Bayesian Nonparametric Approaches for Reinforcement Learning in Partially Observable Domains

by

Finale Doshi-Velez

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .....
Department of Electrical Engineering and Computer Science
April 27, 2012

Certified by...
Nicholas Roy
Associate Professor
Thesis Supervisor

Accepted by .....
Leslie Kolodziejski
Chairman, Department Committee on Graduate Students

Bayesian Nonparametric Approaches for Reinforcement Learning in Partially Observable Domains

by

Finale Doshi-Velez

Submitted to the Department of Electrical Engineering and Computer Science
on April 27, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Making intelligent decisions from incomplete information is critical in many applications: for example, medical decisions must often be made based on a few vital signs, without full knowledge of a patient's condition, and speech-based interfaces must infer a user's needs from noisy microphone inputs. What makes these tasks hard is that we do not even have a natural representation with which to model the task; we must learn about the task's properties while simultaneously performing the task. Learning a representation for a task also involves a trade-off between modeling the data that we have seen previously and being able to make predictions about new data streams.

In this thesis, we explore one approach for learning representations of stochastic systems using Bayesian nonparametric statistics. Bayesian nonparametric methods allow the sophistication of a representation to scale gracefully with the complexity in the data. We show how the representations learned using Bayesian nonparametric methods result in better performance and interesting learned structure in three contexts related to reinforcement learning in partially-observable domains: learning partially observable Markov Decision processes, taking advantage of expert demonstrations, and learning complex hidden structures such as dynamic Bayesian networks. In each of these contexts, Bayesian nonparametric approach provide advantages in prediction quality and often computation time.

Thesis Supervisor: Nicholas Roy
Title: Associate Professor

Dedication

To seekers of truth in world of hidden variables.

Acknowledgments

As an undergraduate in 16.410, Principles of Autonomy and Decision-Making, I was present when Nick gave a guest lecture on particle filters—one of his first lectures as a new professor at MIT. I already thought that AI was cool and probability was awesome, but this? AI *and* probability? I was hooked. Over the years, I'm very grateful to have had an advisor who not only collaborated on my research but helped me discover what research was, with whom I could both agree and disagree, and who encouraged me to follow my passions for even when they took me abroad to study Bayesian nonparametric statistics at the University of Cambridge. I hope that I can emulate his dedication to both his research and his lab in my future career.

I also thank my committee: Zoubin, Leslie, Tomas, and Josh not only personally provided insightful suggestions for my research, but by welcoming me into each of their groups, they provided opportunities for me to understand these multifaceted problems in a way that would not have been possible otherwise. I am also grateful to all the wonderful people in those groups—RRG, MLG, LIS, CoCoSci—who joined me in innovative research discussions and collaborations and perhaps even more innovative ways to not take work too seriously.

And last but not least: these years would have been extremely dreary without my wonderful family and friends. I owe much to my walking buddies, pub and coffeehouse discussion friends, pathshala peoples, artists and martial artists, partners in silliness and stories—the many, many wonderful people who made me look forward to each day. I owe much to my family, especially my parents, who never failed to ring no matter how often I forgot to call, and my brother, who never failed to provide both distractions and Costco runs... and finally, I owe much to my husband Javi, my anchor, who never failed to show me where the silver lining was, when I couldn't find it myself.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine exploring a new city for the first time. Busy intersections, quiet plazas, beckoning cafes—there are countless places to see and countless connections between them. Trying to memorize everything from the start is a daunting task, especially when it is unclear what "everything" means: Do we mean every location? Every connecting road? Is it worth remembering where each piece of litter was? Which corners have which street performers? Some kinds of knowledge, such as traffic patterns, cannot be learned immediately: multiple visits to the same intersection are needed to separate pattern from coincidence. When faced with such a challenge, most people will build their model of the city gradually: we may start off by memorizing the few key intersections encountered on our daily commute, and, after some time, we may start to pay attention to finer details such as alternate routes or weekly variations in traffic. Most people will also focus on models that make good predictions: a model that recalls where a particular piece of litter was spotted is less useful for future excursions than a model that learns how late the buses tend to run.

