Compressed Predictive State Representation:

An Efficient Moment-Method for Sequence Prediction and Sequential Decision-Making

William L Hamilton

Computer Science
McGill University, Montreal

August 13, 2014

Dedication

This thesis is dedicated to my sister Julianna.

Acknowledgements

I would like to thank everyone who helped and encouraged me, who constructively challenged my ideas or even just took the time to listen to them.

I am deeply grateful to my supervisor Joelle Pineau for her continual, constructive guidance, to Doina Precup for providing invaluable advice throughout my BSc and MSc, to Mahdi Milani Fard for helping me to get the theory of CPSRs off the ground, and to Borja Balle for showing me a whole new perspective on my own work. Special thanks to lab members Clement Gehring, Ouais Alsharif, and Yuri Grinberg for engaging with me in countless stimulating discussions.

Abstract

The construction of accurate predictive models over sequence data is of fundamental importance in the pursuit of developing artificial agents that are capable of (near)-optimal sequential decision-making in disparate environments. If such predictive models are available, they can be exploited by decision-making agents, allowing them to reason about the future dynamics of a system. Constructing models with sufficient predictive capacity, however, is a difficult task. Under the standard maximum likelihood criterion, these models tend to have non-convex learning objectives, and heuristics such as expectation-maximization suffer from high computational overhead. In contrast, an alternative statistical objective, the so-called method of moments, leads to convex optimizations that are often efficiently solvable via spectral decompositions.

This work further improves upon the scalability, efficiency, and accuracy of this moment-based framework by employing techniques from the field of compressed sensing. Specifically, random projections of high-dimensional data are used during learning to (1) provide computational efficiency and (2) regularize the learned predictive models. Both theoretical analyses, outlining an explicit bias-variance trade-off, and experiments, demonstrating the superior empirical performance of the novel algorithm (e.g., compared to uncompressed moment-methods), are provided. Going further, this work introduces a sequential decision-making framework which exploits these compressed learned models. Experiments demonstrate that the combination of the compressed model learning algorithm and this decision-making framework allows for agents to successfully plan in massive, complex environments without prior knowledge.

Abrégé

Pour pouvoir agir optimalement, il est important de pouvoir prédire les séquences d'observations à venir. Ceci est une tâche difficile puisque le problème d'estimation du maximum de vraisemblance est non-convexe. Les méthodes standard, tel que l'algorithme d'espérance-maximisation, sont très coûteuses et inefficaces. Une application récente de la méthode des moments offre une interprétation différente du problème qui est convexe et efficace.

La méthode présentée améliore l'efficacité et la précision de la méthode des moments dans le contexte de prédiction de séquences d'observations. Ceci est fait grâce à des projections aléatoires qui augmentent l'efficacité de l'algorithme. Une analyse théorique de notre méthode démontre que notre algorithme réduit la variance au dépend d'un peu plus de biais. Nos résultats empiriques démontret une meilleure performance comparativement aux méthodes précédentes. De plus, nous offrons un moyen d'exploiter les prédictions de notre algorithme de façon à agir optimalement. Ceci nous permet de produire des agents capable de raisonner dans des environnements complexes et partiellement observables.

Contents

List of Figures

1

Introduction

Prediction, the general task of estimating future values based on past experiences, lies at the heart of machine learning. From basic statistical modelling, where models are judged primarily via their predictive capacities, to the construction of complex artificially intelligent agents, where such agents invariably require the ability to predict future outcomes in their environments, the task of prediction remains paramount.

This work is concerned with a particular kind of prediction: sequence prediction, where predictions concern dynamic sequential entities (e.g., time-series) as opposed to simple static objects. This type of prediction problem is especially important in the pursuit of creating artificially intelligent agents that are capable of adapting and learning, as interacting with an environment is inherently a dynamic, sequential task. Given an accurate predictive model of a dynamic system, the task of constructing an agent capable of (near)-optimal sequential decision-making is greatly simplified. The complexity inherent in such dynamic entities greatly complicates the predictive modelling process, however, and as such, there are many aspects of the sequence prediction problem that remain as of yet unsolved or where substantial improvements are possible.

One particularly powerful and flexible model class employed in the area of sequence prediction is the class of probabilistic models with latent variables. The key insight in this class of models is that observed sequence data is often the manifestation of some underlying, or hidden, dynamics. This natural assumption is formalized by positing the existence of latent (i.e., hidden) variables, which control the dynamics of the system and emit the observed sequence data. By modelling the transition

structure between different hidden states and the probabilities governing the emission of observations from these hidden states, a succinct and powerful predictive model can be obtained.

Unfortunately, the power of latent variable models comes at a price. Unlike with other simpler models, learning and predicting with latent variable models is typically a computationally expensive task, to the point where obtaining exact solutions can become intractable. Designing efficient, scalable, and accurate approximation algorithms for these tasks is an important challenge that needs to be solved if we want to use these models for solving large-scale real world problems.

1.1 Problem Statement

This work addresses two main problems: (1) how to efficiently learn a latent variable model over sequence data; and (2) how to construct a sequential decision-making framework that exploits a learned predictive latent variable model.

1.1.1 Learning a Predictive Model using Moments

Learning a latent variable model from observed sequence data is a difficult task. It requires that one simultaneously (1) associate individual observations with particular hidden states (with the ground truth being unknown), (2) model the emission distributions of observations from the hidden states (given inferred associations), and (3) determine the optimal transition structure among the hidden states (i.e., the underlying dynamics of the system). The primary source of difficulty in this task is its simultaneous nature: the optimal association of observations to hidden states requires knowing the true probabilistic model of emissions and transitions which in turn requires knowing the optimal associations. In essence, the problem has the classic "chicken-and-egg" structure.

A standard solution to this learning problem under the maximum likelihood criterion is the expectation–maximization (EM) algorithm [20], which approximates a solution by alternating between steps of associating hidden states with observa-

tions and optimizing the probabilistic model given these associations. However, this algorithm suffers from two fundamental limitations: there is a high computational cost on large state spaces, and no statistical guarantees about the accuracy of the solutions obtained are available.

A recent alternative line of work consists of designing learning algorithms for latent variable models exploiting an entirely different statistical principle: the so-called method of moments [55]. Intuitively, this approach ameliorates the "chicken-and-egg" problem by simultaneously solving the association and optimization tasks using the method of matching statistical moments. The key idea underlying this principle is that, since the low order moments of a distribution are typically easy to estimate, by writing a set of equations that relate the moments with the parameters of the distribution and solving these equations using estimated moments, one can obtain approximations to the parameters of the target distribution. In some cases, solving these equations only involves spectral decompositions of matrices or tensors and basic linear algebra operations [3, 2]. In addition, statistical analyses show that these algorithms are robust to noise and can learn models satisfying some basic assumptions from samples of size polynomial in the relevant parameters [see 34, 3, 9, 33, and references therein].

A witness of the generality of the method of moments is the wide and ever-growing class of sequence prediction models that can be learned with this approach. These include classic latent variable models such as hidden Markov models (HMMs) [34] but also more general models such as weighted automata (WA) [10] and predictive state representations (PSRs) [16] (which are the focus of this work). Moreover, there are a number of works exploring direct applications of these sequences prediction models, e.g. algorithms for learning context-free formalisms used in natural language processing [6, 4, 18, 19, 45, 23].

Despite these favourable attributes and promising initial results, moment-based algorithms still remain computationally expensive when applied to environments that have high-dimensional observation spaces and require long-trajectories during learning (e.g., robotics environments), as the basic moment algorithms have complexity super-linear in these quantities (see chapter 4). In this work, we address the problem

of efficiently learning latent variable models using the method of moments, with a particular emphasis on designing a learning algorithm that scales well (i.e., linearly) with respect to both the observation dimension and the length of trajectories used in learning.

1.1.2 Sequential Decision-Making using a Predictive Model

The second half of this thesis deals with how to use latent variable models for planning and sequential decision-making. In the general task of sequential decision-making, an agent is viewed as acting in some environment with the goal of maximizing some measure of goodness (e.g., utility or rewards). More informally, sequential decision-making is the task of an intelligent agent deciding on a best course-of-action given its current situation [43].

The formalization of this procedure takes many forms depending on the assumptions made about both the agent and the environment. In this work, we use the standard approach of assuming discrete time states and view an agent as having three fundamental abilities, which are employed at each discrete point in time: (1) the ability to receive observations, which confer some (but not complete) information about the current state of the system; (2) the ability to infer a measure of goodness (i.e., reward or utility); and (3) the ability to influence the system by taking some action. Importantly, we do not assume that the agent has complete access to the system state, meaning that the agent is tasked with the problem of sequential decision-making under uncertainty. Connecting this discussion back to predictive modelling, we assume that the agent has access to sequence data that is generated by processes within an environment, but we do not assume that the agent has knowledge of the governing processes. Environments of this type are often termed partially observable [40]. Thus in the framework used here, an agent must act in a partially observable system and simultaneously deal with uncertainty while also attempting to maximize its utility or reward.

A classic solution to the problem of sequential decision making under uncertainty is to use the partially observable Markov decision process (POMDP) formalism,

a generalization of a latent variable model to the actuated (i.e., decision-making) paradigm [40]. More formally, the latent variable model is augmented such that hidden states (1) take an input (i.e., action) that affects the transition structure and (2) emit rewards (which may or may not depend on the action taken).

It should be noted, however, that in the vast majority of work on sequential decision-making using POMDPs, the model is assumed to be known, and the primary task is simply to do sequential decision-making using this known probabilistic model (which incorporates uncertainty) [40]. These models are usually specified by domain experts and often lead to state-of-the-art results on difficult problems, such as navigating robotic helicopters or playing complex games (e.g., chess or PacMan) [50, 66].

In this work we examine the more difficult problem of both learning the probabilistic model (using the method of moments) and performing sequential decision-making given this learned model. An important insight facilitating our approach is that since POMDPs are the generalization of latent variable models over sequence data, the moment-based framework can also be applied to learn their parameters (and similarly for related models), making the learning problem more tractable. In these cases where the moment-method is used to learn a model that incorporates actions, the learned models are often termed predictive state representations (PSRs) [44].

In the following chapters, we show how to efficiently learn a PSR model, the moment-based generalization of a POMDP, and we develop a sequential decision-making framework that exploits the information contained within the PSR model state. Moreover, we treat the combined learning and planning problem in an *agnostic* fashion, where we assume that the agent has no prior knowledge about the domain in which it is acting. In this agnostic learning framework an agent must build a predictive model and learn to make (near)-optimal sequential decisions using only execution traces, i.e. action, observation, and reward sequences sampled via interactions with the environment.

This agnostic learning problem is both important and lags behind in terms of research results. 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 are more agnostic and thus free from unintended biases. At a more theoretical level, the development of agnostic and general learners is fundamental in the pursuit of creating truly intelligent artificial agents that can learn and succeed independent of prior domain knowledge.

Here, we explore applications of agnostic moment-based sequential decision-making to simulated robot path planning and obstacle avoidance domains. However, this type of method could be applied in countless domains where both learning and sequential decision-making are necessary. For example, there are potential applications to ecological management (learning and determining optimal intervention strategies) [51], health care [63], and adaptive dialogue management [76].

1.2 Thesis Statement

This work introduces a novel method of moments learning algorithm, termed compressed predictive state representation (CPSR), for modelling sequence data, along with a sequential decision-making framework built upon this algorithm. By combining techniques from the field of compressed sensing [24] with the moment-method framework, this algorithm provides state-of-the-art performance in terms of computational efficiency without sacrificing predictive accuracy. Specifically, we use random projections [11], where high-dimensional vectors are projected onto randomly generated low-dimensional manifolds, during learning in order to increase the efficiency and scalability of the moment-based framework. And we show how this approach regularizes the learning the process.

The sequential decision-making framework we propose exploits these low-dimensional models in a principled manner. Moreover, the combination of this efficient learning algorithm with this principled decision-making framework allows agents to learn to make (near)-optimal decisions in complex systems with no prior knowledge of the system's dynamics.

1.3 Statement of Contributions

The main contributions of the thesis are as follows:

- The unified presentation of the different moment-based learning algorithms for sequence prediction (chapter 2).
- The generalization of the tensor decomposition based moment-method to work with variable length sequence prefixes/suffixes during learning (chapter 2).
- The explicit characterization of the relationship between PSRs and the more general method of moments (chapter 3).
- The derivation of an efficient and scalable moment-based learning algorithm, CPSR, that uses random projections to increase efficiency and provide regularization (chapter 4).
- The explicit characterization of how the CPSR learning algorithm can be used in both batch and incremental/online learning settings (chapter 4).
- A bias-variance analysis of the learning algorithm that bounds the excess risk of learning in a compressed space (chapter 5).
- The extension of the compressed regression framework [46, 47, 28] to deal with compression on both (noisy) input features and a (noisy) regression target (chapter 5).
- The derivation of a sequential decision-making (i.e., planning) framework that exploits the concise CPSR states in a principled manner (chapter 6).
- The specification of how CPSR learning and planning can be interleaved to incrementally explore hard to reach regions of an environment (chapter 6).
- Experiments demonstrating the use of the CPSR based sequential decision-making framework on simulated robot navigation domains (chapter 7).

- Empirical results comparing the performance of different random projection matrices (chapter 7).
- Empirical results demonstrating that CPSR maintains predictive performance competitive with uncompressed moment-based models (chapter 7).
- Empirical results demonstrating that a CPSR based learning and decision-making framework is capable of outperforming both a uncompressed moment-based framework and a baseline memoryless planner (chapter 7).
- Empirical results demonstrating that a CPSR based learning and decision-making framework is capable of scaling to domains that are infeasible for an uncompressed moment-based framework (chapter 7).
- A description of how a cache-based implementation of the random projections can increase empirical runtime efficiency (chapter 7).

Certain aspects of thesis are taken from works that are in preparation for publication or have been published. In particular, the introduction and chapter 2, include modified parts of [8]. Substantial portions of the rest of this thesis are taken from [32] and [31], with [31] being an extended version of [32]. The author of this thesis is the primary author of all these works[1], and the collaborators acknowledge the use of these works in this thesis. Moreover, this thesis represents a substantial contribution beyond these independent works. In particular, this thesis synthesizes results from the generic method of moments setting and the more specific predictive state representation approach and, in chapter 3, explicitly derives predictive state representation as a special case of the more general method of moments.

[1]The author of this thesis shares first authorship with Borja Balle in [8].

2

Technical Foundations

2.1 Notation

2.1.1 Matrix Algebra Notation

Bold letters denote vectors $\mathbf{v} \in \mathbb{R}^d$, matrices $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$, and third-order tensors $\mathbf{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$. Given a matrix $\mathbf{M}$, $\|\mathbf{M}\|$ denotes its Frobenius norm and $\|\mathbf{M}\|_*$ its trace/nuclear norm. $\mathbf{M}^+$ is used to denote the Moore–Penrose pseudo-inverse 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.

A matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ is symmetric if $\mathbf{M} = \mathbf{M}^\top$. Similarly, a tensor $\mathbf{T} \in \mathbb{R}^{d \times d \times d}$ is symmetric if for any permutation ρ of the set $\{1, 2, 3\}$ we have $\mathbf{T} = \mathbf{T}^\rho$, where $[\mathbf{T}^\rho]_{i_1,i_2,i_3} = [\mathbf{T}]_{i_{\rho(1)},i_{\rho(2)},i_{\rho(3)}}$ for every $i_1, i_2, i_3 \in [d]$. Given vectors $\mathbf{v}_i \in \mathbb{R}^{d_i}$ for $1 \leq i \leq 3$, we can take tensor products to obtain matrices $\mathbf{v}_1 \otimes \mathbf{v}_2 = \mathbf{v}_1\mathbf{v}_2^\top \in \mathbb{R}^{d_1 \times d_2}$ and tensors $\mathbf{v}_1 \otimes \mathbf{v}_2 \otimes \mathbf{v}_3 \in \mathbb{R}^{d_1 \times d_2 \times d_3}$. For convenience we also write $\mathbf{v} \otimes \mathbf{v} \otimes \mathbf{v} = \mathbf{v}^{\otimes 3}$, which is a third-order symmetric tensor.

Given a tensor $\mathbf{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and matrices $\mathbf{M}_i \in \mathbb{R}^{d_i \times d_i'}$ we consider the contraction operation that produces a tensor $\mathbf{T}' = \mathbf{T}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3) \in \mathbb{R}^{d_1' \times d_2' \times d_3'}$ with entries given by $[\mathbf{T}']_{j_1 j_2 j_3} = [\mathbf{M}_1]_{i_1 j_1} [\mathbf{M}_2]_{i_2 j_2} [\mathbf{M}_3]_{i_3 j_3} [\mathbf{T}]^{i_1 i_2 i_3}$, where Einstein's summation convention is used.

2.1.2 Sequence Notation

In the context of general sequence prediction, we equate sequences with strings composed of symbols from some finite alphabet.[1] Let Σ be a finite alphabet. $\Sigma^\star$ denotes the set of all finite strings over Σ, and λ, the empty string. Given two strings $u, v \in \Sigma^\star$, $w = uv$ denotes their concatenation, in which case we say that u is a prefix of w, and v is a suffix of w. Given two sets of strings $\mathcal{P}, \mathcal{S} \subseteq \Sigma^\star$, $\mathcal{P}\mathcal{S}$ denotes the set obtained by taking every string of the form uv with $u \in \mathcal{P}$ and $v \in \mathcal{S}$. When singletons are involved, we write $u\mathcal{S}$ instead of $\{u\}\mathcal{S}$ for convenience. If $f : \Sigma^\star \to \mathbb{R}$ is a function, we use $f(\mathcal{P})$ to denote $\sum_{u \in \mathcal{P}} f(u)$. Given strings $u, v \in \Sigma^\star$, we denote by $|v|_u$ the number of occurrences of u as a substring of v; that is, $|v|_u = |\{(w, z) | v = wuz\}|$.

2.1.3 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 simplicity, $|$ also denotes conditioning upon an agents policy (i.e., plan). That is, when the $|$ symbol is followed by an ordered list of actions, it denotes that we are conditioning upon the knowledge that the agent will "intervene" in a system by executing the specified actions.

[1] The sharp-eyed reader will notice that we have restricted the sequence prediction problem to the case of discrete observations. The majority of this work assumes discrete observations; however, continuous alphabets can be handled in this framework by working with features of continuous observations [60] or by using kernel embeddings [13].

2.2 Moment-Methods for Sequence Prediction

In the general setting of sequence prediction, the goal is to use the method of moments to efficiently learn a distribution over sequences coming from some unknown distribution. There are countless applications and concrete instantiations of this framework (e.g., natural language processing or reinforcement learning)[16, 75, 6, 4, 18, 19, 45, 23]; however the discussion in this section will remain quite general, making minimal assumptions. Section 2.2.1 provides the general theoretical framework, which we will build upon throughout this work, and section 2.2.3 describes the prototypical method of moments algorithm in this setting. This prototypical algorithm serves as a foundation both for a brief review of existing moment-methods in section 2.2.4 and for the presentation of the novel compressed predictive state representation algorithm in chapter 4.

2.2.1 A Concrete Framework: Latent Variable Models and Weighted Automata

.

A *stochastic language* is a probability distribution over $\Sigma^\star$. More formally, it is a function

$$f : \Sigma^\star \to \mathbb{R} \tag{2.1}$$

such that $f(x) \geq 0$ for every $x \in \Sigma^\star$ and $\sum_{x\in\Sigma^\star} f(x) = 1$ [57]. The main learning problem is thus to infer a stochastic language $\hat{f}$ from a sample $S = (x^1, \ldots, x^m)$ of i.i.d. strings generated from some stochastic language f. In order to give a succinct representation for $\hat{f}$ we use hypothesis classes based on finite automata. In the following we present several types of automata that are used throughout this work.

