. Figure 8 shows the average sum of discounted rewards obtained using each method while Figure 9 shows the average total (i.e., undiscounted) sum of rewards obtained by each method. Both these test metrics are included as the discounted return is what the fitted-Q algorithm optimizes for while the average total return is important from an intuitive perspective in that ecological conservationists do not necessarily discount the future. (Note, however, that the discount factor is necessary algorithmically for convergence since this domain technically has an infinite horizon).

Clearly, the CPSR methods are the top-performers with respect to both metrics. In fact, the memoryless baseline does no better than random. We also note that returns achieved by all methods are quite high. The cause of the high return and the fact that the memoryless does no better than random are closely related. Specifically, in the domain all actions are positive in that the agent must increase protection somewhere at each time-step. (The simulator does not allow for no action to be taken). Thus, the random policy still leads to

reasonable results since it will tend to spread its protection actions out uniformly randomly among the candidate nodes. Moreover, without building a model and with access only to the observation vector at each time step, a reasonable strategy is to allocate protection to areas that have relatively low protection levels, compared to the other nodes. That is, a reasonable memoryless strategy is also to simply spread out the protection among the candidate nodes, since without knowledge of the underlying dynamics one must assume that all nodes are equal. Thus, intuitively the optimal memoryless strategy should be close to uniformly random, and this explains the similarity in scores between these two baselines.

Between the different CPSR methods, the Rademacher-projection based method performed the best with the spherical-projection method only performing slightly worse. This result is expected in that there are stronger theoretical guarantees for these methods compared to the hashed projection method.

Lastly, we see that the results are consistent across the two metrics. Interestingly, however, the performance increase between the top CPSR method and the random baseline is greater for the total (undiscounted) reward metric. For that metric, the total reward obtained via the top-performing CPSR method is 4.6% greater than the baseline, whereas for the discounted metric the top-performing method scores 3.7% greater than the random baseline. This makes sense in that the CPSR models should benefit more at longer horizons, since (1) it takes time for the CPSR model to incorporate observed information into its predictions and (2) the non-stationary in the domain, which is captured via the CPSR model, is only a factor at longer time-scales (Nicol et al., 2013).

7. Discussion

The CPSR approach provides a new avenue for model-based reinforcement learning where agents must formulate policies in large, complex partially observable domains without access to a fully-specified prior system model (i.e., where the system model must be learned prior to planning). The compressed learning algorithm allows accurate approximations of PSR models to be constructed in a memory and time efficient manner, and the use of random projections regularizes the learned solutions, preventing high variance models (over-fitting) and potentially leading to more accurate results. We elucidated theoretical guarantees bounding the induced approximation error of this model-learning approach, showing that the low-dimensional embeddings of the models retain predictive accuracy. In addition, we proposed a planning approach which exploits these compressed models in a principled manner, allowing for high-quality plans to be constructed without prior domain knowledge. Finally, we outlined how model-learning and planning can be combined at a high-level.

The empirical results we obtained demonstrate the efficacy of this approach and delineate domains in which its use is beneficial. The model quality experiments demonstrate that CPSR models achieve predictive accuracy competitive to that of uncompressed models, while taking a fraction of the runtime, and the planning results demonstrate that these models can be exploited by efficient planners, providing a novel and powerful framework for model-based reinforcement learning. Moreover, the results highlight the fact that the benefits of such a model-based approach are most stark in domains that are not only partially observable in the traditional sense but that are also strongly partially observable in that the Q-function (or a good approximation of it) is not discernible from the observation vec-

tors. In other words, the results demonstrate that aliased observations (and an unobserved hidden state) alone do not necessitate the need for a model-based learning algorithm. A model-based approach only becomes necessary when the observations are not sufficient for learning a reasonable approximation of the Q-function.

7.1 Practical Concerns

The implementation of complex RL frameworks often reveals practical issues that are not immediately apparent given formal descriptions. In order to facilitate the use of the CPSR algorithm in applications, we outline some pertinent practical issues that arise while implementing the CPSR algorithm and describe our solutions.

7.1.1 Selecting the Projection Matrices

First, it is necessary to reiterate the sensitivity of the approach with respect to both the projection dimension and type of projection used. Empirically, we found that the results could be quite sensitive to these parameters, though this was only the case for some domains. For example, selecting a projection dimension that is too small may lead to suboptimal (near-random) performance. This issue is further exacerbated by the fact that the true dimension of the underlying system is unknown.