Many sequential decision-making problems exhibit a similar structure in which predictive information is gradually learned from data. Medical decisions must be made based on a limited number of tests and a limited knowledge of the patient's physiology. Still, doctors can often do quite well by starting with a few key principles and refining their knowledge about the patient will react over time. Recommender systems must suggest items that users may wish to buy from a limited purchasing

history. Still, these systems can also do well by starting with notions of items that are generally popular and refining their knowledge about user's preferences over time. Game-playing agents must adjust the difficulty of a player's experience based on a limited number of games. As with the previous examples, these agents can still do well by trying some basic strategies first and refining their knowledge about the specific player's strategy over time.

All of the problems above are examples of reinforcement learning in partially observable domains: reinforcement learning studies how agents can learn to accomplish tasks by collecting experience, and partially observable domains are those in which the agent's entire history of interactions with the environment—whether it is an entire clinical record or purchasing history—may be relevant to making future decisions. Reinforcement learning in partially-observable settings is particularly challenging because the history is such a high-dimensional object, and most reinforcement learning approaches rely on different methods to compact histories into lower dimensional knowledge representation. However, the choice of what knowledge representation to use is not always obvious: for example, can all the relevant information in a patient's clinical record be summarized by a set of conditions? How their organs are functioning at a molecular level? Their genetic sequence? There are often many ways which we can choose to model the data, and each approach will have different gaps and uncertainties.

The work in this thesis addresses this fundamental question in partially observable reinforcement learning. Specifically, we demonstrate how Bayesian nonparametric statistics allows agents to incrementally learn about aspects of their environments that have predictive value. The Bayesian aspect of these methods provide ways for the agent to explicitly track its uncertainty and thus focus its knowledge-refinement process toward aspects that are less certain. Bayesian nonparametric methods have the added advantage of increasing the size of the agent's model only as needed to explain the agent's observations. For example, variables corresponding to key conditions that affect patient outcomes would be learned before inferring variants of these. Information about a patient's physiology at a molecular level may never be inferred if

simpler structures are sufficient for predicting clinical outcomes. Using Bayesian nonparametric statistics in the learning process results in both reduced computational requirements, as the agents start out with small models and expand them only as need, as well as reduced sample requirements, as agents are able to start performing well early-on by picking up the major trends in the data.

By automatically adjusting the sophistication of the model with the complexity of the data, Bayesian nonparametric methods provide an approach for learning a representation for a system that is the "right size" for the data. The learned models also have the capacity to incorporate new, unexpected events that may not be present in an initial training set. Of course, not all applications need such flexible or general methods to learn representations. For example, in many robotics and other engineering applications, the representation of the system is known by design, and sensors can be individually calibrated to fit necessary parameters. When designing controllers for dynamical systems, long sequences of data from the relevant operating regimes may be available to perform a more traditional system identification. Bayesian nonparametric methods are best-suited for problems where the choice of the knowledge representation is non-obvious—for example, how to characterize a patient's health or a player's strategy—and it is important to be able to adjust the size of the representation based on sparse data.

The core contributions of this thesis, summarized in section 1.2, are three models for applying Bayesian nonparametric methods to reinforcement learning in partially observable domains. The first, the infinite partially observable Markov decision process (iPOMDP), posits that the world consists of an infinite number of latent states, and instantiates states as they are needed to explain the agents' observations. The chapter on nonparametric policy priors extends the iPOMDP by considering how to combine expert input with the iPOMDP models learned from the agent's experience. Finally, the infinite dynamic Bayesian network (iDBN) extends the iPOMDP by considering situations where the latent state of the iPOMDP might be made up of multiple factors.