A *weighted automaton (WA)* over Σ is a tuple $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}_{\sigma\in\Sigma}\rangle$, with $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ and $\mathbf{A}_\sigma \in \mathbb{R}^{n\times n}$. The vectors $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_\infty$ are called the initial and final weights, respectively. Matrices $\mathbf{A}_\sigma$ are transition operators containing transition

weights. The size n of these objects is the number of states of A. A weighted automaton A computes a function $f_A : \Sigma^\star \to \mathbb{R}$ as follows:

$$f_A(x_1 \cdots x_t) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_t} \boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_0^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty \ . \tag{2.2}$$

We say that a function $f : \Sigma^\star \to \mathbb{R}$ is *realized* by A if $f_A = f$. If f_A is a stochastic language, then we say that A is a *stochastic automaton*.

A learning task one might consider in this setting is the following: assuming the target stochastic language can be realized by some WA, try to find a stochastic WA realizing approximately the same distribution (w.r.t. some metric). The methods given in [34, 5] – which we review in section 2.2.4 – can be used to solve this problem, provided one is content with a WA A such that $\hat{f} = f_A$ approximates f but is not necessarily stochastic. It turns out that this is an essential limitation of using WA as a hypothesis class: in general, checking whether a WA A is stochastic is an undecidable problem [21]. Thus, if one imperatively needs the hypothesis to be a probability distribution, it is necessary to consider methods that produce a WA which is stochastic *by construction*. These include probabilistic automata and hidden Markov models.

A *probabilistic automaton (PA)* is a WA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\} \rangle$ where the weights satisfy the following conditions:

1. $\boldsymbol{\alpha}_0 \geq 0$ with $\boldsymbol{\alpha}_0^\top \mathbf{1} = 1$; and,
2. $\boldsymbol{\alpha}_\infty \geq 0$, $\mathbf{A}_\sigma \geq 0$, with $\sum_\sigma \mathbf{A}_\sigma \mathbf{1} + \boldsymbol{\alpha}_\infty = \mathbf{1}$.

These conditions say that $\boldsymbol{\alpha}_0$ can be interpreted as probabilities of starting in each state and that $\mathbf{A}_\sigma$ and $\boldsymbol{\alpha}_\infty$ define a collection of emission/transition and stopping probabilities that describe all the possible events that can occur from a given state. It is easy to check that PA are stochastic by construction; that is, when A is a PA the function f_A is a stochastic language.

A *factorized weighted automaton (FWA)* is a tuple $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \mathbf{T}, \{\mathbf{O}_\sigma\}_{\sigma \in \Sigma} \rangle$ with initial and final weights $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$, transition weights $\mathbf{T} \in \mathbb{R}^{n \times n}$, and emission weights $\mathbf{O}_\sigma \in \mathbb{R}^{n \times n}$, where the matrices $\mathbf{O}_\sigma$ are diagonal. One can readily transform a FWA into a WA by taking $B = \langle \boldsymbol{\beta}_0, \boldsymbol{\beta}_\infty, \{\mathbf{B}_\sigma\} \rangle$ with $\boldsymbol{\beta}_0 = \boldsymbol{\alpha}_0$, $\boldsymbol{\beta}_\infty = \boldsymbol{\alpha}_\infty$,

and $\mathbf{B}_\sigma = \mathbf{O}_\sigma \mathbf{T}$. A *hidden Markov model (HMM)* is a FWA where the weights satisfy the following conditions:

1. $\boldsymbol{\alpha}_0 \geq 0$ with $\boldsymbol{\alpha}_0^\top \mathbf{1} = 1$;
2. $\mathbf{T} \geq 0$ with $\mathbf{T}\mathbf{1} = \mathbf{1}$; and,
3. $\boldsymbol{\alpha}_\infty \geq 0$, $\mathbf{O}_\sigma \geq 0$, with $\sum_\sigma \mathbf{O}_\sigma \mathbf{1} + \boldsymbol{\alpha}_\infty = \mathbf{1}$.

It can be easily checked that these conditions imply that the WA obtained from a HMM is a PA. For convenience, given a HMM we also define the observation matrix $\mathbf{O} \in \mathbb{R}^{\Sigma \times n}$ with entries $\mathbf{O}(\sigma, i) = \mathbf{O}_\sigma(i, i)$.

Note that unlike with WA, both PA and HMM readily define probability distributions. But this comes at a price: there are stochastic WA realizing probability distributions that cannot be realized by any PA or HMM with a finite number of states [21]. In terms of representational power, both PA and HMM are equivalent when the number of states is unrestricted. However, in general PA provide more compact representations: given a PA with n states one can always obtain an HMM with $\min\{n^2, n|\Sigma|\}$ states realizing the same distribution. On the other hand, there are PA with n states such that every HMM realizing the same distribution needs more than n states [26]. These facts imply that different hypothesis classes for learning stochastic languages impose different limitations to the class of distributions we might be able to learn and to the extent to which we can compress these representations. Of particular importance to this work is the relative conciseness of PA (or more generally WA) compared to HMMs.

Given a stochastic language f that assigns probabilities to strings, there are two functions computing aggregate statistics that one can consider: f^{p} for probabilities of prefixes, and f^{s} for expected number of occurrences of substrings. In particular, we have $f^{\mathrm{p}}(x) = f(x\Sigma^\star) = \sum_{y \in \Sigma^\star} f(xy)$, and $f^{\mathrm{s}}(x) = \mathbb{E}_{y \sim f}[|y|_x] = \sum_{y,z \in \Sigma^\star} f(yxz)$. Note that given a sample S of size m generated from f, it is equally easy to estimate the empirical probabilities $\hat{f}_S(x) = (1/m) \sum_{i=1}^m \mathbb{I}[x^i = x]$, as well as empirical prefix probabilities $\hat{f}^{\mathrm{p}}_S(x) = (1/m) \sum_{i=1}^m \mathbb{I}[x^i \in x\Sigma^\star]$ and empirical substring occurrence expectations $\hat{f}^{\mathrm{s}}_S(x) = (1/m) \sum_{i=1}^m |x^i|_x$. It is shown in [7] that when f is realized by a

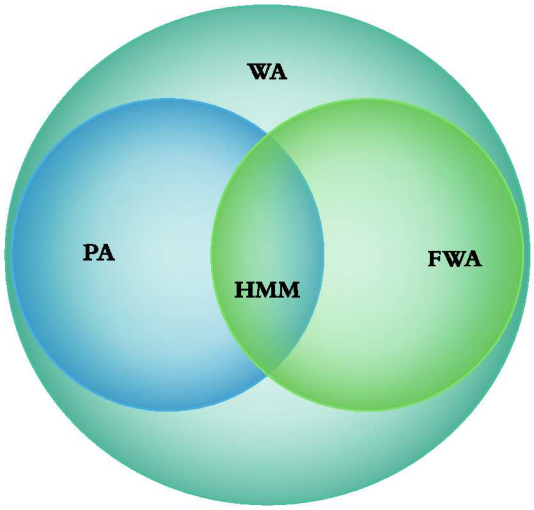


Figure 2.1: Venn diagram summarizing the relative expressiveness of different subclasses of weighted automata with respect to representing stochastic languages.

WA, PA, or HMM, then so are f^{p} and f^{s}. This result provides an explicit conversion that preserves the number of states and that can be easily reversed. Therefore, in terms of learning algorithms, one can work with any of these three representations indistinctively.

2.2.2 Learning Latent Variable Models with EM

A classic approach to learning latent-variable models (e.g., HMMs) under the maximum likelihood criterion is the expectation-maximization (EM) algorithm [20]. Though this algorithm is fundamentally distinct from the moment-based algorithms that are the focus of this work, the wide-spread use of this approach and its historical significance necessitate a brief description.

In short, EM is an iterative algorithm for locally minimizing the non-convex negative log-likelihood function of a latent variable model [20]. It alternates between two types of steps: an *expectation* (E) step, where the expected distribution of the hidden variables are computed, and a *maximization* (M) step, where the parameters

of the model are updated by maximizing the joint likelihood of the observed data and the expected hidden variables' distributions. These two-steps are iterated until the algorithm converges and the parameters stop changing (empirically, a ϵ convergence tolerance is used).

For HMMs, the time complexity of each of iteration is $O(n^2 L|S|)$, where n is the number of hidden states, $|S|$ is the number of training sequences (i.e., the cardinality of the sample set S), and is L the max-length of a training sequence in S [58]. No closed form expression for the number of iterations necessary to achieve convergence is known, but the empirical runtime for EM is generally quite high for complex models such as HMMs, since each iteration incurs significant computational cost.

Moreover, to avoid getting stuck in local-minima, restarts and other heuristics are usually necessary to achieve good statistical performance [35]. In practice, given enough time to explore the space of parameters, these heuristics yield very competitive models on problems with relatively small state spaces (i.e., state spaces of size $O(10^2)$) [e.g., 71]. In this work, however, we are concerned with much larger domains (e.g., domains with $\approx 10^{56}$ states), where explicit state-space representations are intractable, and as such, EM will not be of practical use for the problems we consider.

2.2.3 A Simple Moment-Based Algorithm: Factorizing a Hankel Matrix

A key step underlying the method of moments algorithms for learning stochastic languages is the arrangement of a finite set of values of f into a Hankel matrix or tensor in a way such that spectral factorizations of these linear objects reveal information about the operators of a WA, PA, or HMM realizing f. As a simple example, consider $f = f_A$ for some WA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ with n states. Given two sets of strings $\mathcal{P}, \mathcal{S} \subset \Sigma^\star$ which we call prefixes and suffixes, consider the matrix $\mathbf{H} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ with entries given by $[\mathbf{H}]_{u,v} = f(uv)$. This is the Hankel matrix[2] of f on

[2]In real analysis a matrix $\mathbf{M}$ is Hankel if $[\mathbf{M}]_{i,j} = [\mathbf{M}]_{k,l}$ whenever $i + j = k + l$, which in particular implies that $\mathbf{M}$ is symmetric because of the commutativity $i + j = j + i$ [54]. In our case

prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$. Writing $f(u, v) = (\boldsymbol{\alpha}_0^\top \mathbf{A}_u)(\mathbf{A}_v \boldsymbol{\alpha}_\infty)$ we see that this Hankel matrix can be written as

$$\mathbf{H} = \mathbf{P}\mathbf{S}, \tag{2.3}$$

where $\mathbf{P} \in \mathbb{R}^{\mathcal{P} \times n}$ with the uth row equal to $\boldsymbol{\alpha}_0^\top \mathbf{A}_u$, and $\mathbf{S} \in \mathbb{R}^{n \times \mathcal{S}}$ with vth column equal to $\mathbf{A}_v \boldsymbol{\alpha}_\infty$. Then it is easy to see that the Hankel matrix $\mathbf{H}_\sigma \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ with entries $[\mathbf{H}_\sigma]_{u,v} = f(u\sigma v)$ for some $\sigma \in \Sigma$ can be written as

$$\mathbf{H}_\sigma = \mathbf{P}\mathbf{A}_\sigma\mathbf{S}. \tag{2.4}$$

Thus, a way to recover the operators $\mathbf{A}_\sigma$ of A is to obtain a factorization $\mathbf{H} = \mathbf{P}\mathbf{S}$ and use it to solve for $\mathbf{A}_\sigma$ in the expression of $\mathbf{H}_\sigma$.

2.2.4 Overview of Existing Methods

In this section, we describe three algorithms for using the methods of moments to learn a weighted automaton from data. This selection is representative of the possible approaches to the method of moments, all of which involve either singular value decompositions, convex optimization, or symmetric tensor decompositions. Moreover, all these methods build upon the idea that by arranging particular values (i.e., estimates of low-order moments) in matrices or tensors, the equations relating the target parameters of the automaton to these observed statistics are solvable via factorizations of these algebraic objects. For ease of presentation, we assume that we have access to the target stochastic language f, which can be used to compute the probability $f(x)$ of any string $x \in \Sigma^\star$. Very few modifications are needed when the algorithms are applied to empirical estimates $\hat{f}_S$ computed from a sample S. We give detailed descriptions of these modifications wherever they are needed.

2.2.4.1 The Spectral Method

The first of the three methods of moments presented here is simply a concrete instantiation of the Hankel factorization approach, elucidated in section 2.2.3, using

we have $[\mathbf{H}]_{u,v} = [\mathbf{H}]_{w,z}$ whenever $uv = wz$, but $\mathbf{H}$ is not symmetric because string concatenation is not commutative.

singular value decomposition (SVD) as the factorization method. We now proceed to give the details of the algorithm, which is based on [34, 5]. The algorithm computes a minimal WA that approximates f but which, in general, is not stochastic. As input, the method requires sets of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^\star$, and the number of states n of the target automaton.

The algorithm starts by computing the Hankel matrices $\mathbf{H}, \mathbf{H}_\sigma \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ for each $\sigma \in \Sigma$. It also computes vectors $\mathbf{h}_{\lambda,\mathcal{S}} \in \mathbb{R}^{\mathcal{S}}$ with $\mathbf{h}_{\lambda,\mathcal{S}}(v) = f(v)$ and $\mathbf{h}_{\mathcal{P},\lambda} \in \mathbb{R}^{\mathcal{P}}$ with $\mathbf{h}_{\mathcal{P},\lambda}(u) = f(u)$. Next, it computes the reduced SVD[3] decomposition $\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ with $\mathbf{U} \in \mathbb{R}^{\mathcal{P}\times n}$, $\mathbf{V} \in \mathbb{R}^{\mathcal{S}\times n}$ and diagonal $\mathbf{D} \in \mathbb{R}^{n\times n}$. The algorithm then returns a WA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ given by

$$\boldsymbol{\alpha}_0^\top = \mathbf{h}_{\lambda,\mathcal{S}}^\top \mathbf{V} \ , \tag{2.5}$$

$$\boldsymbol{\alpha}_\infty = \mathbf{D}^{-1}\mathbf{U}^\top \mathbf{h}_{\mathcal{P},\lambda} \ , \tag{2.6}$$

$$\mathbf{A}_\sigma = \mathbf{D}^{-1}\mathbf{U}^\top \mathbf{H}_\sigma \mathbf{V} \ . \tag{2.7}$$

We note that this algorithm is implicitly using the factorization $\mathbf{H} = \mathbf{P}\mathbf{S} = (\mathbf{U}\mathbf{D})(\mathbf{V}^\top)$, i.e. where $\mathbf{P} = (\mathbf{U}\mathbf{D})$ and $\mathbf{S} = \mathbf{V}^\top$. The alternative factorization $\mathbf{H} = (\mathbf{U})(\mathbf{D}\mathbf{V}^\top)$ produces a modified but equivalent algorithm. We choose this factorization in order to maintain consistency with later sections.

This algorithm is remarkably simple, requiring only a single SVD, and has been shown to be statistically consistent [34]. It is important to reiterate that this method is guaranteed to return a stochastic automaton only in the case where infinite data is used in estimating $\mathbf{H}$ and thus in practice only a general WA is returned which gives *scores* not probabilities to strings. In practice, these scores tend to closely approximate probabilities [16, 13, 34]. The method is reasonably efficient, especially when compared to EM and more complex moment-methods, with a time complexity of $O(n|\Sigma||\mathcal{P}||\mathcal{S}|)$, assuming that $\mathbf{H}$ is given as an input.

Given its simplicity and efficiency, the spectral method is used in numerous applications where WA are learned from data, e.g. reinforcement learning [16, 13, 52, 32] and natural language processing [6, 45, 18]. In addition, its generality has lead to its

[3]When using an approximation $\hat{\mathbf{H}}$, the algorithm computes the n-truncated SVD instead.

use in problems not directly related to sequence prediction. For example it has been used robotics to perform range-only simultaneous localization and mapping (SLAM) [14].

2.2.4.2 Convex Optimization Method

This method recovers the operators of a WA by solving an optimization problem with the sum of a loss function involving the Frobenius norm and a trace norm regularizer. To motivate the algorithm, recall that if f is computed by a WA with n states, then a Hankel matrix of f admits a factorization of the form $\mathbf{H} = \mathbf{PS}$, $\mathbf{P} \in \mathbb{R}^{\mathcal{P}\times n}$, $\mathbf{S} \in \mathbb{R}^{n\times\mathcal{S}}$. Now suppose that $\mathbf{P}$ has rank n. Then, taking $\mathbf{B}_\sigma = \mathbf{P}\mathbf{A}_\sigma\mathbf{P}^+ \in \mathbb{R}^{\mathcal{P}\times\mathcal{P}}$ we have $\mathbf{B}_\sigma\mathbf{H} = \mathbf{H}_\sigma$ and $\text{rank}(\mathbf{B}_\sigma) \leq n$. Since the WA given by $B = \langle \boldsymbol{\beta}_0, \boldsymbol{\beta}_\infty, \{\mathbf{B}_\sigma\}\rangle$ with $\boldsymbol{\beta}_0^\top = \boldsymbol{\alpha}_0^\top\mathbf{P}^+$ and $\boldsymbol{\beta}_\infty = \mathbf{P}\boldsymbol{\alpha}_\infty$ satisfies $f_A = f_B$, this motivates an algorithm that looks for a low-rank solution of $\mathbf{MH} = \mathbf{H}_\sigma$; the learned $\mathbf{M}$ t corresponding to a low-rank approximation of $\mathbf{B}_\sigma$.

An algorithm based on this principle is described in [10]. As input, the method requires sets of prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$ with $\lambda \in \mathcal{P} \cap \mathcal{S}$, and a non-negative regularization parameter $\tau \in \mathbb{R}$. The number of states of the WA produced by this algorithm is equal to the number of prefixes $|\mathcal{P}|$. We start by describing this method as it was originally presented in [10]. Then we show how a tiny variation of it can be used to obtain a PA as output.

The algorithm starts by computing two Hankel matrices $\mathbf{H} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ and $\mathbf{H}_\Sigma \in \mathbb{R}^{\mathcal{P}\Sigma\times\mathcal{S}}$, where $\mathbf{H}$ is defined like before, and $[\mathbf{H}_\Sigma]_{u\sigma,v} = f(u\sigma v)$. Note that because we have $\lambda \in \mathcal{P} \cap \mathcal{S}$, now the vectors $\mathbf{h}_{\mathcal{P},\lambda}$ and $\mathbf{h}_{\lambda,\mathcal{S}}$ are contained inside of $\mathbf{H}$. The operators of the hypothesis are obtained by solving the optimization problem

$$\mathbf{A}_\Sigma \in \underset{\mathbf{M}\in\mathbb{R}^{\mathcal{P}\Sigma\times\mathcal{P}}}{\text{argmin}} \quad \|\mathbf{MH} - \mathbf{H}_\Sigma\|_F^2 + \tau\|\mathbf{M}\|_* \ , \tag{2.8}$$

and then taking the submatrices $\mathbf{A}_\sigma \in \mathbb{R}^{\mathcal{P}\times\mathcal{P}}$ given by $[\mathbf{A}_\sigma]_{u,u'} = [\mathbf{A}_\Sigma]_{u\sigma,u'}$. The output automaton is obtained by taking $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$, with the operators recovered from $\mathbf{A}_\Sigma$, $\boldsymbol{\alpha}_0 = \mathbf{e}_\lambda$ the indicator vector corresponding to the empty prefix, and $\boldsymbol{\alpha}_\infty = \mathbf{h}_{\mathcal{P},\lambda}$.

In order to obtain a PA as output, one can add a set of convex constraints to the search space of (2.8). In particular, this can be achieved by looking for $\mathbf{M}$ inside the set of matrices $\mathbf{P}(\mathcal{P}\Sigma, \mathcal{P}, \mathbf{h}_{\mathcal{P},\lambda}) \subseteq \mathbb{R}^{\mathcal{P}\Sigma\times\mathcal{P}}$ satisfying

1. $\mathbf{M} \geq 0$,
2. $\sum_{u',\sigma} \mathbf{M}(u\sigma, u') + \mathbf{h}_{\mathcal{P},\lambda}(u) = 1$ for every $u \in \mathcal{P}$.