The cause of the sensitivity with respect to the projection size is quite evident (smaller dimensions lose information but provide more regularization). However, the underlying cause of the differing performance between the different projection types is not as clear. One would expect the hash-type projection matrices to perform differently than the Rademacher and spherical projections, since the hash-type matrices do not satisfy the JL lemma, but we witnessed substantial variation between all three projection types, especially on the PacMan-type planning domains. Moreover, for the *ColoredGridWorld* domain, the difference between the projection types was more stark for planning performance compared to prediction performance.

The results thus indicate that planning performance is more sensitive to the choice of the projection matrix (compared to prediction performance). One explanation for this is simply that small discrepancies in the prediction performance of the models are amplified when agents must plan using the predictive models. The differing results obtained using the different projection matrices may then be due to the fact that a coarse-grained search (necessitated by computational requirements) for the compressed dimension-size was used and that different random projections may be optimal for slightly different projection sizes (Achlioptas, 2001). For example, a Rademacher projection may be near optimal at one point on the coarse-grained search while a spherical projection may be optimized at a point not included in the coarse-grained search. The slight differences in model-quality induced by the coarse-grained search would then propagate and lead to large variations in planning performance.

In order to cope with the sensitivity of the CPSR approach with respect to the projection sizes and dimension, we recommend using multiple phases of grid search (starting with exponentially separated values). Moreover, it is useful to narrow down the size-range for the projections using model-quality experiments (before performing hyperparameter opti-

mization for planning), since model-quality experiments are not as computational expensive (compared to planning experiments).

7.1.2 Improving Efficiency by Caching

In Section 3.2 we defined the projection operators via the functions $\phi_{\mathcal{T}} : \mathcal{T} \to \mathbb{R}^{d_{\mathcal{T}}}$ and $\phi_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^{d_{\mathcal{H}}}$. This specification engenders a number of benefits. Specifically, the full projection matrices do not need to be held in memory and the number of tests and histories do not need to be specified in advance. There is a runtime penalty associated with the technique, however, as the mappings must be recomputed each time a particular test or history is encountered while iterating over the sample trajectories. In order to ameliorate this issue, while retaining the benefits of specifying the projections as functions, we implemented a least-recently-used (LRU) cache. By caching the mappings for frequently encountered tests and histories, we improved the empirical runtime of the algorithm considerably.

7.1.3 Numerical Stability Issues

At its core, the CPSR algorithm relies on standard linear algebra techniques, namely SVD and matrix inversions, which are prone to numerical stability issues. If the matrices upon which these operations are performed are ill-formed, suboptimal results will be obtained (or the algorithm will simply fail). In this work, we found one common situation where such stability issues arise.

Since we do not normalize the probability estimates in Section 3.2, the singular values of $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ in (15) grow with the size of the training set. This leads to stability issues when inverting the matrix of singular values in order to compute the implicit pseudoinverse in (17) and (18). This stability issue can be alleviated by normalizing the probability estimates, or more generally, by scaling $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ by a small constant. Since this constant cancels out during learning, it can be picked arbitrarily, but it should be chosen such that the magnitude of the values in $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ are near unity. The most straightforward approach is to simply normalize the probability estimates, though this may not always suffice (e.g., if there are extremely unlikely events, the normalizer may make certain entries too small leading to further stability issues). We also empirically observed that setting $d' < d_{\mathcal{T}}$ and/or removing singular values below a certain threshold (a standard technique) helped with numerical stability.

7.1.4 Q-function Approximation and Sampling Strategies

Algorithm 2 in Section 5 permits a wide-variety of sampling strategies, and the sampling strategy used implicitly constrains the Q-function approximation obtained. In this work, we used an unbiased random sampling strategy in the batch setting. That is, we collected a large batch of random samples, which we used to both learn a model and construct plans. We opted for this framework as (1) our simulators were designed for the batch setting and (2) the theoretical results of Section 4 assume a blind (random) sampling strategy is used.