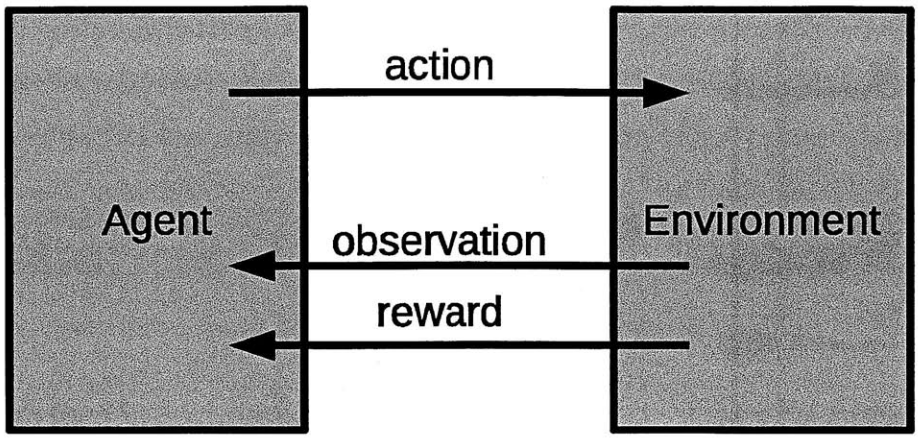


Figure 1-1: General reinforcement learning framework. At each time-step, the agent interacts with the environment with an action and receives an observation and reward.

1.1 Framework

We begin by formalizing the problem of reinforcement learning in partially observable domains. The most general version of the reinforcement learning problem [Sutton and Barto, 1998], summarized in figure 1-1, involves a sequence of exchanges between the agent (left) and the environment (right). At each time step, the agent interacts with the environment through an action a. The environment sends back an observation o and an immediate reward r. The agent's goal is to maximize the discounted sum of its expected rewards $E[\sum_t \gamma^t r_t]$, where r_t is the reward that the agent receives at time t, and $\gamma \in [0, 1)$ trades off between the importance of current rewards and future rewards.

As a simple example, consider a robot trying to navigate to a goal location. The actions a might correspond to a vector of motor commands, the observations o might be data from the robot's laser scanner, and the immediate reward r might indicate whether the robot has reached its destination. Even from this relatively simple scenario, we can see that every element of the agent's interaction *history* $h = \{a_0, o_0, r_0, ..., a_t, o_t, r_t\}$ may provide clues as to where the robot is and how it might try and reach the goal. For example, all corners of a room might produce the same observation o from the laser scanner, but with the entire history of actions and observations h, the agent might be able to disambiguate its position.

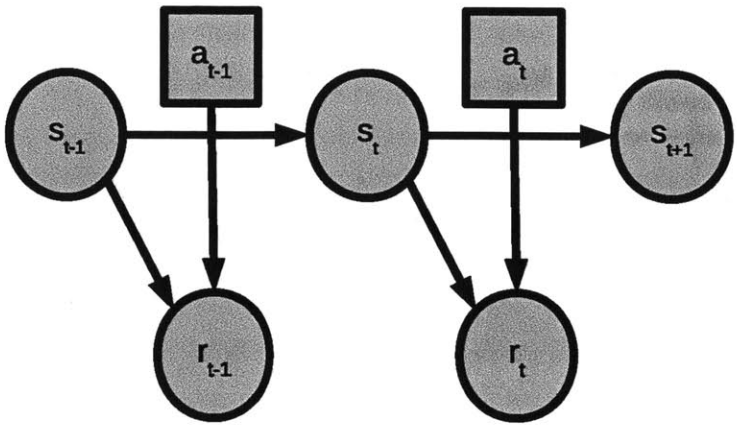


Figure 1-2: Graphical Model for the MDP. The shaded nodes s represent the state of the world, which are observed by the agent. The shaded squares a represent the agent's actions, and the shaded nodes r represent the agent's rewards.