Compared to the spectral method, the convex optimization approach offers more flexibility in terms selecting the model complexity, as it allows for a continuous penalty on the nuclear norm of the learned model instead of a discrete specified model size. Experiments on natural language process problems have demonstrated that this extra flexibility with respect to model complexity can lead to more accurate predictive models [10]. The method also offers more flexibility in that additional constraints upon the convex optimization can be used to enforce particular structure in the learned automaton. The enforcement that the output is a PA described above is an example of such a constraint.

The efficiency of this method largely depends upon the convex optimization routine being used. However, it is worth noting that the proximal gradient operator, used by the majority of efficient solvers, for the nuclear norm regularization condition requires the singular value decomposition of the Hankel matrix. Thus, in a majority of cases this method will at least require an SVD to be computed at each iteration during optimization, implying that it may be substantially more expensive than the spectral method.

2.2.5 Symmetric Tensor Decomposition Method

The tensor decomposition method can be applied when the target distribution is generated by a HMM [3]. The idea behind this approach is to observe that when f can be realized by a FWA, then the factorization of the Hankel matrix associated with a symbol $\sigma \in \Sigma$ becomes

$$\mathbf{H}_\sigma = \mathbf{P}\mathbf{O}_\sigma\mathbf{T}\mathbf{S}. \tag{2.9}$$

Since $\mathbf{P}$, $\mathbf{S}$, and $\mathbf{T}$ appear in the decomposition for all σ, and $\mathbf{O}_\sigma$ is diagonal, this implies that under some assumptions on the ranks of these matrices, all the $\mathbf{H}_\sigma$ admit a joint diagonalization. Stacking these matrices together yields a Hankel tensor $\mathbf{H}_{\mathcal{P},\Sigma,\mathcal{S}} \in \mathbb{R}^{\mathcal{P}\times\Sigma\times\mathcal{S}}$ with a particular structure that can be exploited to recover first the $\mathbf{O}_\sigma$ matrices, and then the transition matrix $\mathbf{T}$ and the weight vectors $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_\infty$. The algorithm described in this section implements this idea by following the symmetrization and whitening approach of [2]. This presentation is a variant of their method, which extends the method to work with arbitrary sets of prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$, and also is able to recover the set of stopping probabilities.

Again, the method needs as input sets of prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$ with $\lambda \in \mathcal{P}\cap\mathcal{S}$, and the number of states n of the target HMM, which must satisfy $n \leq |\Sigma|$. The algorithm proceeds in four stages. In its first stage, the algorithm computes a set of Hankel matrices and tensors. In particular, a third order tensor $\mathbf{H}_{\mathcal{P},\Sigma,\mathcal{S}} \in \mathbb{R}^{\mathcal{P}\times\Sigma\times\mathcal{S}}$ with entries $[\mathbf{H}_{\mathcal{P},\Sigma,\mathcal{S}}]_{u,\sigma,v} = f(u\sigma v)$, a Hankel matrix $\mathbf{H}_{\mathcal{P},\mathcal{S}} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ with entries $[\mathbf{H}_{\mathcal{P},\mathcal{S}}]_{u,v} = f(uv)$, and a Hankel matrix $\mathbf{H}^{\mathrm{p}}_{\mathcal{P},\Sigma} \in \mathbb{R}^{\mathcal{P}\times\Sigma}$ with entries $[\mathbf{H}^{\mathrm{p}}_{\mathcal{P},\Sigma}]_{u,\sigma} = f(u\sigma\Sigma^\star)$. Integrating over the different dimensions of the tensor $\mathbf{H}_{\mathcal{P},\Sigma,\mathcal{S}}$, the algorithm obtains three more matrices: $\bar{\mathbf{H}}_{\Sigma,\mathcal{S}} \in \mathbb{R}^{\Sigma\times\mathcal{S}}$ with entries $[\bar{\mathbf{H}}_{\Sigma,\mathcal{S}}]_{\sigma,v} = \sum_u f(u\sigma v)$, $\bar{\mathbf{H}}_{\mathcal{P},\mathcal{S}} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ with entries $[\bar{\mathbf{H}}_{\mathcal{P},\mathcal{S}}]_{u,v} = \sum_\sigma f(u\sigma v)$, and $\bar{\mathbf{H}}_{\mathcal{P},\Sigma} \in \mathbb{R}^{\mathcal{P}\times\Sigma}$ with entries $[\bar{\mathbf{H}}_{\mathcal{P},\Sigma}]_{u,\sigma} = \sum_v f(u\sigma v)$.

The goal of the second stage is to obtain an orthogonal decomposition of a tensor derived from $\mathbf{H}_{\mathcal{P},\Sigma,\mathcal{S}}$ as follows. Assuming $\bar{\mathbf{H}}_{\mathcal{P},\mathcal{S}}$ has rank at least n, the algorithm first finds matrices $\mathbf{Q}_\mathcal{P} \in \mathbb{R}^{n\times\mathcal{P}}$ and $\mathbf{Q}_\mathcal{S} \in \mathbb{R}^{n\times\mathcal{S}}$ such that

$$\tilde{\mathbf{H}}_{\mathcal{P},\mathcal{S}} = \mathbf{Q}_\mathcal{P}\bar{\mathbf{H}}_{\mathcal{P},\mathcal{S}}\mathbf{Q}_\mathcal{S}^\top \tag{2.10}$$

is invertible and then computes

$$\mathbf{N} = \mathbf{Q}_\mathcal{S}^\top\tilde{\mathbf{H}}^{-1}_{\mathcal{P},\mathcal{S}}\mathbf{Q}_\mathcal{P}. \tag{2.11}$$

Combining these, a matrix $\mathbf{X}_\Sigma \in \mathbb{R}^{\Sigma\times\Sigma}$ and a tensor $\mathbf{Y}_\Sigma \in \mathbb{R}^{\Sigma\times\Sigma\times\Sigma}$ are obtained as follows:

$$\mathbf{X}_\Sigma = \bar{\mathbf{H}}_{\Sigma,\mathcal{S}}\mathbf{N}\bar{\mathbf{H}}_{\mathcal{P},\Sigma} \tag{2.12}$$

$$\mathbf{Y}_\Sigma = \mathbf{H}_{\mathcal{P},\Sigma,\mathcal{S}}(\mathbf{N}^\top\bar{\mathbf{H}}^\top_{\Sigma,\mathcal{S}}, \mathbf{I}, \mathbf{N}\bar{\mathbf{H}}_{\mathcal{P},\Sigma}). \tag{2.13}$$

One can show that both $\mathbf{X}_\Sigma$ and $\mathbf{Y}_\Sigma$ are symmetric.[4] Then, assuming $\mathbf{X}_\Sigma$ is positive definite of rank at least n, we can find $\mathbf{W} \in \mathbb{R}^{\Sigma \times n}$ such that $\mathbf{W}^\top \mathbf{X}_\Sigma \mathbf{W} = \mathbf{I}$. This is used to whiten the tensor $\mathbf{Y}_\Sigma$ by taking

$$\mathbf{Z}_\Sigma = \mathbf{Y}_\Sigma(\mathbf{W}, \mathbf{W}, \mathbf{W}). \tag{2.14}$$

Next we compute the robust orthogonal eigendecomposition

$$\mathbf{Z}_\Sigma = \sum_{i \in [n]} \gamma_i \mathbf{z}_i^{\otimes 3} \tag{2.15}$$

using a power method for tensors similar to that used to compute eigendecompositions of matrices [2]. Using these robust eigenpairs $(\gamma_i, \mathbf{z}_i)$ we build a matrix $\tilde{\mathbf{O}} \in \mathbb{R}^{\Sigma \times n}$ whose ith column is $\gamma_i (\mathbf{W}^\top)^+ \mathbf{z}_i$. After a normalization operation, this will be the observation matrix of the output model.

The third stage recovers the rest of parameters (up to normalization) via a series of matrix manipulations. Let $\tilde{\mathbf{O}}_\mathcal{P} = \bar{\mathbf{H}}_{\mathcal{P},\Sigma}(\tilde{\mathbf{O}}^\top)^+ \in \mathbb{R}^{\mathcal{P} \times n}$ and $\tilde{\mathbf{O}}_\mathcal{S}^\top = \tilde{\mathbf{O}}^+ \bar{\mathbf{H}}_{\Sigma,\mathcal{S}} \in \mathbb{R}^{n \times \mathcal{S}}$. We start by taking $\tilde{\boldsymbol{\alpha}}_0^\top = \mathbf{e}_\lambda^\top \tilde{\mathbf{O}}_\mathcal{P}$ and $\tilde{\boldsymbol{\alpha}}_\infty = \tilde{\mathbf{O}}_\mathcal{S}^\top \mathbf{e}_\lambda$: respectively, the rows of $\tilde{\mathbf{O}}_\mathcal{P}$ and $\tilde{\mathbf{O}}_\mathcal{S}$ corresponding to λ. Similarly, the algorithm computes

$$\tilde{\mathbf{T}} = \tilde{\mathbf{O}}_\mathcal{P}^+ \bar{\mathbf{H}}_{\mathcal{P},\mathcal{S}} \mathbf{H}_{\mathcal{P},\mathcal{S}}^+ \tilde{\mathbf{O}}_\mathcal{P}. \tag{2.16}$$

In the last stage the model parameters are normalized as follows. Let $\tilde{\mathbf{D}}_\gamma = \mathrm{diag}(\gamma_1^2, \ldots, \gamma_n^2) \in \mathbb{R}^{n \times n}$ and $\tilde{\mathbf{D}}_\mathcal{S} = \tilde{\mathbf{O}}^\top {\mathbf{H}^\mathrm{p}_{\mathcal{P},\Sigma}}^+ \tilde{\mathbf{O}}_\mathcal{P} \in \mathbb{R}^{n \times n}$. Now, to obtain $\boldsymbol{\alpha}_\infty$ we first compute

$$\boldsymbol{\beta} = \tilde{\mathbf{D}}_\mathcal{S} \tilde{\mathbf{T}}^+ \tilde{\mathbf{D}}_\gamma \tilde{\boldsymbol{\alpha}}_\infty \tag{2.17}$$

and then let

$$\boldsymbol{\alpha}_\infty(i) = \boldsymbol{\beta}(i)/(1 + \boldsymbol{\beta}(i)). \tag{2.18}$$

The initial weights are obtained as

$$\boldsymbol{\alpha}_0^\top = \tilde{\boldsymbol{\alpha}}_0^\top \tilde{\mathbf{D}}_\Sigma^+ \tilde{\mathbf{D}}_\mathcal{S}^+, \tag{2.19}$$

[4] When working with approximate Hankel matrices and tensors this is not necessarily true. Thus one needs to consider the symmetrized versions $(\mathbf{X}_\Sigma + \mathbf{X}_\Sigma^\top)/2$ and $\sum_\rho \mathbf{Y}_\Sigma^\rho / 6$, where the sum is taken over all the permutations ρ of $\{1, 2, 3\}$.

where $\tilde{\mathbf{D}}_\Sigma = \mathbf{I} - \text{diag}(\boldsymbol{\alpha}_\infty)$. And finally, we let

$$\mathbf{O} = \tilde{\mathbf{O}}\tilde{\mathbf{D}}_\Sigma \tag{2.20}$$

and

$$\mathbf{T} = \tilde{\mathbf{D}}_\mathcal{S}\tilde{\mathbf{T}}\tilde{\mathbf{D}}_\mathcal{S}^+\tilde{\mathbf{D}}_\Sigma^+. \tag{2.21}$$

When working with empirical approximations these matrices are not guaranteed to satisfy the requirements in the definition of a HMM. In this case, a last step is necessary to enforce the constraints by projecting the parameters into the simplex [25].

As with the spectral and convex optimization methods, the symmetric tensor decomposition approach is statistically consistent (when the target automaton is an HMM) [8]. Unlike these other methods, it is guaranteed to return a stochastic automaton as HMMs are probabilistic by construction. This guarantee, however, does come at a cost: the tensor decomposition involves significantly more algebra operations compared to the spectral approach and suffers from more constraints, namely that the learned model has dimension at most $|\Sigma|$ [2]. The constraint that the model dimension is at most $|\Sigma|$ is particularly problematic in domains such as bioinformatics, where hidden states may correspond to genes (i.e., underlying encoding states in a DNA sequence) and observations to the set of nucleobases {A,C,T,G}; clearly, there are many possible (i.e. > 4) genes while in this case $|\Sigma| = 4$ [41]. The time-complexity of this method is $O((|\Sigma| + n)|\mathcal{P}||\mathcal{S}| + Rn^2)$, where R is number of tensor-power iterations used in the decomposition and again assuming that the Hankel estimates are provided as input.

3

Moment-Methods in Sequential Decision-Making

In this chapter, we move away from the general problem of sequence prediction using the method of moments to focus on a particular instantiation of the moment-method framework within the context of sequential decision-making. Here we consider an agent reasoning about the dynamics of a system, where learning distributions over possible action-observation sequences is of fundamental importance, as it allows agents to predict outcomes of actions and plan accordingly. Moment-methods are an attractive candidate for use in this context given their desirable theoretical and algorithmic properties, in particular, their efficiency and generality.

Section 3.1 will introduce the sequential decision-making framework that we employ. Section 3.2 introduces the basic moment-method model for sequential decision-making: the predictive state representation (PSR) model, which builds on ideas introduced in the previous chapter. A novel contribution of this work is the presentation of PSRs within the general framework of learning WA via the method of moments.

3.1 Sequential Decision-Making Framework

The sequential decision-making framework used in this work makes minimal assumptions about both the agent and the environment. Specifically, the framework assumes

only (1) that the environment is partially observable and Markovian with respect to some underlying state, (2) that each hidden state emits a numerical reward signal, and (3) that the agent is interested in maximizing a cumulative function of these rewards, where immediate rewards are given more weight (i.e., importance) than potential rewards far in the future. The following sections (3.1.1 and 3.1.2) elaborate on these assumptions and formalize this framework.

3.1.1 Partial State Observability

Much of sequential decision-making is concerned with settings in which agents have complete knowledge of the system state [68, 30]. In this work, however, we are interested in the more difficult task of sequential decision-making under partial state observability, i.e., where single observations do not fully characterize the state of the system. In other words, we assume that at each time-point the system is fully described by some underlying hidden, or latent, state that controls the system's dynamics and emits an observation.

Assuming partial state observability is a necessity in many interesting application domains. For example, in the field of robotics it is most often the case that agents only have access to noisy sensors, which confer some information about their environment but do not fully disambiguate their state (e.g., their global position). Moreover, assuming partial observability is non-restrictive in that the vast majority of sequential decision-making domains may be modelled as such [40]. The only environments which are not captured in this partially observable framework are those that are not Markovian with respect to any finite-dimensional state-space. However, without approximations (e.g., approximate Markovian assumptions) sequential decision-making is generally not feasible in such domains, as they could require unbounded amounts of memory [64].

Since partial observability implies that single observations are far from sufficient statistics for the state of the system, it is necessary for an agent in this setting to incorporate knowledge of its history within a particular execution trace. One solution to this problem is for the agent to simply incorporate a fixed window of

history into its plans. However, this approach lacks in generality, as it requires knowing (or discovering) the necessary window size. Even more troubling are the facts that the state-space increases combinatorially with larger window sizes and that no (non-trivial) a priori upper bound on the necessary window size can be efficiently obtained [22]. Extensions of this approach which combine different window sizes into one single model are possible [12, 22], but these extensions still lack expressive capability compared to models that explicitly account for the existence of a hidden state [12].

A more general approach, and the one that is taken in this work, is to have the agent learn a probabilistic model of the system that explicitly models the existence of a hidden state and that facilitates tracking and the prediction of future events. The model state then serves as the sufficient statistic for the system's history, and plans can be made by reasoning using the model.

3.1.2 The POMDP Model

The classic method for formalizing sequential decision-making under uncertainty, while explicitly taking into account the underlying hidden state, is to use the partially observable Markov decision process (POMDP) model. Intuitively, POMDPs are simply the generalization of HMMs that include action inputs in the system transition dynamics. Formally, a POMDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, R \rangle$, where [40]

- $\mathcal{S}$ is a set of hidden states,
- $\mathcal{A}$ is a set of actions,
- $\mathcal{O}$ is a set of observations,
- T is a set of (conditional) transition probabilities, specifying how the system transitions from a hidden state after taking some action,
- Ω specifies the emission distributions of observations from hidden states, and
- $R : \mathcal{A} \times S \to \mathbb{R}$ is a reward function.

Thus in this framework, an agent takes an action from the set $\mathcal{A}$, which induces a transition to a new hidden state (determined by T). And from this new hidden state, the agent receives an observation (determined by Ω) and a reward (determined by R). The assumption that numerical rewards are emitted from each hidden state is made in order to simplify analysis and is well-motivated in that it simply corresponds to an agent having a quantified measure of goodness at each point in time.

In addition to this formal model, in most applications using POMDPs it is necessary to maintain a vector belief-state $\mathbf{b} \in \Delta^{|\mathcal{S}|-1}$ that encodes the probability of the agent being in each hidden state (at each point in time) [40]. This belief-state serves as a sufficient statistic for the system's history and can be maintained by an agent as it tracks through a system. It is also necessary to define a Q-function

$$Q : \mathcal{A} \times \Delta^{|\mathcal{S}|-1} \to \mathbb{R}, \tag{3.1}$$

which defines the quality of a belief-state and action pair [49].[1] In this work, we formally define this Q via the recursive relation

$$Q(a, \mathbf{b}_t) = \sum_{i=0}^{|\mathcal{S}|} [\mathbf{b}_t]_i R(a, i) + \gamma \max_a \mathop{\mathbb{E}}_{\mathbf{b}_{t+1} \sim a} [Q(a, \mathbf{b}_{t+1})], \tag{3.2}$$

where $\mathbb{E}_{\mathbf{b}_{t+1} \sim a}$ is used to denote the expectation of the next belief-state given that action a is taken and γ is a discount factor in $[0, 1]$. A policy, or mapping $\pi : \Delta^{|\mathcal{S}|-1} \to \mathcal{A}$ from belief states to actions, can be easily defined using the Q-function by taking the argmax over actions when in a belief-state [40].

Intuitively, the Q-function combines (via summation) the immediate reward obtainable by taking a particular action when in a belief-state and a discounted measure of the quality of the new belief-state that will be induced by taking the action [49]. Thus an action may be desirable in cases where it leads to high immediate rewards or where it causes a transition to a desirable belief-state. The discounting of potential

[1] For completeness, we note that the Q-function is most-often defined over the underlying system states (and not belief-states) [43]. As was mentioned in section 3.1.1, these system states are often assumed to fully observable, and this is the natural definition in that setting. The Q-function is then extended to POMDP belief-states. However, since we are only concerned with partially observable domains, we omit the observable definition and define the Q-function directly over belief-states.

future rewards is motivated by the intuition that agents should give primacy to their immediate situation (and not, for example, incur irreparable physical damage for the promise of future rewards). Moreover, it is a mathematical necessity in domains that have potentially infinite planning horizons (i.e., where there is no well-defined notion of a start or stop state), since the use of discounts prevents the Q-function from diverging to infinity [43].

Given a POMDP, it is possible to use various techniques to determine an optimal sequence of decisions (i.e., a plan or policy) [49]. At a high level, these techniques usually involve learning some approximation of the Q-function. We forgo a detailed discussion of such techniques since the majority are not of direct relevance to this work and refer the reader to [49] for an extensive overview. In particular, here we are interested in both learning a POMDP-type model of a system and planning given this learned model, while POMDP planning algorithms assume that a complete model is given a priori.[2] Combining learning and planning leads us to a generalization of POMDPs for which the classic POMDP planning algorithms are not directly applicable.