We did, however, experiment with a goal-directed sampling approach (Ong et al., 2012), where phases of exploration and exploitation are interleaved. In the goal-directed paradigm, a number of mini-batch sampling iterations are used, and the sampling policy (π_s) is updated at each iteration to be ϵ-greedy over the agent's current policy (π_i). Ong et al. (2012) found that this approach led to better performance in the small-sample setting. In our

experiments, where we used larger numbers of samples (on the order of 10000), we found that the goal-directed approach did not improve over random sampling and, in fact, often led to worse results and numerical instabilities. In particular, the bias in the sampling strategy led to an imbalance in the $\hat{\mathbf{\Sigma}}_{\mathcal{T},\mathcal{H}}$ matrix in that certain entries dominated in terms of magnitude. As a result of this imbalance, the SVD in (15) became unstable, and poor results were obtained. Such stability problems are likely to be an issue whenever biased sampling strategies are used in the large-sample batch setting. However, in online or small sample settings, such strategies will likely lead to performance increases due to the fact that their exploration is myopic and focuses on areas of the state-space relevant to planning (as shown by Ong et al., 2012).

7.1.5 Compressing Histories

The theoretical analysis of Section 4 assumes that $\mathbf{\Phi}_{\mathcal{H}}$ has orthonormal columns. However, in order to obtain maximal computational benefits, it is necessary to use a compressive $\mathbf{\Phi}_{\mathcal{H}}$, i.e. a $\mathbf{\Phi}_{\mathcal{H}}$ that acts as a feature selector on histories. In fact, for massive domains such as the PacMan-style domains, compressing histories is necessary for tractable learning and planning.

Viewing CPSR learning from the perspective of regression (as was done throughout this paper), the compression of histories is equivalent to compressing the samples used for regression; that is, it is equivalent to linearly mixing the samples. More formally, we use the transformation

$$\mathbf{y} = \mathbf{X}^\top \mathbf{w} + \boldsymbol{\eta} \rightarrow \mathbf{\Phi}_{\mathcal{H}}\mathbf{y} = \mathbf{\Phi}_{\mathcal{H}}\mathbf{X}^\top \mathbf{w} + \mathbf{\Phi}_{\mathcal{H}}\boldsymbol{\eta},$$

where as usual $\mathbf{X}$ is a design matrix, $\mathbf{w}$ a vector of regression weights, $\mathbf{y}$ a vector of targets, and $\boldsymbol{\eta}$ a vector of noise terms. Intuitively, we can view this projection by $\mathbf{\Phi}_{\mathcal{H}}$ as roughly averaging over training samples. The number of samples for the regression will then be reduced, but the averaged samples will have reduced (maximum) variance in their noise terms.

Of course, in this work, we use random $\mathbf{\Phi}_{\mathcal{H}}$ matrices, which do not correspond directly to taking averages over samples. The most important implication of this is that the noise terms of the new combined samples are not independent. This more complex setting has been analyzed in detail by Zhou et al. (2007) (for random Gaussian matrices). In that work, they focus on the more specific setting of l_1 regularized regression, and they prove a number of important results. Of particular relevance is Claim 4.3, which shows (under certain conditions) that the entry-wise discrepancy between $\mathbf{Q}^\top\mathbf{Q}$ and $\mathbf{Q}^\top\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{Q}$ decreases asymptotically to zero almost surely, where $\mathbf{Q} \in \mathbb{R}^{n\times m}$ and $\mathbf{\Phi} \in \mathbb{R}^{d\times n}$ is a random Gaussian matrix defined as in Theorem 1. This key result facilitates bounding the discrepancy between the compressed training error and the true error of the regressor and does not rely on l_1 regularization assumptions. We refer the interested reader to that work for detailed proofs.

Finally, we reiterate that in this work the compression of histories is a computational necessity, as it allows us to scale the learning algorithm to domains that would be intractable otherwise. And empirical investigations in Section 6 show that the compression of histories to $d_{\mathcal{H}} = d_{\mathcal{T}}$ introduces only a small amount of error during model-learning.

7.2 Related Work

The CPSR algorithm is closely related to work on using features or kernel embeddings with PSRs (Boots et al., 2010; Boots and Gordon, 2011; Boots et al., 2013), where features of tests, histories, and/or observations are employed. Indeed, one view of the CPSR learning approach is that it is an instantiation of the feature-based learning approach where principled random features are employed. However, this view is limited in the sense that the random features used here facilitate an analysis in terms of compression, whereas with other feature-based PSR methods it is simply assumed that the specified features are sufficient to capture the structure of $\boldsymbol{\mathcal{P}}_{\mathcal{T},\mathcal{H}}$; that is, the standard feature-based methods assume features that are not compressive (Boots et al., 2010; Boots and Gordon, 2011; Boots et al., 2013).