3.2 Predictive State Representation

Predictive state representations (PSRs) are both an extension of the basic Hankel factorization method of moments for sequence prediction and a generalization of POMDPs. The primary difference between PSR learning and the basic Hankel factorization method being that PSRs model sequences of action-observations pairs instead of sequences of only observations. The relationship between PSR models and POMDPs is analogous to the relationship between WA and HMMs elucidated in chapter 2: PSRs are a super-set of POMDPs where the transition function, T, and observation emission functions, Ω, are combined into a (potentially) more concise

[2] In general, the learning problem for POMDPs has received relatively little attention, primarily due to the intractabilities associated with maximum likelihood learning with such complex models, as the addition of actions significantly complicates EM-style learning to the point of intractability [40]. There are some circumscribed examples of learning in this setting, e.g. using Bayesian adaptation [62]; however, these learning frameworks assume considerable a priori knowledge.

set of linear operators.[3] This relationship trivially holds by noting that POMDPs are simply HMMs with the addition of actions and decision-making. Moreover, the relative generality of WAs compared to HMMs transfers over to this decision-making setting in that any POMDP can be represented by a PSR model while the reverse does not hold [44].

As with the generic moment-method, a PSR model is constructed directly from observable quantities, in this case execution traces, without utilizing any prior information about the domain [44, 67]. PSRs thus offer an expressive and powerful framework for modelling dynamical systems without prior knowledge and provide a suitable foundation for an agnostic sequential decision-making.

It is worth noting that PSRs correspond to the simplest of the generic moment-methods described in previous sections; that is, a PSR model is learned via a rank-revealing decomposition of the Hankel matrix. Thus, the spectral method will be employed in learning PSRs (under the name of transformed PSRs or TPSRs), but neither convex optimization nor tensor decomposition will be used in the sequential decision-making setting. The tensor and convex optimization moment-methods could, in principle, be applied to sequential decision-making. However, in the sequential decision-making setting it is necessary to have fast (possibly even incremental/online) model learning, and high-dimensional observation and state spaces are the norm, rendering the tensor and convex optimization methods computationally intractable in the majority of cases. Even the spectral method which is considerably more efficient than both the tensor and convex optimization approaches nears intractability in many sequential decision-making domains, as we will demonstrate in the coming chapters.

3.2.1 The PSR Model: Independent Derivation

A PSR model is a WA that incorporates both actions and observations. However, given the additional meaning implicated by such a WA, in this section, we provide

[3] A necessary point of clarification is that the term PSR refers simultaneously to the model class and the (moment-based) learning algorithm. In situations where the meaning may be ambiguous we will refer separately to the PSR model and the PSR learning algorithm.

additional intuition and derivations independent of the general theory of the method of moments. We will elucidate the close relationship between PSRs and generic moment-methods for sequence prediction in the next section.

Formally, a PSR maintains 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 the | symbol in this case denotes that the agent "intervened" by performing the 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

$$\mathbb{P}(\tau_i) = \mathbb{P}(o_{t+1}^{k_1}, o_{t+2}^{k_2}, ..., o_{t+n}^{k_n} | a_{t+1}^{l_1}, a_{t+2}^{l_2}, ..., a_{t+n}^{l_n}). \tag{3.3}$$

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 actions are not separated from the observations with the | symbol in the definition of a history, as the sequence of actions specified in a particular 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

$$\mathbb{P}(\tau_i^{\mathcal{O}} | h_j, \tau_i^{\mathcal{A}}), \tag{3.4}$$

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 vectors of actions and observations, respectively, specified in τ_i.

It is not difficult to see that a dynamical 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) [44].

Fortunately, it has been shown that it suffices to remember only the conditional probabilities for a small (not necessarily unique) *core set* of tests, and the conditional probabilities for all other tests may be defined as functions of the conditional probabilities for the tests in this core set [44]. Perhaps more importantly, [44] has shown that it suffices to consider only linear functions of tests in a core set. That is, given a core set of tests $\mathcal{Q}$, we can compute the conditional probability of some test $\tau_i \notin \mathcal{Q}$ as

$$\mathbb{P}(\tau_i^{\mathcal{O}}|h_j, \tau_i^{\mathcal{A}}) = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_j, \mathcal{Q}^{\mathcal{A}})^\top \mathbf{r}_{\tau_i}, \tag{3.5}$$

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 $q_i \in \mathcal{Q}$. Integral to this approach is the fact that restricting the model to linear functions of tests in a core set does not preclude the modelling of non-linear dynamical systems, as the dynamics implicit in the probabilities may specify non-linear behaviours [44].

Thus, given the functions mapping tests in a 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.

Formally, a PSR model of a system is defined by $\{\mathcal{O}, \mathcal{A}, \mathcal{Q}, \mathcal{F}, \mathbf{m}_0\}$, where $\mathcal{O}$ and $\mathcal{A}$ define the possible observations and actions respectively, $\mathcal{Q}$ is a core set of tests (usually assumed to be *minimal* in terms of cardinality), $\mathcal{F}$ defines a set of linear functions mapping success probabilities of 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., $\mathbf{m}_0 = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|\mathcal{Q}^{\mathcal{A}})$). As mentioned above, we restrict $\mathcal{F}$ to contain linear functions, so 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

$$\mathbb{P}(o^k_{t+1}|h_t, a^l_{t+1}) = \mathbb{P}(Q^{\mathcal{O}}|h_t, Q^{\mathcal{A}})^\top \mathbf{M}_{a^l o^k} \mathbf{m}_\infty \tag{3.6}$$

$$= \mathbf{m}_t^\top \mathbf{M}_{a^l o^k} \mathbf{m}_\infty, \tag{3.7}$$

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

These operators map probabilities for test in the (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

$$\mathbb{P}(\tau_i^{\mathcal{O}}|h_t, \tau_i^{\mathcal{A}}) = \mathbb{P}(Q^{\mathcal{O}}|h_t, Q^{\mathcal{A}})^\top \mathbf{r}_{\tau_i} \tag{3.8}$$

$$= \mathbf{m}_t^\top \mathbf{M}_{a^{l_1} o^{k_1}} \mathbf{M}_{a^{l_2} o^{k_2}} \cdots \mathbf{M}_{a^{l_n} o^{k_n}} \mathbf{m}_\infty. \tag{3.9}$$

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

$$\mathbb{P}(o^j_{t+T}|h_t) = \mathbf{m}_t^\top (\mathbf{M}_\star)^{n-1} \mathbf{M}_{a^l o^k} \mathbf{m}_\infty, \tag{3.10}$$

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 [73].[4]

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:

$$\mathbf{m}_{t+1}^\top = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_{t+1}, \mathcal{Q}^{\mathcal{A}})^\top \tag{3.11}$$

$$= \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_t a^l o^k, \mathcal{Q}^{\mathcal{A}})^\top \tag{3.12}$$

$$= \frac{\mathbf{m}_t^\top \mathbf{M}_{a^l o^k}}{\mathbf{m}_t^\top \mathbf{M}_{a^l o^k} \mathbf{m}_\infty}. \tag{3.13}$$

[4] The computation of $\mathbf{M}_\star$ assumes an random open-loop (i.e., blind) action policy. If a non-blind policy were used then the summands would need to be weighted according to the action policy.

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

3.2.2 The PSR Model: Method of Moments Interpretation

The mapping from PSRs to generic moment-methods for sequence prediction is straightforward. First, we note that the parsing of sequences of action-observation pairs into test and histories maps readily to the parsing of a string into suffixes and prefixes, respectively. The set $\mathcal{F}$ then corresponds directly to a WA computing a stochastic language $f^{\mathrm{P}} : \Sigma^\star \to \mathbb{R}$ with $\Sigma := \mathcal{A} \times \mathcal{O}$: the linear operators, i.e. $\mathbf{M}_{ao}\ \forall ao \in \mathcal{A} \times \mathcal{O}$, corresponding to the transition operators in the WA; the $\mathbf{m}_0$ vector to the initial weights; and the $\mathbf{m}_\infty$ normalization vector to the final weights.[5] We emphasise that a PSR, as defined here, realizes a function computing prefix probabilities and not string probabilities. This makes intuitive sense in a decision-making setting, as an agent is interested in computing the probabilities of different possible future action-observation pairs given some past as opposed to computing the probability of entire trajectories.

The statement that only a finite number of operators, corresponding to the tests in a minimal core set, are necessary to characterize a system is equivalent to the statement that the Hankel matrix defined over tests and histories has rank $|\mathcal{Q}|$. The tests in the minimal core set then correspond to a column-basis of $\mathbf{H}$.

We reiterate that, given this interpretation, the conciseness of PSRs compared to POMDP models is immediately apparent. Recalling from section 2.2.1 that there are PA (and thus WA) with n states such that every HMM realizing the same distribution needs more than n states [26], we immediately have that an analogous relation holds for PSRs versus POMDPS as they are extensions of WA and HMMs, respectively, to the decision-making setting. This is a more powerful result than that provided by

[5]The reader is advised that the PSR operators defined in this work are the transposition of the usual operators used in the PSR setting. This change is trivial and is made to maintain consistency with the WA framework.

[44]. In that work they give a mapping from any POMDP to a PSR, demonstrating a constructive proof that PSRs are at least as concise. The relation outlined above shows that there cases where PSRs are more concise.

We also note that the PSR derivation elucidates a novel interpretation of the Hankel matrix factorization. First, we define $\mathbf{R} \in \mathbb{R}^{|\mathcal{Q}|\times|\mathcal{T}|}$ to be the matrix with $\mathbf{r}_{\tau_i}$ as columns, recalling that $\mathbf{r}_{\tau_i}$ defines the linear map from probabilities of tests in a minimal core set to the probability of test τ_i. Next, we define $\mathbf{Q} \in \mathbb{R}^{|\mathcal{H}|\times|\mathcal{Q}|}$ to be the matrix with rows given by the vectors $\mathbf{m}_{h_j} = \mathbb{P}(\mathcal{Q}^{\mathcal{O}}|h_j, \mathcal{Q}^{\mathcal{A}})$, the expected PSR state given a history h_j. And, we define a matrix $\mathbf{N} = \text{diag}\{\mathbf{h}_{\mathcal{H},\lambda}\} = \text{diag}\{\mathbb{P}(\mathcal{H})\}$ containing the marginal history probabilities along the diagonal. Now we have that

$$[\mathbf{R}]_{*,\tau_i} = \mathbf{M}_{\tau_i}\mathbf{m}_\infty \tag{3.14}$$

by definition, and

$$[\mathbf{Q}]_{h_j,*} = \frac{\mathbf{m}_0^\top \mathbf{M}_{h_j}}{\mathbf{m}_0 \mathbf{M}_{h_j}\mathbf{m}_\infty}. \tag{3.15}$$

Thus,

$$[\mathbf{NQ}]_{h_j,*} = \mathbf{m}_0^\top \mathbf{M}_{h_j}, \tag{3.16}$$

since $[\mathbf{N}]_{h_j,h_j} = \mathbb{P}(h_j) = \mathbf{m}_0^\top \mathbf{M}_{h_j}\mathbf{m}_\infty$. Combining these results, we see that

$$\mathbf{H} = \mathbf{PS} = \mathbf{NQR}. \tag{3.17}$$

So a rank-revealing factorization of $\mathbf{H}$ is given by $\mathbf{P} = \mathbf{NQ}$ and $\mathbf{S} = \mathbf{R}$, recalling the results of section 2.2.3 and that histories and tests are playing the roles of prefixes and suffixes, respectively, in this setting. We will refer to this as the PSR factorization of the Hankel matrix. Importantly, this perspective on the factorization reveals the relationship between the core test set view of PSRs and the more generic Hankel matrix factorization view.

3.2.3 Learning PSRs

There is a considerable amount of literature describing different approaches to learning PSRs. We provide an overview of the standard approaches, as chapter 4 describes,

in detail, the efficient compressed learning approach we propose.[6] The description of these learning approaches will, in some cases, amount to alternative descriptions and motivations of the Hankel matrix factorization approach. We include this alternative perspective in order to highlight the independent development of PSRs with respect to the more generic moment-methods. The alternative perspective also provides further insight and intuition into the Hankel factorization approach. We will provide illustrative commentary on the relationship between the generic moment-method and PSR perspectives as necessary.

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 a core set of tests (which may turn out to be non-minimal), and the PSR model is then computed in a straightforward manner given the explicit knowledge of $\mathcal{Q}$ [38, 39]. 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 [34, 16, 61]. 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 $\mathcal{Q}$ [61]. 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 [16].

More formally, we let $\mathbf{H}$ and $\mathbf{H}_{ao}$ $\forall ao \in \mathcal{A} \times \mathcal{O}$ be Hankel matrices defined over some set of histories $\mathcal{H}$ and tests $\mathcal{T}$, and we assume that the core test set $\mathcal{Q}$ is known. The discovery-based approach builds a PSR model by

$$\mathbf{m}_0^\top = \mathbf{h}_{\lambda,\mathcal{Q}}^\top, \tag{3.18}$$

$$\mathbf{m}_\infty = ([\mathbf{H}]_{*,\mathcal{Q}})^+ \mathbf{h}_{\mathcal{H},\lambda}, \tag{3.19}$$

$$\mathbf{M}_{ao} = ([\mathbf{H}]_{*,\mathcal{Q}})^+ [\mathbf{H}_{ao}]_{*,\mathcal{Q}}, \tag{3.20}$$

[6] For a slightly more detailed discussion of existing PSR learning approaches see [73].

while the subspace-based approach builds a model by

$$\boldsymbol{\beta}_0^\top = \mathbf{h}_{\lambda,\mathcal{T}}^\top \mathbf{Z}, \tag{3.21}$$

$$\boldsymbol{\beta}_\infty = (\mathbf{HZ})^+ \mathbf{h}_{\mathcal{H},\lambda}, \tag{3.22}$$

$$\mathbf{B}_{ao} = (\mathbf{HZ})^+ \mathbf{H}_{ao} \mathbf{Z}, \tag{3.23}$$

where $\mathbf{Z}$ is a matrix such that $\mathbf{J} = \mathbf{SZ}$ is invertible (recalling that $\mathbf{S}$ is the right-side term of the Hankel factorization).

In general, the complexity of the discovery-based learning approach is dominated by the combinatorial search for the core set of 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 core-test set is provided as input, the discovery-based method has complexity $O(|\mathcal{H}||\mathcal{Q}|^2)$; however, the assumption that the core-test set 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 core-test set is known (an unrealistic assumption) then $d_{\mathbf{Z}} = |\mathcal{Q}|$ in order to satisfy that $\mathbf{SZ}$ is invertible. In practice, $d_{\mathbf{Z}}$ and $\mathbf{Z}$ itself are chosen via spectral methods in order to guarantee that $\mathbf{SZ}$ is invertible (section 3.2.4 elaborates on this point).

3.2.4 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 $\boldsymbol{\beta}_0 = \mathbf{J}^\top \mathbf{m}_0$, $\boldsymbol{\beta}_\infty = \mathbf{J}^{-1}\mathbf{m}_\infty$, and $\mathbf{B}_{ao} = \mathbf{J}^{-1}\mathbf{M}_{ao}\mathbf{J}$, $\forall ao \in \mathcal{A} \times \mathcal{O}$, and these new transformed matrices constitute the TPSR model [13]. It is easy to see that the $\mathbf{J}$'s cancel out in the prediction equation (3.8) and update equation (3.11). Intuitively, TPSRs can be thought of as maintaining a predictive state upon an invertible linear transform of the state defined by the tests in a minimal core set. The final piece of a TPSR is the specification

of $\mathbf{Z}$, the projection matrix defining the subspace used during learning (recall that $\mathbf{J} = \mathbf{SZ}$). The standard method for choosing $\mathbf{Z}$ is via spectral techniques; that is, $\mathbf{Z}$ is set to be $\mathbf{V}$, the transpose of the matrix of right singular vectors (from the thin-SVD of $\mathbf{H}$).

Examining the equations for the different learning methods (i.e., (3.18) and (3.21)) and recalling the PSR specific Hankel factorisation $\mathbf{H} = \mathbf{NQR}$, we see first that for the discovery-based method, which learns a true untransformed PSR, we have that

$$[\mathbf{H}]_{*,\mathcal{Q}} = \mathbf{NQI}, \tag{3.24}$$

where $\mathbf{I}$ is the identity. In this case only core tests are in $[\mathbf{H}]_{*,\mathcal{Q}}$, and thus the core test set mapping operator $\mathbf{R}$ is replaced by the identity. Similarly for the symbol Hankels we have

$$[\mathbf{H}_{ao}]_{*,\mathcal{Q}} = \mathbf{NQM}_{ao}\mathbf{I}. \tag{3.25}$$

Thus for the discovery method

$$\begin{aligned} ([\mathbf{H}]_{*,\mathcal{Q}})^{+}[\mathbf{H}_{ao}]_{*,\mathcal{Q}} &= (\mathbf{NQ})^{+}\mathbf{NQM}_{ao} && (3.26)\\ &= \mathbf{M}_{ao}, && (3.27) \end{aligned}$$

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

$$\begin{aligned} \mathbf{B}_{ao} &= (\mathbf{HZ})^{+}\mathbf{H}_{ao}\mathbf{Z} && (3.28)\\ &= (\mathbf{NQRZ})^{+}\mathbf{NQM}_{ao}\mathbf{RZ} && (3.29)\\ &= (\mathbf{RZ})^{+}(\mathbf{NQ})^{+}\mathbf{NQM}_{ao}\mathbf{RZ} && (3.30)\\ &= (\mathbf{RZ})^{-1}\mathbf{M}_{ao}(\mathbf{RZ}), && (3.31) \end{aligned}$$

where we used the fact that $\mathbf{NQ}$ has full column-rank and that $\mathbf{RZ}$ is invertible (both by definition). This confirms that $\mathbf{B}_{ao}$ is a transformed representation of $\mathbf{M}_{ao}$ with $\mathbf{J} := \mathbf{RZ}$. Similar results hold for $\boldsymbol{\beta}_{\infty}$ and $\boldsymbol{\beta}_{0}$, showing that the subspace learning method does, in fact, return TPSRs.

3.2.5 Two Views: Factorization versus Least-Squares

We conclude the discussion of learning PSRs by relating the PSR learning approaches back to the generic method of moments learning algorithms. First, we note that there is a fundamental difference in the motivations behind PSR learning and the generic moment-methods: the PSR learning approaches phrase the problem as a least-squares regression (i.e., pseudoinverse) problem, while the prototypical moment-method approach is phrased in terms of factorizing the Hankel matrix.

More formally, if we take the subspace-learning approach with $\mathbf{Z} = \mathbf{V}$, where as usual $\mathbf{V}$ contains the right singular vectors from $\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, we have that

$$\mathbf{B}_{ao} = (\mathbf{H}\mathbf{V})^+(\mathbf{H}_{ao}\mathbf{V}). \tag{3.32}$$

And we note that

$$(\mathbf{D}^{-1}\mathbf{U}^\top)(\mathbf{H}\mathbf{V}) \tag{3.33}$$
$$= (\mathbf{D}^{-1}\mathbf{U}^\top)(\mathbf{U}\mathbf{D}\mathbf{V}^\top\mathbf{V}) \tag{3.34}$$
$$= \mathbf{I}, \tag{3.35}$$

so

$$(\mathbf{D}^{-1}\mathbf{U}^\top) = (\mathbf{H}\mathbf{V})^+. \tag{3.36}$$

Thus we see that the Hankel factorisation approach with $\mathbf{H} = (\mathbf{U}\mathbf{D})(\mathbf{V}^\top)$ is in fact the solution to the least-squares regression subspace-based PSR approach when $\mathbf{Z} = \mathbf{V}$. A TPSR with $\mathbf{Z} = \mathbf{V}$ is thus equivalent to learning a WA via the spectral method described in section 2.2.4.1.

To our knowledge, this is the first work making this subtle relationship explicit, and we hope that this will further facilitate the viewing of these two different approaches as simply alternative instantiations of a common underlying framework.