This distinction of whether or not features are assumed as compressive also highlights the differing motivations between existing feature-based PSR learning and the CPSR approach: in the CPSR approach, compressive random features are employed to increase the efficiency and scalability of learning, whereas in other works (e.g. Boots et al., 2010; Boots and Gordon, 2011; Boots et al., 2013) the features are used to facilitate learning in domains with continuous or structured observation spaces.

It should be noted, however, that since the general PSR learning framework assumes discrete observations, decomposing a continuous domain via feature extraction is necessary for learning in that setting. Moreover, Boots et al. (2013) shows how the well-known "kernel trick" can be employed to learn in feature-spaces of infinite dimension. The penalty associated with this kernel embedded approach is that learning scales cubically with the number of training examples, leading to high computational overhead (Boots et al., 2013). Boots and Gordon (2011) show how to partially alleviate this cost by using random features to approximate certain kernels, a technique that also relies on random projections (though not in the compressed sensing setting).

In a similar vein, the CPSR-based planner is closely related to the goal-directed planning and learning approach of Ong et al. (2012). The primary difference between our work and this goal-directed approach is that we present a more general combined learning and planning framework, which accommodates the use of a wide variety of sampling strategies.

Beyond these works, our approach bears similarities to the memory PSR (mPSR) approach of James et al. (2005), which uses a type of hybrid PSR-MDP model to reduce computational costs and increase predictive accuracy, and the hierarchical PSRs (HPSRs) of Wolfe and Singh (2006), which use the option framework (Sutton et al., 1999) to increase the predictive capacity of PSRs. Importantly, the improvements suggested by both these approaches are not incompatible with our compressed learning algorithm.

Our approach also shares similarities with certain model-based reinforcement learning algorithms, which use adaptive history-based techniques. Examples of these algorithms include *U-Tree* (McCallum, 1996) and the *Monte-Carlo AIXI approximation* (Veness et al., 2011). These approaches share the motivation of developing agents that can learn a model of dynamical system and plan using this model. They differ, however, in the instantiation of their model-based approach, as they use an adaptive history-based approach, which intuitively corresponds to learning mixtures of different k-order MDPs (where k varies adaptively). A key aspect of these approaches is focusing the model-learning on areas of the state-space relevant to achieving goals (similar to the goal-directed sampling routine)

(McCallum, 1996; Veness et al., 2011). Thus, a fundamental difference between Monte-Carlo AIXI-like approaches and the one proposed here is that they efficiently learn myopic models, necessarily constrained by the planning aspect of the problem, whereas in this work we retain the option of learning full-unbiased models of domains (i.e., our model-learning may be decoupled from planning). One implication of this is that the models learned via the CPSR learning approach may be reused in different planning contexts. However, a disadvantage of learning complete (i.e., full and unbiased) models is that it can be impractical in very large and complex domains.

7.3 Future Directions

Given the above discussion, an interesting direction for future work would be an analysis of the inductive bias associated with both the PSR and Monte-Carlo AIXI paradigms. Though these methods bear similarities, their theoretical motivations are quite distinct: PSRs being motivated by the theory of observable operators while certain AIXI-like methods have information-theoretic (and/or Bayesian) motivations (Veness et al., 2011). Recently, there have been a number of theoretical advancements in the understanding of observable operator methods, such as the local loss formulation of Balle et al. (2012) and the method of moments formulation of Anandkumar et al. (2012). These advancements could serve as tools in such an analysis. Perhaps the most interesting question in this area is understanding the regularization induced by these different paradigms (e.g., due to the restriction of the model classes). For example, the Monte-Carlo AIXI method explicitly penalizes model complexity, while this does not explicitly factor into the optimization of PSR-type methods (besides through the hyper-parameter selection of the model-size).

Another interesting avenue for the continuation of this work is exploring the use of different optimization frameworks during learning. In this work, we implicitly use the standard least-squares objective when solving the pseudoinverse in (17) and (18). However, there is no a priori reason to believe that this is the optimal formulation, and in fact, promising results have been obtained by modifying this optimization (e.g., through convex-relaxation) (Balle et al., 2012). Moreover, it is possible that alternative formulations may reveal novel regularization strategies (e.g., regularization on the implicit observable-operator structure) and additional algorithmic efficiencies.