3.3 Discussion

In this chapter, we motivated and introduced the PSR framework, a moment-based learning method for sequential decision-making and a generalization of the POMDP

formalism. By leveraging the method of moments, PSRs facilitate agnostic model learning in the partially observable setting, where it would otherwise be intractable. The following chapters outline how we improve upon the PSR learning framework, vastly increasing its efficiency via compression, and describe a principled algorithm for planning using these learned compressed PSR models, since, as we will discuss, standard POMDP planning techniques do not directly generalize to the PSR setting.

Moreover, unlike previous work on PSRs, here we explicitly outline the relationship between PSR learning and the more general method of moments approach, providing new insights into the Hankel factorization approach and further elucidating the asymmetric relationship between PSRs and POMDPs.

In addition to existing work on PSRs [e.g. 44, 16, 61, 67, 73], the work in this chapter is closely related to work on observable operator models (OOMs), a conceptual predecessor to both PSRs and the moment-methods discussed in chapter 2 [37]. The primary distinction between OOMs and PSRs being that the former did not explicitly incorporate actions and assumed a priori knowledge of the a core set of tests (called characteristic events in that literature) during learning.

It is also worth noting that there a number of extensions of the PSR framework, such as extensions to deal with temporal abstraction [75] and mixed-observability [53]. And an interesting direction for future work is the characterization of these extensions within the more general context of learning WA via the method of moments, as such a characterization could potentially reveal novel insights in applications domains that do not require the modelling of actions (e.g., natural language processing).

Lastly, we note that the Q-function based sequential decision-making framework we present is closely related to a number of techniques within the field of reinforcement learning [68], a widely-used framework for formalising sequential decision-making [40, 43, 49, 56]. Indeed, the technique we present can be viewed as an instantiation of the reinforcement learning framework, as the key assumptions underlying reinforcement learning are identical to those used in this work (elucidated in section 3.1) [68]. Moreover, POMDPs are traditionally presented within the context of reinforcement learning [40]. Our work differs from classic reinforcement learning in that

we do not emphasize the dynamic programming view of sequential decision-making, where Q-function (or a variant thereof) is specified via Bellman's equation [68], and thus, we opted to present our algorithms within the more general context of sequential decision-making. Nonetheless, many of the ideas in this chapter and chapter 6 are influenced by reinforcement learning literature and many of the works cited are presented within the reinforcement learning framework [e.g. 13, 27, 36, 39, 40, 43]. We do not attempt to enumerate all relevant works within that field, as it a vast research area; however, we refer the interested reader to [68] for a thorough presentation of standard reinforcement learning methods.

4

Compressed Predictive State Representation

In this chapter, we present our novel compressed predictive state representation (CPSR) learning algorithm. 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. In section 4.1, we provide motivation and the foundations for our compressed learning algorithm. Section 4.2 describes the learning algorithm in detail, specifying how a model can be learnt from a batch of data and also incrementally updated in an online manner.

4.1 Intuition and Motivation

Despite the fact that non-compressed subspace-based algorithms, such as TPSR, can specify a small dimension for a transformed space, there are still a number of computational limitations. To begin, TPSRs require that the $|\mathcal{T}| \times |\mathcal{H}|$ matrix, $\mathbf{H}$, be estimated in its entirety, and that the $\mathbf{H}_{ao}$ matrices be partially estimated as well. Moreover, since the naive TPSR approach must compute a spectral decomposition of $\mathbf{H}$ it has computational complexity $O(|\mathcal{H}||\mathcal{T}||\mathcal{Q}|)$, in the batch (and incremental mini-batch) setting, assuming $\mathbf{H}$ is 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 chapter 7, 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 $|Z|$.[1,2] In order to circumvent these computational constraints (and provide a form of regularization), the CPSR learning algorithm we propose 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:

$$\mathbf{X} = \Phi\mathbf{Y}, \tag{4.1}$$

where Φ is a $d \times m$ *Johnson-Lindenstrauss matrix* (i.e., a matrix satisfying the Johnson-Lindenstrauss lemma) [11]. 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 Φ are discussed in chapter 7. It is worth noting that in our case, the matrix multiplication in (4.1) is in fact performed "online", and the matrices corresponding to $\mathbf{X}$ and Φ are never explicitly held in memory (details in section 4.2).

The fidelity of this technique depends only on 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:

$$k \geq ||\mathbf{y}_i||_0 \forall \mathbf{y}_i \in \mathbf{Y}, \tag{4.2}$$

where $||\cdot||_0$ denotes Donoho's zero "norm" (which simply counts number of non-zero entries in the vector). In the case of PSRs (and moment-methods in general), we have that $\mathbf{Y} := \mathbf{H}$; that is, we are compressing the Hankel matrix and thus require that $\mathbf{H}$ is sparse.

[1]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.

[2]It is worth noting that no explicit bounds on the sample complexity of PSR learning have been elucidated. However, the sample complexity bounds of [34] 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).

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 seems realistic in many domains. For example, in the PocMan domain described in section 7.2, 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 chapter 7, certain noisy observation models induce sparsity that can be exploited by this approach.

Of course, there are other methods for reducing the size and computational load of learning TPSRs through the use of domain specific feature construction (e.g., kernel methods) [16, 13]. A major contribution of the algorithm presented here is to relieve the requirement for "specialized" feature selection and push towards an out-of-the-box agnostic learning algorithm.

4.2 A Compressed Learning Algorithm PSRs

We now formally present the CPSR algorithm. Section 4.2.2 describes how to incrementally update a learned model with new data efficiently for deployment in online settings.

4.2.1 Batch Learning of CPSRs

To begin, we define two functions: $\phi_{\mathcal{T}} : \mathcal{T} \to \mathbb{R}^{d_{\mathcal{T}}}$ and $\phi_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^{d_{\mathcal{H}}}$. These functions can be viewed as extracting features of tests and histories, respectively. To connect these feature mappings with the compressed sensing motivation, we can view the features as mapping to columns of independent random full-rank Johnson-Lindenstrauss (JL) projection matrices $\Phi_{\mathcal{T}} \in \mathbb{R}^{d_{\mathcal{T}} \times |\mathcal{T}|}$ and $\Phi_{\mathcal{H}} \in \mathbb{R}^{d_{\mathcal{H}} \times |\mathcal{H}|}$, 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. It is also worth noting that we require $\phi_{\mathcal{T}}(\tau_i) = \phi_{\mathcal{T}}(\tau_j)$ for $\tau_i = \tau_j$ (and similarly for $\phi_{\mathcal{H}}$) for this specification to be well-defined.

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-observation pairs in z correspond exactly to h_j, with $\mathbb{I}_{\tau_j}(z)$ defined analogously. We then 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 $\tau_i \in \mathcal{T}$.[3]

Given a batch of training trajectories Z we compute:[4]

$$\hat{\boldsymbol{\theta}}_{\mathcal{H},\lambda} := \Phi_{\mathcal{H}} \hat{\mathbf{h}}_{\mathcal{H},\lambda} \tag{4.3}$$

$$= \sum_{z \in Z} \sum_{h_j \in \mathcal{H}} \mathbb{I}_{h_j}(z) \phi_{\mathcal{H}}(h_j), \tag{4.4}$$

$$\hat{\boldsymbol{\theta}}_{\lambda,\mathcal{T}} := \Phi_{\mathcal{T}} \hat{\mathbf{h}}_{\lambda,\mathcal{T}} \tag{4.5}$$

$$= \sum_{z \in Z} \sum_{\tau_i \in \mathcal{T}} \mathbb{I}_{\tau_i}(z) \phi_{\mathcal{T}}(\tau_i), \tag{4.6}$$

$$\hat{\boldsymbol{\Theta}} := \Phi_{\mathcal{H}} \hat{\mathbf{H}} \Phi_{\mathcal{T}}^{\top} \tag{4.7}$$

$$= \sum_{z \in Z} \sum_{\tau_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j,\tau_i}(z) \left[\phi_{\mathcal{H}}(h_j) \oplus \phi_{\mathcal{T}}(\tau_i) \right]. \tag{4.8}$$

Next, we compute the $\hat{\mathbf{U}}\hat{\mathbf{D}}\hat{\mathbf{V}}^{\top}$ rank-d' truncated SVD of $\hat{\boldsymbol{\Theta}}$:

$$(\hat{\mathbf{U}}, \hat{\mathbf{D}}, \hat{\mathbf{V}}) = \text{SVD}(\hat{\boldsymbol{\Theta}}). \tag{4.9}$$

Given these matrices we can construct $\mathbf{c}_0$ and $\mathbf{c}_\infty$, the compressed and transformed estimates of $\mathbf{m}_0$ and $\mathbf{m}_\infty$, respectively:

$$\mathbf{c}_0^{\top} = \hat{\boldsymbol{\theta}}_{\lambda,\mathcal{T}}^{\top} \hat{\mathbf{V}}, \tag{4.10}$$

[3] In this work we use $k = 0$. That is we do not use the suffix history estimation algorithm [74], 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.

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

$$\mathbf{c}_\infty = \hat{\mathbf{D}}^{-1}\hat{\mathbf{U}}^\top \hat{\boldsymbol{\theta}}_{\mathcal{H},\lambda}. \tag{4.11}$$

We note, however, that in some cases this estimation method for $\mathbf{c}_0$ will not suffice. In particular, many sequential decision-making domains do not have a well-defined start-state, and thus the start-states of training trajectories are in fact arbitrary, making the initial prediction (i.e., weight) vector $\mathbf{c}_0$ biased according to how the start-states are arbitrarily specified. For instance, training trajectories could come from random-restarts in a domain or could be constructed by parsing one long exploration sequence into smaller training trajectories. In these cases, there is in fact no well-defined start-state but (4.10) will provide an initial prediction vector that is biased in that it will assume the arbitrarily specified start-states have meaning.

In these settings, it is advantageous to learn an arbitrary feasible state as the starting state [16]. In other words, it is advantageous to learn an initial prediction vector using information from all histories and not just λ, the empty history. To do this, we slightly modify the above approach and specify that $\lambda \in \mathcal{H}$; i.e., we make λ a member of our history set. Then we learn an arbitrary feasible state $\mathbf{c}_*$ via

$$\mathbf{c}_* = \hat{\mathbf{U}}\hat{\mathbf{D}}\mathbf{1}, \tag{4.12}$$

where $\mathbf{1} = (1, 1, ..., 1)^\top \in \mathbb{R}^d$.

Of course this introduces an extra degree of uncertainty in our model, as $\mathbf{c}_*$ does not correspond to any particular model state; it, in fact, corresponds to a linear mixtures of all feasible states. The uncertainty in our state estimate should decrease, however, as we update and track through our system and the process mixes [16]. 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 [16].

Using the SVD of $\hat{\boldsymbol{\Theta}}$, 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 $\hat{\mathbf{\Theta}}$, we compute, for each $ao \in \mathcal{A} \times \mathcal{O}$:

$$\mathbf{C}_{ao} = \sum_{z \in Z} \sum_{\tau_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j,ao,\tau_i}(z) \left[\left(\hat{\mathbf{D}}^{-1} \hat{\mathbf{U}}^\top \phi_{\mathcal{H}}(h_j) \right) \oplus \left(\phi_{\mathcal{T}}(\tau_i) \hat{\mathbf{V}} \right) \right]. \tag{4.13}$$

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 SVD. Since we are performing SVD on a compressed matrix, the computational complexity is uncoupled from the number of tests and histories in the set of observed trajectories Z. Recalling that L is the max length of a trajectory in Z and that $|Z|$ denotes the number of trajectories in the training set Z, this approach has a computational complexity of

$$O\left(L|Z|d_{\mathcal{T}}d_{\mathcal{H}} + d_{\mathcal{H}}d_{\mathcal{T}}^2\right) = O\left(L|Z|\right) \tag{4.14}$$

since $d_{\mathcal{T}}$ and $d_{\mathcal{H}}$ are user-specified constants[5] (assuming the standard cubic computational cost for the SVD). Without compression (i.e., with naive TPSR), a computational cost of

$$O\left(L|Z| + |\mathcal{H}||\mathcal{T}|d_{TPSR}\right) = O\left(L^2|Z|^2\right) \tag{4.15}$$

is incurred, where d_{TPSR} is the chosen truncated SVD dimension for the TPSR model. Note that these bounds differ from previous bound in that they do not assume that $\mathbf{H}$ is given as an input. In addition, we have that the memory footprint of CPSR is

$$O(d_{\mathcal{H}}d_{\mathcal{T}} + |\mathcal{A}||\mathcal{O}|d_{\mathcal{T}}^2) = O(|\mathcal{A}||\mathcal{O}|), \tag{4.16}$$

while for naive TPSR the memory footprint is

$$O(|\mathcal{H}||\mathcal{T}| + |\mathcal{A}||\mathcal{O}|d_{TPSR}^2) = O(L^2|Z|^2). \tag{4.17}$$

[5]In general, the the user has a great deal of flexibility in setting these constants but not absolute freedom, as they should be at least logarithmic in the true (unknown) dimension of the system and linear in the sparsity of the system in order to guarantee good performance. Chapter 5 describes these issues in more detail.

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 chapter 5).

4.2.2 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' [13]. This is especially useful in that it facilitates alternating model learning and non-blind (i.e., non-random) exploration 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, [52] show that non-blind sampling approaches can lead to better planning results in a small sample setting.[6]

Briefly, we obtain a new $\hat{\mathbf{\Theta}}$ estimate and update our $\hat{\boldsymbol{\theta}}_{\mathcal{H},\lambda}$ and $\hat{\boldsymbol{\theta}}_{\lambda,\mathcal{T}}$ estimates using using (4.3) and (4.5) with Z'. Next, we update our SVD matrices, given our additive update to $\hat{\mathbf{\Theta}}$, using the methods of [17]. The $\mathbf{c}_0$ and $\mathbf{c}_\infty$ vectors are then re-computed exactly as in equations (4.10) and (4.11).

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

$$\begin{aligned}\mathbf{C}_{ao}^{\text{new}} = &\sum_{z \in Z'} \sum_{\tau_i, h_j \in \mathcal{T} \times \mathcal{H}} \mathbb{I}_{h_j, ao, \tau_i}(z) \left[\left(\hat{\mathbf{D}}_{\text{new}}^{-1} \hat{\mathbf{U}}_{\text{new}}^\top \phi_{\mathcal{H}}(h_i) \right) \oplus \left(\phi_{\mathcal{T}}(\tau_i) \hat{\mathbf{V}}_{\text{new}} \right) \right] \\ &+ \hat{\mathbf{D}}_{\text{new}}^{-1} \hat{\mathbf{U}}_{\text{new}}^\top \hat{\mathbf{U}}_{\text{old}} \hat{\mathbf{D}}_{\text{old}} \mathbf{C}_{ao}^{\text{old}} \hat{\mathbf{V}}_{\text{old}}^\top \hat{\mathbf{V}}_{\text{new}} \end{aligned} \tag{4.18}$$

[6] 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 chapter 7 for details.

The first term in (4.18) 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.

4.3 Discussion

In this chapter, we introduced the CPSR learning algorithm, a moment-based learning algorithm for sequential decision-making which integrates methods from compressed sensing in order to drastically increase efficiency. We described how this algorithm could be employed to learn in both batch and incremental/online regimes and provided complexity analysis making the computational benefits of this approach explicit. Chapter 5 analyses how the use of compression impacts the predictive accuracy of the learned model.

The CPSR algorithm is closely related to work on using features or kernel embeddings with PSRs [16, 13, 15], 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 $\mathbf{H}$; that is, the standard feature-based methods assume features that are not compressive [16, 13, 15].

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. 16, 13, 15] the features are used to facilitate learning in domains with continuous or structured observation spaces.

Since the general PSR learning framework assumes discrete observations, decomposing a continuous domain via feature extraction is necessary for learning in that

setting. Moreover, [15] 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 [15].

An interesting open question is how random projections, or related techniques, can be combined with these feature-based methods, allowing for efficient learning in continuous domains. The CPSR learning approach could certainly be combined (in a straightforward manner) with standard feature-based learning. However, it is unclear how sparse these feature spaces are, so such a combination would require novel theoretical analysis.

5

Theoretical Analysis of CPSR Learning

In the following section, we describe theoretical properties of the CPSR learning approach presented in chapter 4. Our analysis of 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.

5.1 Consistency of the Learning Approach

The following adapts the results of [16] 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 core test set $|\mathcal{Q}|$). We then describe the implications of this result for the case where we are in fact projecting down to a dimension smaller than $|\mathcal{Q}|$.

5.1.1 Consistency in the Non-Compressed Setting

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

map:

$$\mathbb{P}(Q^{\mathcal{O}}|h_t, Q^{\mathcal{A}})^\top \mathbf{r}_i = \mathbb{P}(\tau_i^{\mathcal{O}}|h_t, \tau_i^{\mathcal{A}}). \tag{5.1}$$

Supposing that $d_{\mathcal{T}} \geq |\mathcal{Q}|$ and $d_{\mathcal{H}} \geq |\mathcal{Q}|$ (recalling that these are the dimensions of the feature/compression matrices), we have

$$\mathbf{c}_0^\top = \mathbf{m}_0^\top (\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})^{-1}, \tag{5.2}$$

$$\mathbf{c}_\infty = (\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})\mathbf{m}_\infty, \tag{5.3}$$

$$\mathbf{C}_{ao} = (\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})\mathbf{M}_{ao}(\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})^{-1}. \tag{5.4}$$

$$\tag{5.5}$$

That is, we simply recover a TPSR where $\mathbf{J} = (\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})$, and it has been shown that the above implies a *consistent* learning algorithm [16, 13]. We note that $\Phi_{\mathcal{T}}$ appears in these consistency equations, while $\Phi_{\mathcal{H}}$ does not, emphasizing the different roles these two matrices occupy. This difference will play an important role in the theoretical analysis below.

5.1.2 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 (5.2)-(5.4) no longer hold as $(\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})$ is no longer invertible (it is in fact, no longer square). The primary focus of our theoretical analysis is the effect of this fact, i.e. $(\mathbf{R}\Phi_{\mathcal{T}}^\top \mathbf{V})$ not being invertible. We show how we can view $\Phi_{\mathcal{T}}$ 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 $\Phi_{\mathcal{H}}$ when $d_{\mathcal{H}} < |\mathcal{Q}|$, as in that setting it is no longer possible to remove $\Phi_{\mathcal{H}}$ from the consistency equations (5.2)-(5.4). From the perspective of regression, $\Phi_{\mathcal{H}}$ can be viewed as compressing the number of samples, while $\Phi_{\mathcal{T}}$ can be viewed as compressing the features. For clarity, we discuss the effects of these different compressions independently. In particular, we provide detailed analysis of how compressing tests (i.e., features) affects the implicit linear regression performed, and following this, we briefly discuss how compressing the histories (i.e., samples) during regression impacts performance.

5.2 Effects of Compressing Tests

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

5.2.1 Preliminaries

This approach is justified by recalling that in equations (4.11) and (4.13) of our learning algorithm we are in fact performing implicit compressed linear regression. That is, for $(\hat{\mathbf{U}}, \hat{\mathbf{D}}, \hat{\mathbf{V}}) = \text{SVD}(\hat{\boldsymbol{\Theta}})$:

$$\hat{\mathbf{D}}^{-1}\hat{\mathbf{U}}^{\top} = (\hat{\boldsymbol{\Theta}}\hat{\mathbf{V}})^{+} \tag{5.6}$$

$$= (\Phi_{\mathcal{H}}\hat{\mathbf{H}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top})^{+}. \tag{5.7}$$

Following the discussion in the previous section and to avoid unnecessary complication, we assume $\Phi_{\mathcal{H}}$ has orthonormal columns (i.e., is not compressive) while analysing the effects of compressing the tests (section 5.3 will discuss the $\Phi_{\mathcal{H}}^{\top}\Phi_{\mathcal{H}} \neq \mathbf{I}$ case). In the case where $\Phi_{\mathcal{H}}$ has orthonormal columns, we see that

$$\begin{aligned}
\mathbf{C}_{ao} &= (\Phi_{\mathcal{H}}\hat{\mathbf{H}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top})^{+}(\Phi_{\mathcal{H}}\hat{\mathbf{H}_{ao}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top}) \\
&= (\hat{\mathbf{H}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top})^{+}(\Phi_{\mathcal{H}})^{+}\Phi_{\mathcal{H}}(\hat{\mathbf{H}_{ao}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top}) \\
&= (\hat{\mathbf{H}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top})^{+}(\hat{\mathbf{H}_{ao}}\hat{\mathbf{V}}\Phi_{\mathcal{T}}^{\top})
\end{aligned}$$

Moreover, we ignore the $\hat{\mathbf{V}}$ 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. Thus, under the assumption that $d' = d_{\mathcal{T}}$, we have that

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow \mathbf{A}\hat{\mathbf{V}}\mathbf{x} = \hat{\mathbf{V}}\mathbf{b} \tag{5.8}$$

holds, since $\hat{\mathbf{V}}$ is full rank and square for $d' = d_{\mathcal{T}}$. Thus, the appearance of $\hat{\mathbf{V}}$ in the pseudoinverse is inconsequential in an analysis of the effect of compressing prior to regression.

To simplify the analysis one step further, assume that our test set is a core test set $\mathcal{Q}$. Therefore, random projections are applied on $[\hat{\mathbf{H}}]_{*,\mathcal{Q}}$ and $[\hat{\mathbf{H}}_{ao}]_{*,\mathcal{Q}}$ matrices. By the results of Section 5.1, this first projection leads to a consistent model, i.e. a model that is a linear transform of the model learned directly from $[\hat{\mathbf{H}}]_{*,\mathcal{Q}}$ and $[\hat{\mathbf{H}}_{ao}]_{*,\mathcal{Q}}$ matrices, since $\hat{\mathbf{U}}^\top \Phi_{\mathcal{T}} \mathbf{R}$ is invertible with probability 1 when the projected dimension is equal to $|\mathcal{Q}|$ [16]. The assumption that we work with the $[\hat{\mathbf{H}}]_{*,\mathcal{Q}}$ and $[\hat{\mathbf{H}}_{ao}]_{*,\mathcal{Q}}$ 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.

Recall:

$$\mathbf{M}_{ao} = ([\mathbf{H}]_{*,\mathcal{Q}})^{+}[\mathbf{H}_{ao}]_{*,\mathcal{Q}}, \quad \mathbf{m}_{\infty} = ([\mathbf{H}]_{*,\mathcal{Q}})^{+}\mathbf{h}_{\mathcal{H},\lambda}. \tag{5.9}$$

Since $\mathcal{Q}$ is a core test set, the above is a PSR. Assume we have enough histories in $\mathcal{H}$ such that matrices are full rank. Defining $[\mathbf{H}]^{\top}_{h,\mathcal{Q}}$ and $[\mathbf{H}]^{\top}_{hao,\mathcal{Q}}$ to be the vectors containing the joint probabilities of all test in the core set and fixed history h, we have (by the linearity of PSRs):

$$\forall h : [\mathbf{H}]_{hao,\mathcal{Q}} = [\mathbf{H}]_{h,\mathcal{Q}}\mathbf{M}_{ao}, \quad [\mathbf{H}]_{h,\lambda} = [\mathbf{H}]_{h,\mathcal{Q}}\mathbf{m}_{\infty}. \tag{5.10}$$

Thus reiterating the foundations of PSR learning (with slightly altered notation), one can think of finding the $\mathbf{M}_{ao}$ and $\mathbf{m}_{\infty}$ parameters as regression problems, having the estimates $[\hat{\mathbf{H}}]_{h,\mathcal{Q}}$ of $[\mathbf{H}]_{h,\mathcal{Q}}$ as noisy input features. We also have noisy observations $[\hat{\mathbf{H}}]_{hao,\mathcal{Q}}$ and $[\hat{\mathbf{H}}]_{h,\lambda}$ of the outputs $[\mathbf{H}]_{hao,\mathcal{Q}}$ and $[\mathbf{H}]_{h,\lambda}$, respectively. Since the sample set 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 estimation error (variance) at the cost of a controlled approximation error (bias). And we get the added benefits that working in the compressed space also helps with the computation complexity of the algorithm. 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.

5.2.2 Error of One Step Regression

When the size of the projections is smaller than the size of the core test 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 [46, 47, 28]. 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. 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 $1 - \delta$:

$$\|f(\mathbf{x}) - \hat{f}_d(\mathbf{x})\|_{\rho(\mathbf{x})} \leq \epsilon(n, D, d, \|\mathbf{w}\|^2, \|\mathbf{x}\|^2_{\rho(\mathbf{x})}, \|b(\mathbf{x})\|^2_{\rho(\mathbf{x})}, \sigma_\eta^2, \delta). \tag{5.11}$$

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 [46] and [47] 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 show that $\|\mathbf{w}\|^2$ can be bounded well below its dimension.

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, $[\mathbf{H}]_{h,\mathcal{Q}}$ and $[\mathbf{H}]_{hao,\mathcal{Q}}$ are k-sparse. Given that the empirical estimates of zero elements in these vectors are not noisy and thus zero, for $\Delta_x = [\hat{\mathbf{H}}]_{h,\mathcal{Q}} - [\mathbf{H}]_{h,\mathcal{Q}}$ we have that Δ_x is at least k-sparse. A similar argument applies to $\Delta_y = [\hat{\mathbf{H}}]_{hao,\mathcal{Q}} - [\mathbf{H}]_{hao,\mathcal{Q}}$.

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.[1]

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 $\mathbf{M}_{ao}[\Phi_{i,*}]^\top$ 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.

Let $[\mathbf{A}]_{-i,*}$ be matrix $\mathbf{A}$ with the ith row removed (with an analogous definition for columns). We have the following:

Theorem 1 *Let $\mathcal{H}$ be a large collection of sampled histories according to ρ, and let $\Phi \in \mathbb{R}^{d\times|\mathcal{Q}|}$ be an i.i.d. normal random projection: $\Phi_{ij} \sim \mathcal{N}(0, 1/d)$. We observe noisy estimate $[\hat{\mathbf{H}}]_{h,\mathcal{Q}} = [\mathbf{H}]_{h,\mathcal{Q}} + \Delta_x$ of input and $[\hat{\mathbf{H}}]_{hao,\mathcal{Q}} = [\mathbf{H}]_{hao,\mathcal{Q}} + \Delta_y$ of the*

[1] The core modifications necessary are analogous to those used made in [1] to adapt the Johnson-Lindenstrauss lemma to more general random matrices.

output, where we assume that the 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 $E_{\rho(h)}[[\mathbf{H}]_{hao,\mathcal{Q}}^\top [\mathbf{H}]_{hao,\mathcal{Q}}]$. Choose $1 \leq m \leq |\mathcal{Q}|$ such that $\sigma_m^2 \leq 1$ and define $\nu = \sum_{i=m+1}^{|\mathcal{Q}|} \sigma_i^2$. For $1 \leq i \leq d$, define:

$$\mathbf{u}_i = ([\hat{\mathbf{H}}]_{*,\mathcal{Q}}[\Phi^\top]_{*,-i})^+ [\hat{\mathbf{H}}_{ao}]_{*,\mathcal{Q}}[\Phi^\top]_{*,i}.$$

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

$$[\mathbf{C}_{ao}]_{*,i} = [\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,i-1}, 0, \mathbf{u}_{i,i}, \mathbf{u}_{i,i+1}, \ldots, \mathbf{u}_{i,d-1}]^\top.$$

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

$$\left\| ([\mathbf{H}]_{h,\mathcal{Q}}\Phi^\top)\mathbf{C}_{ao} - [\mathbf{H}]_{hao,\mathcal{Q}}\Phi^\top \right\|_{\rho(h)} \leq \sqrt{d}\epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/4d), \quad (5.12)$$

where:

$$w^2 = \|\mathbf{M}_{ao}\|^2(m + 4\sqrt{m}\ln(4d/\delta)), \quad (5.13)$$

$$x^2 = \|[\mathbf{H}]_{h,\mathcal{Q}}^\top\|_{\rho(h)}^2, \quad (5.14)$$

$$b^2 = \nu + 4\sqrt{\nu}\ln(4d/\delta), \quad (5.15)$$

$$\sigma_\eta^2 = \frac{4k\ln(4|\mathcal{Q}|/\delta)}{d}\sigma_y^2 + w^2\sigma_x^2. \quad (5.16)$$

Proof With eigenvalue decomposition we have $E_{\rho(h)}[[\mathbf{H}]_{hao,\mathcal{Q}}[\mathbf{H}]_{hao,\mathcal{Q}}^\top] = \mathbf{V}\mathbf{D}\mathbf{V}^\top$, 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 element set to 1 and 0 elsewhere. For all $1 \leq i \leq d$, define: $[\tilde{\Phi}]_{i,*} = [\Phi]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top$ and $[\Phi']_{i,*} = [\Phi]_{i,*}\mathbf{V}$. Note that since $\mathbf{V}$ is an orthonormal basis and $[\Phi]_{i,*}$ is i.i.d. normal, $[\Phi']_{i,*}$ will also have an i.i.d. normal distribution with the same covariance.

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

$$\|[\tilde{\Phi}]_{i,*}^\top\|^2 = [\Phi]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top\mathbf{V}\mathbf{I}_m\mathbf{V}^\top[\Phi]_{i,*}^\top \quad (5.17)$$

$$= [\Phi]'_{i,*}\mathbf{I}_m[\Phi']_{i,*}^\top = \sum_{j=1}^{m}([\Phi']_{i,j})^2 \quad (5.18)$$

$$\leq m + 4\sqrt{m}\ln(4d/\delta). \quad (5.19)$$

The tail bound in last line is union bounding over a corollary of Lemma 1 in [42]. The bias induced by using $[\tilde{\Phi}]_{i,*}$ can be bounded as well. Define $b(h) = [\mathbf{H}]_{hao,\mathcal{Q}}[\Phi]_{i,*}^\top - [\mathbf{H}]_{hao,\mathcal{Q}}[\tilde{\Phi}]_{i,*}^\top$. With probability no less than $1 - \delta/4$ for all $1 \leq i \leq d$:

$$\begin{aligned}
\|b(h)\|_{\rho(h)}^2 &= E_{\rho(h)}[([\Phi]_{i,*} - [\tilde{\Phi}]_{i,*})[\mathbf{H}]_{hao,\mathcal{Q}}[\mathbf{H}]_{hao,\mathcal{Q}}^\top(\Phi_i - [\tilde{\Phi}]_{i,*})^\top] && (5.20)\\
&= ([\Phi]_{i,*} - [\tilde{\Phi}]_{i,*})\mathbf{V}\mathbf{D}\mathbf{V}^\top([\Phi]_{i,*} - [\tilde{\Phi}]_{i,*})^\top && (5.21)\\
&= ([\Phi]_{i,*} - [\Phi]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top)\mathbf{V}\mathbf{D}\mathbf{V}^\top([\Phi]_{i,*} - [\Phi]_{i,*}\mathbf{V}\mathbf{I}_m\mathbf{V}^\top)^\top && (5.22)\\
&= [\Phi]_{i,*}\mathbf{V}(\mathbf{I} - \mathbf{I}_m)\mathbf{D}(\mathbf{I} - \mathbf{I}_m)\mathbf{V}^\top[\Phi]_{i,*}^\top && (5.23)\\
&= [\Phi']_{i,*}(\mathbf{I} - \mathbf{I}_m)\mathbf{D}(\mathbf{I} - \mathbf{I}_m)[\Phi']_{i,*}^\top && (5.24)\\
&= \sum_{j=m+1}^{|\mathcal{Q}|}([\Phi']_{ij})^2\sigma_j^2 && (5.25)\\
&\leq \nu + 4\sqrt{\nu}\ln(4d/\delta). && (5.26)
\end{aligned}$$

The tail bound again is due to Lemma 1 in [42] using the assumption $\sigma_m^2 \leq 1$. Using the above bounds, we have for for all $1 \leq i \leq d$:

$$\forall h : [\mathbf{H}]_{hao,\mathcal{Q}}[\Phi]_{i,*}^\top = [\mathbf{H}]_{hao,\mathcal{Q}}[\tilde{\Phi}]_{i,*}^\top + b(h) = [\mathbf{H}]_{h,\mathcal{Q}}(\mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^\top) + b(h). \quad (5.27)$$

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

By definition, $\mathbf{u}_i$ is the COLS estimate with input $[\hat{\mathbf{H}}]_{*,\mathcal{Q}}$, target $[\hat{\mathbf{H}}_{ao}]_{*,\mathcal{Q}}[\Phi]_{i,*}^\top$, and projection $[\Phi]_{-i,*}$. But in order to use the bound of Equation 5.11, 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:

$$\begin{aligned}
[\hat{\mathbf{H}}]_{hao,\mathcal{Q}}[\Phi]_{i,*}^\top &= [\mathbf{H}]_{hao,\mathcal{Q}}[\Phi]_{i,*}^\top + \Delta_y[\Phi]_{i,*}^\top && (5.28)\\
&= [\mathbf{H}]_{hao,\mathcal{Q}}[\tilde{\Phi}]_{i,*}^\top + b(h) + \Delta_y[\Phi]_{i,*}^\top && (5.29)\\
&= ([\hat{\mathbf{H}}]_{h,\mathcal{Q}} - \Delta_x)(\mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^\top) + b(h) + \Delta_y[\Phi]_{i,*}^\top && (5.30)\\
&= ([\hat{\mathbf{H}}]_{h,\mathcal{Q}})(\mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^\top) + b(h) + (\Delta_y[\Phi]_{i,*}^\top - \Delta_x\mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^\top). && (5.31)
\end{aligned}$$

And thus the sample points are:

$$[\hat{\mathbf{H}}]_{h,\mathcal{Q}} \rightarrow ([\hat{\mathbf{H}}]_{h,\mathcal{Q}})(\mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^{\top}) + b(h) + (\Delta_y[\Phi]_{i,*}^{\top} - \Delta_x \mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^{\top}). \tag{5.32}$$

The effective noise $\Delta_y[\Phi]_{i,*}^{\top} - \Delta_x \mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^{\top}$ has mean 0. Since Δ_y is k-sparse and $\|\mathbf{M}_{ao}[\tilde{\Phi}]_{i,*}^{\top}\|^2 \leq w^2$, the variance of the effective noise term is bounded by $\max_j(\Phi_{ij})^2 k\sigma_y^2 + w^2\sigma_x^2$. 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 5.11. With probability no less than $1 - \delta/4$, for all $1 \leq i \leq d$:

$$\left\|([\mathbf{H}]_{h,\mathcal{Q}}[\Phi^{\top}]_{*,-i})\mathbf{u}_i - [\mathbf{H}]_{hao,\mathcal{Q}}[\Phi]_{i,*}^{\top}\right\|_{\rho(h)} \leq \epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/4d). \tag{5.33}$$

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

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 [28], 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* $o \in \mathcal{O}$*:*

$$\left\|([\mathbf{H}]_{h,\mathcal{Q}}\Phi^{\top})\mathbf{C}_{ao} - [\mathbf{H}]_{hao,\mathcal{Q}}\Phi^{\top}\right\|_{\rho(h)} \leq \sqrt{d}\epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, w^2, x^2, b^2, \sigma_\eta^2, \delta/(4d|\mathcal{A}||\mathcal{O}|)), \tag{5.34}$$

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

5.2.2.1 Error of the compressed normalizer

The $\mathbf{c}_\infty$ operator is the normalization operator for the compressed space. Therefore, for any history h, $[\mathbf{H}]_{h,\mathcal{Q}}\Phi^\top \mathbf{c}_\infty$ should equal $[\mathbf{H}]_{h,\lambda}$. 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 $[\hat{\mathbf{H}}]_{*,\mathcal{Q}} = [\mathbf{H}]_{*,\mathcal{Q}} + \Delta_x$ of input and $\hat{\mathbf{h}}_{\mathcal{H},\lambda} = \mathbf{h}_{\mathcal{H},\lambda} + \Delta_z$ 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 $\mathbf{c}_\infty = ([\hat{\mathbf{H}}]_{*,\mathcal{Q}}\Phi^\top)^+ \mathbf{h}_{\mathcal{H},\lambda}$. Then with probability no less than $1-\delta$ we have:*

$$\left\|([\mathbf{H}]_{h,\mathcal{Q}}\Phi^\top)\mathbf{c}_\infty - [\mathbf{H}]_{h,\lambda}\right\|_{\rho(h)} \leq \epsilon(|\mathcal{H}|, |\mathcal{Q}|, d, \|\mathbf{m}_\infty\|^2, \|[\mathbf{H}]_{h,\mathcal{Q}}^\top\|^2_{\rho(\mathbf{h})}, 0, \sigma_\infty^2, \delta),$$

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

Proof Similar to Theorem 1, we have $[\mathbf{H}]_{h,\lambda} = [\mathbf{H}]_{h,\mathcal{Q}}\mathbf{m}_\infty$ for all h. Therefore we have a linear target and by definition $\mathbf{c}_\infty$ is the COLS estimate with projection Φ. We have:

$$[\hat{\mathbf{H}}]_{h,\lambda} = [\mathbf{H}]_{h,\lambda} + \Delta_z = [\mathbf{H}]_{h,\mathcal{Q}}\mathbf{m}_\infty + \Delta_z \tag{5.35}$$

$$= [\hat{\mathbf{H}}]_{h,\mathcal{Q}}\mathbf{m}_\infty - \Delta_x\mathbf{m}_\infty + \Delta_z. \tag{5.36}$$

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 5.11. ■

5.2.3 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 $\mathbf{c}_0\mathbf{C}_{a_1o_1}\mathbf{C}_{a_2o_2}\dots\mathbf{C}_{a_no_n}\mathbf{c}_\infty$.

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 $[\mathbf{H}]_{h,\mathcal{Q}} \sim \mu$. Note that as we increase the size of a history h, the norm of $[\mathbf{H}]_{h,\mathcal{Q}}$ 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 $[\mathbf{H}]_{[ao]_{1:t},\mathcal{Q}}$ is sampled from a scaled version of μ (i.e., $\frac{1}{s_t}[\mathbf{H}]_{[ao]_{1:t},\mathcal{Q}} \sim \mu$). Therefore $\|f([\mathbf{H}]_{h,\mathcal{Q}}^\top)\|_{\rho_t(h)} = \|f(s_t[\mathbf{H}]_{h,\mathcal{Q}}^\top)\|_{\rho(h)}$.