Lastly, the framework presented here provides the necessary ingredients for applying a CPSR-based learning and planning framework to difficult real-world application problems, such as robot navigation problems similar to those solved by U-tree-based approaches (McCallum, 1996). Of course, such applications would introduce certain engineering issues not highlighted here. In particular, the sampling strategy, projection size, and projection type would necessarily be constrained by the problem domain and by hardware limitations; for example, it may be worthwhile to use highly optimized Rademacher projections. Moreover, in domains with extremely large action and observation dimensions, using a distributed implementation (e.g., of Equation 18 in the learning algorithm) would likely engender significant computational benefits. And, in domains with continuous observations, it would be necessary to combine discretization or kernel-based feature extraction with the CPSR compression techniques. These engineering issues, however, should not necessitate altering the core of the CPSR approach.

Acknowledgments

The authors would like to thank Doina Precup, Yuri Grinberg, Sylvie Ong, and Clement Gehring for helpful discussions on this work, and David Silver and Joel Veness for support on the PocMan domain. We are very grateful to our anonymous reviewers for their comments and recommendations. Financial support for this work was provided by NSERC Discovery and CGS-M grants.

Appendix A.

A.1 Proof of Theorem 1

Proof With eigenvalue decomposition we have $E_{\rho(h)}[\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}^\top] = \mathbf{V}\mathbf{D}\mathbf{V}^\top$, where $\mathbf{D}$ is the diagonal matrix containing the eigenvalues and $\mathbf{V}$ is an orthonormal basis. Let $\mathbf{I}_m$ be a $|\mathcal{Q}| \times |\mathcal{Q}|$ matrix with the first m diagonal elements set to 1 and 0 elsewhere. For all $1 \leq i \leq d$, define $[\tilde{\boldsymbol{\Phi}}]_{i,*} = [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top$ and $[\boldsymbol{\Phi}']_{i,*} = [\boldsymbol{\Phi}]_{i,*}\mathbf{V}$. Note that since $\mathbf{V}$ is an orthonormal basis and $[\boldsymbol{\Phi}]_{i,*}$ is i.i.d. normal, $[\boldsymbol{\Phi}']_{i,*}$ will also have an i.i.d. normal distribution with the same covariance.

We wish to substitute $[\boldsymbol{\Phi}]_{i,*}$ with $[\tilde{\boldsymbol{\Phi}}]_{i,*}$ which has a small norm and introduces a small bias. We first bound the norm of $[\tilde{\boldsymbol{\Phi}}]_{i,*}$ as follows. With probability no less than $1 - \delta/4$ for all $1 \leq i \leq d$

$$\begin{aligned}
\|[\tilde{\boldsymbol{\Phi}}]_{i,*}\|^2 &= [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top\mathbf{V}\mathbf{I}_m\mathbf{V}^\top[\boldsymbol{\Phi}]_{i,*}^\top \\
&= [\boldsymbol{\Phi}']_{i,*}\mathbf{I}_m([\boldsymbol{\Phi}']_{i,*})^\top = \sum_{j=1}^{m}([\boldsymbol{\Phi}']_{ij})^2 \\
&\leq m + 4\sqrt{m}\ln(4d/\delta). \qquad (36)
\end{aligned}$$

The tail bound in last line is union bounding over a corollary of Lemma 1 in Laurent and Massart (2000). The bias induced by using $[\tilde{\boldsymbol{\Phi}}]_{i,*}$ can be bounded as well. Define $b(h) = [\boldsymbol{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} - [\tilde{\boldsymbol{\Phi}}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$. With probability no less than $1 - \delta/4$ for all $1 \leq i \leq d$

$$\begin{aligned}
\|b(h)\|_{\rho(h)}^2 &= E_{\rho(h)}[([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}^\top([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})^\top] \\
&= ([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})\mathbf{V}\mathbf{D}\mathbf{V}^\top([\boldsymbol{\Phi}]_{i,*} - [\tilde{\boldsymbol{\Phi}}]_{i,*})^\top \\
&= ([\boldsymbol{\Phi}]_{i,*} - [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top)\mathbf{V}\mathbf{D}\mathbf{V}^\top([\boldsymbol{\Phi}]_{i,*} - [\boldsymbol{\Phi}]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top)^\top \\
&= [\boldsymbol{\Phi}]_{i,*}\mathbf{V}(\mathbf{I} - \mathbf{I}_m)\mathbf{D}(\mathbf{I} - \mathbf{I}_m)\mathbf{V}^\top[\boldsymbol{\Phi}]_{i,*}^\top \\
&= [\boldsymbol{\Phi}']_{i,*}(\mathbf{I} - \mathbf{I}_m)\mathbf{D}(\mathbf{I} - \mathbf{I}_m)([\boldsymbol{\Phi}']_{i,*})^\top \\
&= \sum_{j=m+1}^{|\mathcal{Q}|}([\boldsymbol{\Phi}']_{ij})^2\sigma_j^2 \\
&\leq \nu + 4\sqrt{\nu}\ln(4d/\delta). \qquad (37)
\end{aligned}$$

The tail bound again is due to Lemma 1 in Laurent and Massart (2000) using the assumption $\sigma_m^2 \leq 1$. Using the above bounds, we have for for all $1 \leq i \leq d$

$$\forall h : [\boldsymbol{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} = [\tilde{\boldsymbol{\Phi}}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + b(h) = ([\tilde{\boldsymbol{\Phi}}]_{i,*}\mathbf{B}_{ao})\boldsymbol{\mathcal{P}}_{\mathcal{Q},h} + b(h). \qquad (38)$$

Therefore, we have a target $[\mathbf{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}$ that is near-linear in the sparse features $\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$, with expected bias bounded by $b^2 = \nu + 4\sqrt{\nu}\ln(4d/\delta)$, and norm of the weight vector $[\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}$ bounded by $w^2 = \|\mathbf{B}_{ao}\|^2(m + 4\sqrt{m}\ln(4d/\delta))$.

By definition, $\mathbf{u}_i$ is the COLS estimate with input $\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},\mathcal{H}}$, target $[\mathbf{\Phi}]_{i,*}\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,\mathcal{H}}$, and projection $[\mathbf{\Phi}]_{-i,*}$. But in order to use the bound of Equation 29, we need to find the corresponding noise parameters of the COLS algorithm. Since, unlike the assumption of the general COLS bound, both the input and the output of the regression are noisy, we need to derive the effective overall noise variance in the sample output. We have

$$\begin{aligned}
[\mathbf{\Phi}]_{i,*}\hat{\boldsymbol{\mathcal{P}}}_{\mathcal{Q},ao,h} &= [\mathbf{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + [\mathbf{\Phi}]_{i,*}\Delta_y \\
&= [\tilde{\mathbf{\Phi}}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h} + b(h) + [\mathbf{\Phi}]_{i,*}\Delta_y \\
&= [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}(\hat{\mathbf{P}}_{\mathcal{Q},h} - \Delta_x) + b(h) + [\mathbf{\Phi}]_{i,*}\Delta_y \\
&= ([\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao})\hat{\mathbf{P}}_{\mathcal{Q},h} + b(h) + ([\mathbf{\Phi}]_{i,*}\Delta_y - [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\Delta_x).
\end{aligned}$$

And thus the sample points are

$$\hat{\mathbf{P}}_{\mathcal{Q},h} \rightarrow ([\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao})\hat{\mathbf{P}}_{\mathcal{Q},h} + b(h) + ([\mathbf{\Phi}]_{i,*}\Delta_y - [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\Delta_x). \tag{39}$$

The effective noise $[\mathbf{\Phi}]_{i,*}\Delta_y - [\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\Delta_x$ has mean 0. Since Δ_y is k-sparse and $\|[\tilde{\mathbf{\Phi}}]_{i,*}\mathbf{B}_{ao}\|^2 \leq w^2$, the variance of the effective noise term is bounded by $\max_j([\mathbf{\Phi}]_{ij})^2 k\sigma_y^2 + w^2\sigma_x^2$. Maximization over i and using a tail bound on the maximum of squared normals gives the σ_η^2 defined in the theorem.

We now apply the union bound to Equation 29. With probability no less than $1 - \delta/4$, for all $1 \leq i \leq d$,

$$\|\mathbf{u}_i([\mathbf{\Phi}]_{-i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) - [\mathbf{\Phi}]_{i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},ao,h}\|_{\rho(h)} \leq \epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/4d). \tag{40}$$

Note that by our definition of $\mathbf{C}_{ao}$, we have that $\mathbf{u}_i([\mathbf{\Phi}]_{-i,*}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h}) = (\mathbf{C}_{ao})_i(\mathbf{\Phi}\boldsymbol{\mathcal{P}}_{\mathcal{Q},h})$, which immediately gives the theorem by combining the error bounds on each row. ■