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 $\|f([\mathbf{H}]_{h,\mathcal{Q}})\|_{\rho_t(h)} = \|f(s_t[\mathbf{H}]_{h,\mathcal{Q}})\|_{\rho(h)}$ for all measurable f. With probability $1-\delta$:*

$$\left\|\mathbf{c}_0^\top \mathbf{C}_{a_1o_1}\mathbf{C}_{a_2o_2}\dots\mathbf{C}_{a_no_n}\mathbf{c}_\infty - \mathbb{P}(o_{1:n}|a_{1:n})\right\|_{\rho_n} \;\leq\; \epsilon_\infty s_n + \|\mathbf{c}_\infty\|\epsilon\sum_{t=1}^{n-1} s_t c^{n-t-1}, \quad (5.37)$$

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

Proof For all t, define $\mathbf{e}_t = (\mathbf{c}_0\mathbf{C}_{a_1o_1}\mathbf{C}_{a_2o_2}\dots\mathbf{C}_{a_no_n} - [\mathbf{H}]_{[ao]_{1:t},\mathcal{Q}})^\top$. After applying the nth compressed operator we have:

$$\begin{aligned}
\|\mathbf{e}_n\|_{\rho_n} &= \|(\mathbf{c}_0\mathbf{C}_{a_1o_1}\mathbf{C}_{a_2o_2}\dots\mathbf{C}_{a_no_n} - [\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}})^\top\|_{\rho_n} && (5.38)\\
&= \|\mathbf{C}_{a_no_n}^\top([\mathbf{H}]_{[ao]_{1:n-1},\mathcal{Q}} + \mathbf{e}_{n-1}^\top)^\top - [\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}^\top\|_{\rho_n} && (5.39)\\
&\leq \|\mathbf{C}_{a_no_n}^\top\mathbf{e}_{n-1}\|_{\rho_n} + \|\mathbf{C}_{a_no_n}^\top[\mathbf{H}]_{[ao]_{1:n-1},\mathcal{Q}}^\top - [\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}^\top\|_{\rho_n} && (5.40)\\
&\leq \|\mathbf{C}_{a_no_n}^\top\mathbf{e}_{n-1}\|_{\rho_n} + \max_{o_n,a_n}\|\mathbf{C}_{a_no_n}^\top[\mathbf{H}]_{[ao]_{1:n-1},\mathcal{Q}}^\top - [\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}^\top\|_{\rho_{n-1}} && (5.41)\\
&\leq c\|\mathbf{e}_{n-1}\|_{\rho_n} + \max_{o_n,a_n}\|s_{n-1}\mathbf{C}_{a_no_n}^\top[\mathbf{H}]_{[ao]_{1:n-1},\mathcal{Q}}^\top - s_{n-1}[\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}^\top\|_{\rho} && (5.42)\\
&\leq c\|\mathbf{e}_{n-1}\|_{\rho_{n-1}} + s_{n-1}\epsilon && (5.43)\\
&\leq \epsilon\sum_{t=1}^{n-1} s_t c^{n-t-1}. && (5.44)
\end{aligned}$$

Line 5.42 uses the distribution assumption on ρ_{n-1} and having $[\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}$ linear in

$[\mathbf{H}]_{[ao]_{1:n-1},\mathcal{Q}}$. Line 5.44 follows by induction. We now apply the normalizer operator:

$$\|\mathbf{c}_0^\top \mathbf{C}_{a_1o_1}\mathbf{C}_{a_2o_2}\dots\mathbf{C}_{a_no_n}\mathbf{c}_\infty - \mathbb{P}(o_{1:n}|a_{1:n})\|_{\rho_n} \tag{5.45}$$

$$= \|([\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}} + \mathbf{e}_n)^\top \mathbf{c}_\infty - \mathbb{P}(o_{1:n}|a_{1:n})\|_{\rho_n} \tag{5.46}$$

$$\leq \|\mathbf{c}_\infty \mathbf{e}_n^\top\|_{\rho_n} + \|[\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}^\top \mathbf{c}_\infty - \mathbb{P}(o_{1:n}|a_{1:n})\|_{\rho_n} \tag{5.47}$$

$$\leq \|\mathbf{c}_\infty\|\|\mathbf{e}_n\|_{\rho_n} + \|s_n[\mathbf{H}]_{[ao]_{1:n},\mathcal{Q}}^\top \mathbf{c}_\infty - s_n\mathbb{P}(o_{1:n}|a_{1:n})\|_{\rho} \tag{5.48}$$

$$\leq \|\mathbf{c}_\infty\|\epsilon\sum_{t=1}^{n-1} s_t c^{n-t-1} + \epsilon_\infty s_n. \tag{5.49}$$

Line 5.48 uses the distribution assumption on ρ_n and Line 5.49 uses the bound of Theorem 3. ■

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.

5.3 Effects of Compressing Histories

The above analysis assumes that $\Phi_\mathcal{H}$ has orthonormal columns. However, in order to obtain maximal computational benefits, it is beneficial to use a compressive $\Phi_\mathcal{H}$, i.e. a $\Phi_\mathcal{H}$ that acts as a feature selector on histories.

As with the previous section on the effects of compressing tests, we view the compression of histories from the perspective of linear regression. In this context, 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

$$\mathbf{y} = \mathbf{X}^\top\mathbf{w} + \boldsymbol{\eta} \rightarrow \Phi_\mathcal{H}\mathbf{y} = \Phi_\mathcal{H}\mathbf{X}^\top\mathbf{w} + \Phi_\mathcal{H}\boldsymbol{\eta}. \tag{5.50}$$

Intuitively, we can view this projection by $\Phi_\mathcal{H}$ 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 $\Phi_{\mathcal{H}}$ 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 analysed in detail by [77] (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 $\mathbf{Q}^\top\Phi^\top\Phi\mathbf{Q}$ decreases asymptotically to zero almost surely, where $\mathbf{Q} \in \mathbb{R}^{n\times m}$ and $\Phi \in \mathbb{R}^{d\times n}$ 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.

For completeness, however, we provide a brief sketch of main idea underlying this claim, using CPSR specific notation for clarity. We first recall that $\Phi_{\mathcal{H}}$ has no effect on the regression in the case where $\Phi_{\mathcal{H}}^\top\Phi_{\mathcal{H}} = \mathbf{I}$, i.e when it has orthonormal columns. Though a compressive $\Phi_{\mathcal{H}} \in \mathbb{R}^{d_{\mathcal{H}}\times|\mathcal{H}|}$ with $d_{\mathcal{H}} < n$ may never have orthonormal columns, we outline how existing results demonstrate that a random compressive $\Phi_{\mathcal{H}}$ *acts* as if it where orthonormal with high probability. In particular, we use the following:

Theorem 5 ***(Adapted from [59])*** *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathcal{H}}$ *with* $||\mathbf{x}||, ||\mathbf{y}|| \leq 1$. *Assume that* $\Phi_{\mathcal{H}} \in \mathbb{R}^{d_{\mathcal{H}}} \times |\mathcal{H}|$ *is a random matrix with either (1) independent* $\mathcal{N}(0, 1/d_{\mathcal{H}})$ *entries, or (2) independent normalized Rademacher entries (i.e.,* $\pm 1/\sqrt{d_{\mathcal{H}}}$*). Then for all* $\gamma > 0$*, there exists constants* $1 \leq C_1, C_2 \leq 7.8$ *such that*

$$\mathbb{P}(|\langle\Phi_{\mathcal{H}}\mathbf{x}, \Phi_{\mathcal{H}}\mathbf{y}\rangle - \langle\mathbf{x}, \mathbf{y}\rangle| \geq \gamma) \leq 2\exp\left(-d_{\mathcal{H}}\frac{\gamma^2}{C_1 + C_2\gamma}\right). \tag{5.51}$$

This theorem says that with high probability (that grows with the compressed dimension size)

$$\langle\Phi_{\mathcal{H}}\mathbf{x}, \Phi_{\mathcal{H}}\mathbf{y}\rangle \approx \langle\mathbf{x}, \mathbf{y}\rangle, \tag{5.52}$$

which immediately implies that (under similar conditions)

$$\mathbf{H}^\top\Phi_{\mathcal{H}}^\top\Phi_{\mathcal{H}}\mathbf{H} \approx \mathbf{H}^\top\mathbf{H}, \tag{5.53}$$

since $[\mathbf{H}^\top\mathbf{H}]_{i,j} = \langle[\mathbf{H}]_{*,i}, [\mathbf{H}]_{*,j}\rangle$ and $[\mathbf{H}^\top\Phi_\mathcal{H}^\top\Phi_\mathcal{H}\mathbf{H}]_{i,j} = \langle\Phi_\mathcal{H}[\mathbf{H}]_{*,i}, \Phi_\mathcal{H}[\mathbf{H}]_{*,j}\rangle$. Equation (5.53) more formally captures the notion of $\Phi_\mathcal{H}$ acting as if it has orthonormal columns. And this sketches the proof of the pertinent aspects of Claim 4.3 in [77]. In particular, the proof proceeds by conditioning upon a well behaved $\Phi_\mathcal{H}$ (in the sense of Theorem 5). The formal integration of the results in [77] with our results on compressing tests/features (i.e., replacing the l_1 regularization with feature compression) is the subject of future work

Finally, we note 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. Empirical investigations show that the compression of histories to $d_\mathcal{H} = d_\mathcal{T}$ introduces only a small amount of error during model learning.

Figure 5.1 shows an example of such an empirical investigation, demonstrating that the compression of histories has only a small empirical impact on the predictive quality of the learned models. 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 random Gaussian projection matrices and using (identical) samples generated from a grid-world domain.[2] As is evidenced in the plot, there is only a small difference in likelihood between the two models (c.f. the likelihood difference seen in section 7.3), and in fact, the model with compressed histories does slightly better for the first few time steps.

5.4 Discussion

This chapter presented theoretical results bounding (under some assumptions) the excess-risk of learning a PSR model within a compressed space. The focus of this analysis was on the compression of tests during learning, where we showed bounds on the error of learning a compressed model and also bounds on how this error propagates as an agent updates a compressed model during interactions with a system.

[2]See section 7.3 for details of the experimental set-up, as the settings used in this comparison are identical to those used in that section.

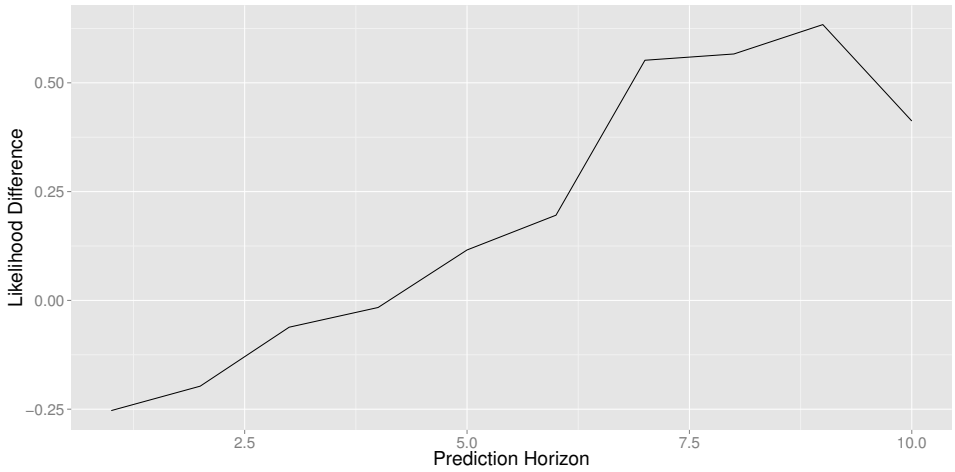


Figure 5.1: Difference in log-likelihood between model where histories are not compressed and where histories are compressed.

As a by-product of this analysis, we also provide insights into the general problem of compressed regression where both noisy features *and* noisy targets are compressed; in existing works [e.g. 46, 47, 28] it is assumed that only noisy features are compressed. A discussion of the effects of compressing histories was included as well.

The results of this chapter build upon existing works on compressed regression, which either analyse compression of features [46, 47, 28] or samples [77] independently. Following these works, we analyse the effects of compression on tests (which correspond to features) and histories (which correspond to samples) independently. The integration of these two frameworks is the subject of future work, and we expect that the regularization induced by the compression of tests will aid in bounding the error introduced by compressing histories.

6

Planning with CPSRs

The learning algorithm presented in chapter 4 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 [52] and learns a PSR variant of the Q-function. In essence, the approach substitutes a predictive state in place of an observable state in the standard fitted-Q learning algorithm of [27].

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 [36], whereas in general PSRs may have no corresponding finite-dimensional POMDP [21].[1] 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 exact PSR model is used) [27].[2] 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). This is preferable to

[1] 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 [16, 36].

[2] 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 [16] and chapter 7 of this thesis) suggest that the impact of such errors is small.

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.

6.1 Fitted-Q with CPSRs

The goal of the fitted-Q algorithm is to learn an approximation of the Q-function from which optimal decisions can be inferred. Formally, we seek to learn an approximation of the Q-function such that a policy, or mapping π from CPSR states to actions, can be defined (via an argmax over actions in Q-function).

Recall from section 3.1.2 that a Q-function specifies the quality of a belief-state and action pair. In the context of PSRs, we define the Q-function as (recalling that d' is the dimension of the learned CPSR model):

$$Q \; : \; \mathcal{A} \times \mathbb{R}^{d'} \rightarrow \mathbb{R}. \tag{6.1}$$

The difference between this Q-function and the one defined in section 3.1.2 is that CPSR states are used in place of belief states. We reiterate that these CPSR states do not necessarily lie on the probability simplex, and thus, the majority of POMDP-based approaches for approximating the Q-function are not applicable, as they exploit the fact that the belief states are proper probability vectors [49, 56].

Fortunately, the fitted-Q algorithm, which we employ, does not require that the input states lie on a probability simplex; it only requires that the states represent a sufficient statistic for the system [27]. PSRs satisfy this requirement, meaning that the convergence results for fitted-Q carry over to this setting [27]. The basic idea behind the method, which is formalized in Algorithm 1, is to use function-approximation to iteratively build more and more accurate approximations of the Q-function. Intuitively, during these iterations, the Q-function incorporates longer and longer planning horizons (i.e., more and more information about future rewards) and approaches the recursive definition in (3.2).

In this work, the *Extra-Trees* algorithm is used as the base regression algorithm [29], as it is a non-linear function approximator for which the fitted-Q convergence results hold [27]. 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 chapter 4):

$$O\left(\Psi(T) \times L|Z| \log\left(L|Z|\right)\right), \tag{6.2}$$

which is a factor $\Psi(T) \times \log(L|Z|)$ greater than the complexity for the model learning algorithm of chapter 4. In practice, we found Algorithm 1 to be several orders of magnitude slower than the CPSR learning algorithm.

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

6.2 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. For example, it naturally accommodates incrementally exploring a domain using a ϵ-greedy type strategy. In this setting π_s, the sampling policy, would be a policy which takes actions from π_i with probability $1-\epsilon$ and random actions otherwise (a similar ϵ-greedy strategy could also be defined via the soft-max function). The hope with this sort of strategy is that at each iteration the policies improve, allowing for exploration to focus more on important regions of the domain; the model learning would then be focused on these important regions, since the samples used for learning would come from these regions. The general specification also permits permits pure unbiased random sampling or even the drawing of samples from some arbitrary (e.g., expert) policy. Of course, if non-blind (i.e., non-random) policies are used then the sample distribution becomes biased (i.e., the samples are no longer i.i.d.), and the analysis of chapter 5 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 (4.14) and (6.2)).

6.3 Discussion

This chapter introduced a principled approach to sequential decision-making under uncertainty. Though the focus of this work is how this planning algorithm can be used with CPSR models, it is applicable to general PSR models as well. By adapting the fitted-Q approach of [27], which was developed for the fully observable setting, the planning algorithm exploits the full expressive capabilities of PSRs. This is in contrast to approaches that simply modify POMDP planning algorithms for the PSR setting, as those type of approaches generally assume that PSR states

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

(i.e., prediction vectors) must represent valid probabilities, which is not the case in general. Moreover, we situate this planning algorithm within a general framework which permits a variety of sampling strategies.

Our approach builds upon, and is closely related to, the planning framework proposed by [52]. The primary difference being that we explicitly motivate the use of fitted-Q instead of a point-based approach and that we situate the fitted-Q algorithm within a flexible framework that allows for diverse sampling strategies.

The primary limitation of this approach is its computational complexity, as the planning algorithm is significantly more expensive than model learning. It is possible that more specialized function approximation techniques (e.g., techniques which exploit CPSR-specific and/or domain-specific regularities) could provide performance increases, and the examination of this possibility is an interesting open question for future work. Another interesting direction for future work is exploring how incremental/online CPSR learning can be combined with online planning algorithms.

7

Empirical Evaluation of CPSR

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 (*i*) the empirical cost (in terms of prediction accuracy) of performing compression (if any), (*ii*) the compute-time reduction engendered by the use of compression, and (*iii*) the impact of the implicit regularization induced by performing compression. In the planning setting, we again seek to elucidate the empirical impact of performing compression. Specifically, we use synthetic robot navigation domains 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.[1] We also use compression to facilitate agnostic planning in a massive partially observable domain that is intractable for classic POMDP-based approaches (even when the underlying model is known), and we provide a qualitative comparison to the Monte-Carlo AIXI algorithm [70], a related approach for sequential decision-making under uncertainty.

7.1 Projection Matrices

In this analysis, we examine three different classes of random projection matrices: spherical, Rademacher, and hashed. The spherical projection matrices contain ran-

[1] Model-free approaches provide a fair baseline, as they learn without using a predictive model of the domain. By contrast, standard POMDP baseline approaches [e.g., 40, 56] all assume that probabilistic models of the domain are known a priori, meaning that they are not applicable to the problems examined here (where both model learning and planning must be performed).

dom Gaussian distributed entries. The Rademacher are a related class of random matrices where each entry is an independent Rademacher variable; these matrices also satisfy the JL lemma [11] 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) [1]. 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 [72, 65].

7.2 Domains

The domains used are based upon previous work on planning with PSRs and on sequential decision-making in large, complex partially observable domains.

7.2.1 ColoredGridWorld

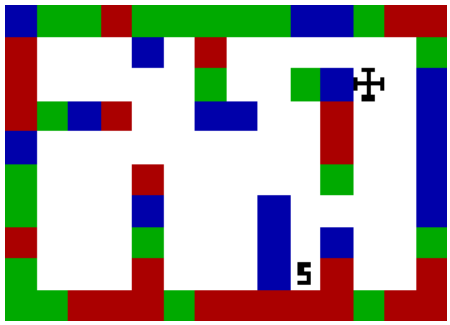


Figure 7.1: Graphical depiction of *ColoredGridWorld*. The **S** denotes the start position and the target denotes the goal.

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 [32] and [52]. 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.

7.2.2 Partially Observable PacMan

The second domain used is based upon the partially observable PacMan domain, denoted *PocMan*, first proposed by [66]. It is an extremely large partially observable domain with on the order of 10^{56} states. The basic dynamics follow that of the videogame PacMan: an agent must navigate a maze-like environment, 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 [70] in their work on a Monte Carlo AIXI approximation. In the second version, which we call *S-PocMan*, we further complicate the environment by dropping the parts of 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

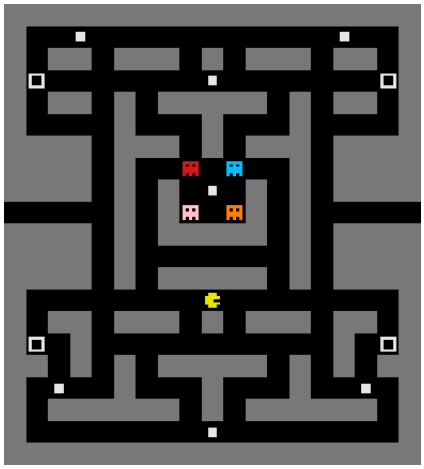


Figure 7.2: Graphical depiction of *S-PocMan*. The white dots denote food and the white annuli denote power-pills.

reason for examining this more difficult version of the domain is that, as summarized in section 7.4, 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.

7.3 CPSR Model Learning 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 chapter 4) set after performing SVD; however, singular values below an tolerance of 10^{-6} were also discarded. All tests, τ_i, with

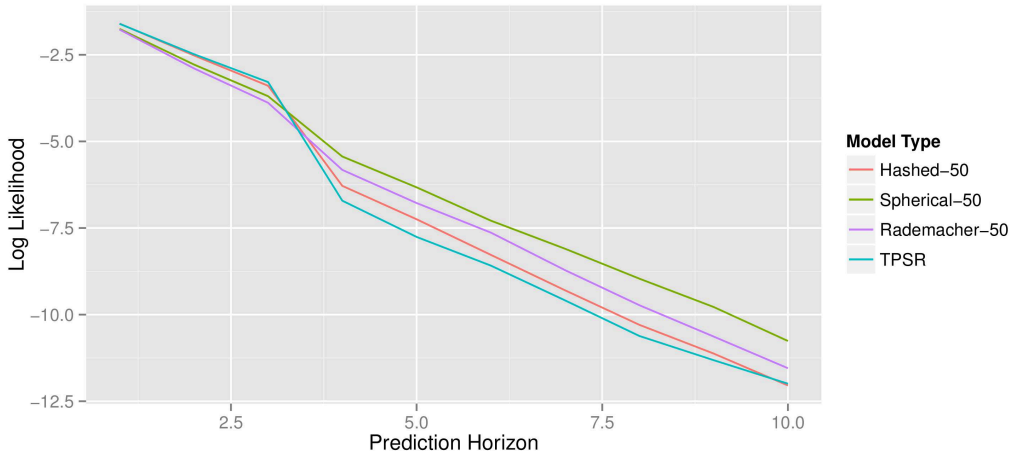


Figure 7.3: Predictive model learning results on the *ColoredGridWorld* domain. Plot shows the log-likelihood of the test data given the different models as the prediction horizon is increased. The numbers adjacent to the CPSR projection types correspond to the compressed dimension, d, used. 95% confidence interval error bars are too small to be visible.

$|\tau_i| \leq 7$ were included in the estimation process[2] (including longer length tests did not improve performance). For the CPSR models, projection dimensions of 25, 50, and 75 were examined and only the best performing size (determined via validation trials) is reported. The same size projections were used for both tests and histories. All models used 10000 train trajectories (of max length 13) and were evaluated with 100000 trajectories. The PacMan-style domains were not examined in this model-quality context as naive TPSRs exhausted memory limits when tests of length longer than 1 were used, making a rigorous comparison is infeasible[3].

Figure 7.3 plots the log-likelihood of the models as the prediction horizon (i.e., number of steps ahead to predict) is increased. From this figure, we see that the compressed models are not only competitive with the uncompressed TPSR, they

[2]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.

[3]Experiments were run on a machine with an 8-core 3.2 GHz Intel Xeon processor (x64 architecture) and 8Gb of RAM.

actually outperform the TPSR at longer prediction horizons. We conjecture that this is due to the regularization induced by the use of random projections. Figure 7.4 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.

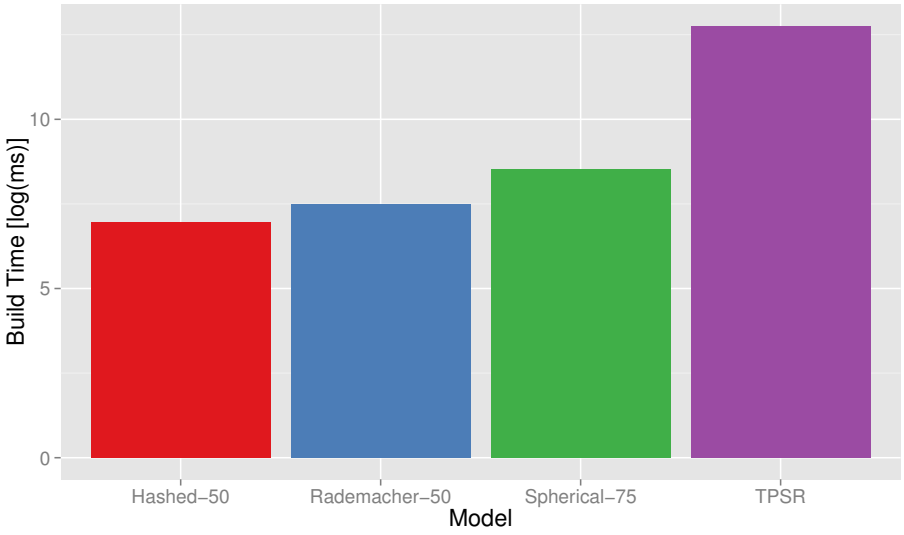


Figure 7.4: Model build times (on a log-scale) for the different model types on the *ColoredGridWorld* domain. Compressed dimension sizes are listed next to the model names. Times do not include time taken to build the training set. 95% confidence interval error bars are too small to be visible.

7.4 CPSR Planning Results

Next, we apply the full learning and planning approach (Algorithm 2) to the domains *ColoredGridWorld, PocMan*, and *S-PocMan*. For *ColoredGridWorld*, the models examined were identical to those described in the model quality experiments above. For planning, we used $I_p = 1000$ with $N = 1$ and a random sampling strategy; this represents the standard unbiased batch-learning setting (section 7.5.1.4 discusses the possibility of using more complex sampling strategies). A discount factor of $\gamma = 0.99$ was used for this domain.

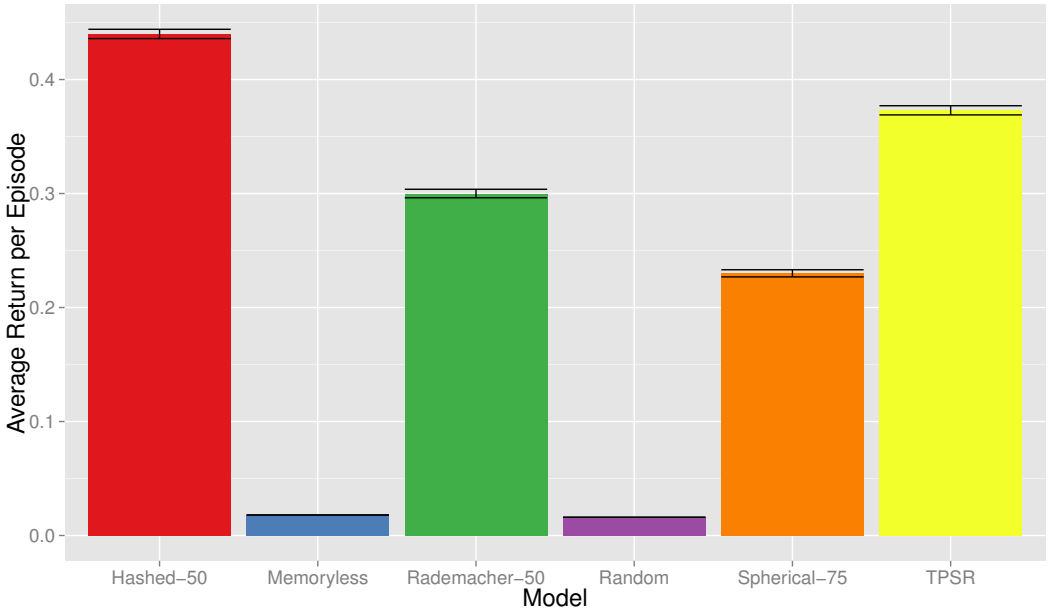


Figure 7.5: Average return (i.e., sum of discounted rewards) per episode achieved in the *ColoredGridWorld* domain using different models and the baselines. Compressed dimension sizes are listed next to the model names. 95% confidence interval error bars are shown.

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. For these domains, we again used 10000 sample trajectories to build the model, and tests of up to length 20 were included. For both domains, we set $I = 1000$, $N = 1$, and $\gamma = 0.99999$.

In all experiments, we used 100 fitted-Q iterations, one *Extra-Tree* ensemble of 25 trees per action, and the default settings for the *Extra-Trees* [29]. 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 [27]. 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.

Figure 7.5 details the performance of the different algorithms on the *Colored-GridWorld* 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.

Figure 7.6 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 [70] 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 -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 8.1), and so it should be possible to boost the performance of the CPSR models to

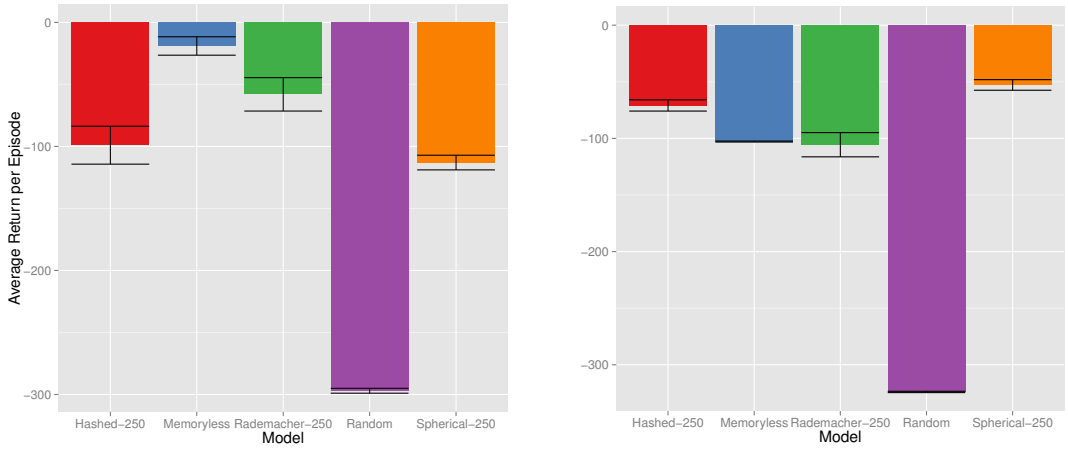
match that of the memoryless controller in that way.

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 sequential decision-making 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.

7.5 Discussion

The empirical results presented in this chapter demonstrate the efficacy of CPSR compared to uncompressed PSR models in terms of predictive capacity and usefulness in a sequential decision-making framework. The model learning 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 sequential decision-making under uncertainty. The domains used are inspired by previous work on learning and planning with PSRs in discrete domains [e.g. 75, 52]. The primary difference between these works and the results presented here is that efficiency of CPSR allows for experimentation in large domains that are infeasible for uncompressed PSR-based approaches.



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

Figure 7.6: Average return (i.e., sum of discounted rewards) 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.

This empirical analysis also bears similarity to the work of [16] and [13], where feature-extraction is used to learn PSRs in domains with continuous observation spaces. An interesting open question is how feature extraction can be combined with compression in an optimal manner, such that efficient learning and planning can be performed in domains with continuous observation spaces.

7.5.1 Practical Concerns

The implementation of complex sequential decision-making 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.

7.5.1.1 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. 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.

Our approach to this issue involved evaluating different projection dimensions (and projection types) via model quality experiments (which are not as computationally expensive as planning experiments). For simplicity, we set $d_{\mathcal{T}} = d_{\mathcal{H}}$, as we experimentation did not reveal any significant benefits of using different projections for histories and tests. In general, techniques such as cross-validated grid-search (starting with exponentially separated values) are recommended. Since compressed models are computationally inexpensive to construct, this approach should suffice for the majority of applications.

7.5.1.2 Improving Efficiency by Caching

In chapter 4 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.

7.5.1.3 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 4.2, the singular values of $\hat{\mathbf{\Theta}}$ in (4.9) 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 (4.11) and (4.13). This stability issue can be alleviated by normalizing the probability estimates, or more generally, by scaling $\hat{\mathbf{\Theta}}$ 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 $\hat{\mathbf{\Theta}}$ 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.

7.5.1.4 Q-function Approximation and Sampling Strategies

Algorithm 2 in chapter 6 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 (i) our simulators were designed for the batch setting and (ii) the theoretical results of chapter 5 assume a blind (random) sampling strategy is used.

We did, however, experiment with a goal-directed sampling approach [52]. 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). [52] 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 $\hat{\mathbf{\Theta}}$ matrix in that certain entries dominated in terms of magnitude. As a result of this imbalance, the SVD in (4.9) 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 focusses on areas of the state-space relevant to planning [as shown by 52].

8

Conclusion

The CPSR approach provides a new avenue for efficient sequence prediction and sequential decision-making. By combing the method of moments with techniques from compressed sensing, the compressed learning algorithm allows accurate PSR models to be constructed in a memory and time efficient manner while also providing regularization. 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 method of moments 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.

Moreover, in deriving this algorithm, we made explicit the subtle connections between PSR models in sequential decision-making and the more general method of moments framework for sequence prediction. The CPSR algorithm is thus not only applicable to sequential decision-making; the algorithm is readily adaptable to more generic sequence prediction tasks.

8.1 Related Work

This work is very related to, and builds upon, a wealth of literature on moment-methods, PSR learning and PSR-based planning approaches. The TPSR approach

[16, 61] to learning PSRs and the spectral Hankel factorization technique for learning WAs serve as foundations for the CPSR learning algorithm. The primary difference between CPSRs and these models, is that CPSR models are no longer linear transforms of true WA/PSR models; they are, in fact, compressed projections. In other words, CPSR models may be viewed as combining the spectral learning method with an efficient, principled, and domain-independent feature selection strategy (where the random projections are viewed as feature selectors). Of course, this view is somewhat limited, as the CPSR paradigm, in fact, fundamentally alters the learning objective: instead of finding a concise linear transform of a WA/PSR, the CPSR objective is to find an accurate approximation of a PSR within some compressed space (such that the compressed model cannot be linearly transformed back to the original).

In a similar vein, the goal-directed planning and learning approach of [52] serves as a foundation for our planning algorithm. 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 [39], which uses a type of hybrid PSR-MDP model to reduce computational costs and increase predictive accuracy, and the hierarchical PSRs (HPSRs) of [75], which uses the option framework [69] 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 sequential decision-making algorithms, which use adaptive history-based techniques. Examples of these algorithms include *U-Tree* [48] and the *Monte-Carlo AIXI approximation* [70]. These approaches share the motivation of developing agnostic agents that can learn and plan without domain knowledge. They differ, however, in the instantiation of their model-based approach, as they use an adaptive history-based approach, which uses variable size windows over histories. A key aspect of these approaches is focussing the model-learning on areas of the state-space relevant to achieving goals (similar to the goal-directed sampling routine) [48]. Thus, a fundamental difference between

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. For example, in a robot navigation problem, there may be a common environment in which different navigation tasks must be carried out. CPSR would benefit in this case since a single model of the domain could be learned and reused for the different navigation tasks.

8.2 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 the latent variable modelling and the method of moments while AIXI-like methods have information-theoretic (and/or Bayesian) motivations [70]. Recently, there have been a number of theoretical advancements in the understanding of moment-based latent variable models, such as the local loss formulation of [10]. 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, AIXI-like methods often explicitly penalize 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 compression via random projections in more diverse moment-methods. For example, the method of moments is used in natural language processing applications to learn probabilistic context free grammars (PCFGs) [18], and these more specialized applications could benefit from the use of compressed learning approaches.

Certain questions related to the theoretical analysis of compression also remain open. For example, a formal integration of results on compressing histories and features simultaneously is an important direction for future work. Moreover, an interesting open question is whether or not it is possible to develop a statistical test for the quality of compressive features within the PSR setting that does not rely on building a complete model with these features and measuring prediction accuracy. It is possible that certain spectral properties of a (compressed) Hankel matrix may provide insight into the quality of compressive features, but the formal investigation of this remains the subject of future work.

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. 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 (4.13) 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. However, these engineering issues should not necessitate altering the core of the CPSR approach.

Bibliography

[1] D. Achlioptas. Database-friendly random projections. In *Proceedings of the 20th Association for Computing Machinery Symposium on Principles of Database Systems*, 2001.

[2] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*, 2012.

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

[4] R. Bailly, X. Carreras, F. Luque, and A. Quattoni. Unsupervised spectral learning of WCFG as low-rank matrix completion. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2013.

[5] R. Bailly, F. Denis, and L. Ralaivola. Grammatical inference as a principal component analysis problem. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[6] R. Bailly, A. Habrard, and F. Denis. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of the 21st Conference on Algorithmic Learning Theory*, 2010.

[7] B. Balle, X. Carreras, F.M. Luque, and A. Quattoni. Spectral learning of weighted automata: A forward-backward perspective. *Machine Learning*, pages 1–31, 2013.

[8] B. Balle, W. L. Hamilton, and J. Pineau. Methods of moments for stochastic languages: Unified presentation and empirical comparison. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[9] B. Balle and M. Mohri. Spectral learning of general weighted automata via constrained matrix completion. In *Advances in Neural Information Processing Systems*, 2012.

[10] 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.

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

[12] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence*, 22:385–421, 2004.

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

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

[15] 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.

[16] B. Boots, S. Siddiqi, and G. Gordon. Closing the learning-planning loop with predictive state representations. In *Proceedings of Robotics: Science and Systems VI*, 2009.

[17] 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.

[18] S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Spectral learning of latent-variable PCFGs. *Assocation for Computational Linguistics*, pages 223–231, 2012.

[19] S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL-HLT*, 2013.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[21] F. Denis and Y. Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 86(1):41–77, 2008.

[22] M. Deshpande and G. Karypis. Selective Markov models for predicting web page accesses. *ACM Transactions on Internet Technology*, 4(2):163–184, 2004.

[23] P. S. Dhillon, J. Rodu, M. Collins, D. P. Foster, and L. H. Ungar. Spectral dependency parsing with latent variables. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

[24] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[25] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the L_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[26] P. Dupont, F. Denis, and Y. Esposito. Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349–1371, 2005.

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

[28] 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.

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

[30] G. Gordon. *Approximate solutions to Markov decision processes*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1999.

[31] W. L. Hamilton, M. M. Fard, and J. Pineau. Efficient learning and planning with compressed predictive states. arXiv preprint arXiv:1312.0286v1, 2013.

[32] 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.

[33] D. Hsu and S. M. Kakade. Learning mixtures of spherical Gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, 2013.

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

[35] M. Hulden. Treba: Efficient numerically stable EM for PFA. In *Proceedings of the 2012 International Conference on Grammatical Inference*, 2012.

[36] 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.

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

[38] 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.

[39] 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.

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

[41] T. Koski. *Hidden Markov models for bioinformatics*, volume 2. Springer, 2001.

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

[43] M. Littman. *Algorithms for sequential decision making*. PhD thesis, Brown University, 1996.

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

[45] F.M. Luque, A. Quattoni, B. Balle, and X. Carreras. Spectral learning in non-deterministic dependency parsing. *European Association for Computational Linguistics*, 2012.

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

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

[48] A McCallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, The University of Rochester, 1996.

[49] K. Murphy. A survey of POMDP solution techniques. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 2000.

[50] A. Ng and M. Jordan. Pegasus: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[51] 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*. AAAI Press, 2013.

[52] 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.

[53] S. C. W. Ong, Y. Grinberg, and J. Pineau. Mixed observability predictive state representations. In *Association for the Advancement of Artificial Intelligence*, 2013.

[54] J. R. Partington. *An introduction to Hankel operators*, volume 13. Cambridge University Press, 1988.

[55] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London*, pages 71–110, 1894.

[56] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2003.

[57] M. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[58] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In A. Waibel and K. Lee, editors, *Readings in speech recognition*, pages 267–296. 1990.

[59] H. Rauhut, K. Schnass, and P. Vandergheynst. Compressed sensing and redundant dictionaries. *IEEE Transactions on Information Theory*, 54(5):2210–2219, 2008.

[60] A. Recasens and A. Quattoni. Spectral learning of sequence taggers over continuous sequences. In *Proceedings of the 2013 European Conference on Machine Learning and Principles of Knowledge Discovery in Databases*, 2013.

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

[62] S. Ross, B. Chaib-draa, and J. Pineau. Bayes-adaptive POMDPs. In *Advances in Neural Information Processing Systems*, 2007.

[63] N. Roy, G. Gordon, and S. Thrun. Planning under uncertainty for reliable health care robotics. In *Field and Service Robotics*, 2006.

[64] S. Russell, P. Norvig, J. Canny, J. Malik, and D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice Hall Englewood Cliffs, 1995.

[65] 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.

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

[67] 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.

[68] R. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge University Press, 1998.

[69] 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.

[70] 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.

[71] S. Verwer, R. Eyraud, and C. Higuera. Results of the PAutomaC probabilistic automaton learning competition. In *Proceedings of the 10th International Conference on Grammatical Inference*, 2012.

[72] 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.

[73] E. Wiewiora. *Modeling probability distributions with predictive state representations.* PhD thesis, University of California at San Diego, 2007.

[74] 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.* ACM, 2005.

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

[76] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

[77] S. Zhou, J. Lafferty, and L. Wasserman. Compressed regression. In *Advancements in Neural Information Processing Systems*, 2007.