A.2 Proof of Theorem 3

Proof Similar to Theorem 1, we have $\boldsymbol{\mathcal{P}}_h = \boldsymbol{\beta}_\infty^\top \boldsymbol{\mathcal{P}}_{\mathcal{Q},h}$ for all h. Therefore we have a linear target and by definition $\mathbf{c}_\infty$ is the COLS estimate with projection $\mathbf{\Phi}$. We have

$$\begin{aligned}
\hat{\boldsymbol{\mathcal{P}}}_h &= \boldsymbol{\mathcal{P}}_h + \Delta_z = \boldsymbol{\beta}_\infty^\top \boldsymbol{\mathcal{P}}_{\mathcal{Q},h} + \Delta_z \\
&= \boldsymbol{\beta}_\infty^\top \hat{\mathbf{P}}_{\mathcal{Q},h} - \boldsymbol{\beta}_\infty^\top \Delta_x + \Delta_z.
\end{aligned} \tag{41}$$

Thus the effective variance is bounded by the σ_∞^2 defined in the theorem. We complete the proof by an application of the bound in Equation 29. ■

A.3 Proof of Theorem 4

Proof For all t, define $\mathbf{e}_t = \mathbf{C}_{a_t o_t}\mathbf{C}_{a_{t-1}o_{t-1}}\dots\mathbf{C}_{a_1 o_1}\mathbf{c}_1 - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:t}}$. After applying the nth compressed operator we have

$$\begin{aligned}
\|\mathbf{e}_n\|_{\rho_n} &= \|\mathbf{C}_{a_n o_n}\mathbf{C}_{a_{n-1}o_{n-1}}\dots\mathbf{C}_{a_1 o_1}\mathbf{c}_1 - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_n} \\
&= \|\mathbf{C}_{a_n o_n}(\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} + \mathbf{e}_{n-1}) - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_n} \\
&\leq \|\mathbf{C}_{a_n o_n}\mathbf{e}_{n-1}\|_{\rho_n} + \|\mathbf{C}_{a_n o_n}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_n} \\
&\leq \|\mathbf{C}_{a_n o_n}\mathbf{e}_{n-1}\|_{\rho_n} + \max_{o_n,a_n}\|\mathbf{C}_{a_n o_n}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} - \boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho_{n-1}} \\
&\leq c\|\mathbf{e}_{n-1}\|_{\rho_n} + \max_{o_n,a_n}\|\mathbf{C}_{a_n o_n}s_{n-1}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}} - s_{n-1}\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}\|_{\rho} \qquad (42) \\
&\leq c\|\mathbf{e}_{n-1}\|_{\rho_{n-1}} + s_{n-1}\epsilon \\
&\leq \epsilon\sum_{t=1}^{n-1} s_t c^{n-i-1}. \qquad (43)
\end{aligned}$$

Line 42 uses the distribution assumption on ρ_{n-1} and having $\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}}$ linear in $\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n-1}}$. Line 43 follows by induction. We now apply the normalizer operator:

$$\begin{aligned}
&\|\mathbf{c}_\infty\mathbf{C}_{a_n o_n}\mathbf{C}_{a_{n-1}o_{n-1}}\dots\mathbf{C}_{a_1 o_1}\mathbf{c}_1 - \mathbb{P}(o_{1:n}||a_{1:n})\|_{\rho_n} \\
&\quad= \|\mathbf{c}_\infty(\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}} + \mathbf{e}_n) - \mathbb{P}(o_{1:n}||a_{1:n})\|_{\rho_n} \\
&\quad\leq \|\mathbf{c}_\infty\mathbf{e}_n\|_{\rho_n} + \|\mathbf{c}_\infty\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}} - \mathbb{P}(o_{1:n}|a_{1:n})\|_{\rho_n} \\
&\quad\leq \|\mathbf{c}_\infty\|\|\mathbf{e}_n\|_{\rho_n} + \|\mathbf{c}_\infty s_n\boldsymbol{\mathcal{P}}_{\mathcal{Q},[ao]_{1:n}} - s_n\mathbb{P}(o_{1:n}||a_{1:n})\|_{\rho} \qquad (44) \\
&\quad\leq \|\mathbf{c}_\infty\|\epsilon\sum_{t=1}^{n-1} s_t c^{n-t-1} + \epsilon_\infty s_n. \qquad (45)
\end{aligned}$$

Line 44 uses the distribution assumption on ρ_n and Line 45 uses the bound of Theorem 3. ■

References

D. Achlioptas. Database-friendly random projections. In *Proceedings of the 20th ACM Symposium on Principles of Database Systems*, 2001.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden Markov models. In *Proceedings of the 25th Annual Conference on Learning Theory*, 2012.

B. Balle, A. Quattoni, and X. Carreras. Local loss optimization in operator models: A new insight into spectral learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

R. Baraniuk and M. Wakin. Random projections of smooth manifolds. *Foundations of Computational Mathematics*, 9:51–77, 2009.

B. Boots and G. Gordon. Predictive state temporal difference learning. In *Advances in Neural Information Processing Systems*, 2010.

B. Boots and G. Gordon. An online spectral learning algorithm for partially observable dynamical systems. In *Association for the Advancement of Artificial Intelligence*, 2011.

B. Boots and G. Gordon. A spectral learning approach to range-only SLAM. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning-planning loop with predictive state representations. In *Proceedings of Robotics: Science and Systems VI*, 2010.

B. Boots, G. Gordon, and A. Gretton. Hilbert space embeddings of predictive state representations. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, 2013.

M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision - European Conference on Computer Vision*, pages 707–720. Springer, 2002.

F. Denis and Y. Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 2008.

D. Ernst, P. Geurts, L. Wehenkel, and L. Littman. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

M.M. Fard, Y. Grinberg, J. Pineau, and D. Precup. Compressed least-squares regression on sparse spaces. In *Association for the Advancement of Artificial Intelligence*, 2012.

M.M. Fard, Y. Grinberg, A. Farahmand, J. Pineau, and D. Precup. Bellman error based feature generation using random projections on sparse spaces. In *Advances in Neural Information Processing Systems*, 2013.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63 (1):3–42, 2006.

G. Gordon. *Approximate Solutions to Markov Decision Processes*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1999.

W. L. Hamilton, M. M. Fard, and J. Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

T. Iwamura. *Spatial Conservation Prioritisation Under Global Threats*. PhD thesis, University of Queensland, 2011.

M. T. Izadi and D. Precup. Point-based planning for predictive state representations. In S. Bergler, editor, *Advances in Artificial Intelligence*, volume 5032 of *Lecture Notes in Computer Science*, pages 126–137. 2008.

H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

M. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

M. James, B. Wolfe, and S. Singh. Combining memory and landmarks with predictive state representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

A. Kulesza, N. R. Rao, and S. Singh. Low-rank spectral learning. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

M. Littman, Richard S., and Satinder S. Predictive representations of state. In *Advances in Neural Information Processing Systems*, 2002.

O.A. Maillard and R. Munos. Compressed least-squares regression. In *Advances in Neural Information Processing Systems*, 2009.

O.A. Maillard and R. Munos. Linear regression with random projections. *Journal of Machine Learning Research*, 13:2735–2772, 2012.

T. G. Martin, I. Chadès, P. Arcese, P. Marra, H Possingham, and D. Norris. Optimal conservation of migratory species. *Public Library of Science One*, 2(8):e751, 2007.

A McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, The University of Rochester, 1996.

S. Nicol, O. Buffet, T. Iwamura, and I. Chadès. Adaptive management of migratory birds under sea level rise. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013.

S. C. W. Ong, Y. Grinberg, and J. Pineau. Goal-directed online learning of predictive models. In S. Sanner and M. Hutter, editors, *Recent Advances in Reinforcement Learning*, volume 7188 of *Lecture Notes in Computer Science*, pages 18–29. 2012.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. V. N. Vishwanathan. Hash kernels for structured data. *The Journal of Machine Learning Research*, 10:2615–2637, 2009.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2010.

S. Singh, M. James, and M. Rudary. Predictive state representations: a new theory for modeling dynamical systems. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 2004.

R. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*, volume 1. Cambridge University Press, 1998.

R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

J. Veness, K.S. Ng, M. Hutter, W. Uther, and D. Silver. A Monte-Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40(1):95–142, 2011.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

E. Wiewiora. *Modeling Probability Distributions with Predictive State Representations*. PhD thesis, University of California at San Diego, 2007.

B. Wolfe and S. Singh. Predictive state representations with options. In *Proceedings of the 23rd International Conference on Machine learning*, 2006.

B. Wolfe, M. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

S. Zhou, J. Lafferty, and L. Wasserman. Compressed regression. In *Advances in Neural Information Processing Systems*, 2007.