Fully Observable Environments A special case of the reinforcement learning framework is one in which the observation o captures all of the information needed to make predictions from the history h. For example, suppose that the robot from the previous scenario had access to an oracle that always provided the robot with its current location. Given its current location, the robot would not require any additional information about its previous locations or previous actions to predict the effect of its next motor command. When the current observation o_t captures all of the information needed to predict the future, the environment is described as *fully observable.*

One very general way of modeling fully observable environments is with a Markov decision process (MDP, figure 1-2). In this formulation, the environment is modeled as consisting of a set of observed *states.* At each time step, the agent takes an action a which causes the environment to transition from its current state s to a new state s' based on a transition probability $T(s'|s,a)$. It also receives a reward $R(s,a)$. The values of the parameters of the transition distributions T and the reward function R are initially unknown. As in the general reinforcement learning framework, the agent's goal is still to maximize the sum of its discounted expected rewards $E[\sum_t \gamma^t r_t]$. While there are many ways to model environments, a key property of the MDP formulation is that the dynamics of the system are Markov in the state s—that is, given the

current state s_t at time t, we do not require any information about the previous states $s_0 ... s_{t-1}$ to predict the state s_{t+1} at time $t+1$.

Partially Observable Environments Environments in which the dynamics are not Markov in the current observation o_t are called *partially observable*. A very general way of describing partially observable environments is to assume that the environment is Markov with respect to some now *unobserved* state s which emits the noisy or partial observation o. For example, even with noisy sensors, the robot's dynamics are still Markov with respect to the robot's position—the only difference is that the robot's position is no longer directly observed. A partially observable Markov decision process (POMDP, figure 1-3) m is defined by the tuple $\{S, A, O, T, \Omega, R, \gamma\}$ [Sondik, 1971]. S, A, and O are sets of states, actions, and observations (all discrete for the purpose of this work). As in the MDP, the transition function $T(s'|s, a)$ gives the probability of transitioning to state s' after performing action a in state s, and the reward function $R(s, a)$ gives the reward for each state-action pair. The observation function $\Omega(o|s, a)$ gives the probability of seeing observation o after taking action a in state s.

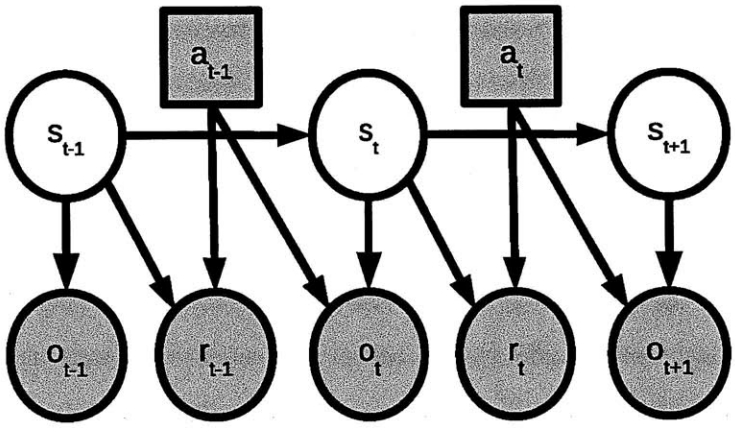


Figure 1-3: Graphical Model for the POMDP. The white nodes s represent the now *hidden* state of the world, and the shaded squares a represent the agent's actions. The shaded nodes o and r represent the observations and rewards.

Action Selection The rule that governs how the agent selects its actions is called a *policy* π. When the environment is fully-observable, the current state s_t encodes everything about the past needed to predict the future; thus, we know that the optimal policy lies in the set of functions $\pi(s,a) = p(a|s)$, which gives the probability of performing action a in state s. When the environment is partially-observable, all elements of the history $h_t = \{a_1, o_1, r_1, ..., a_t, o_t, r_t\}$ might be required to make predictions about the future. Thus, the optimal policy now lies in the set of functions $\pi(h,a) = p(a|h)$, which gives the probability of performing action a in state s.

Action-Selection with a Known Model: Planning Before describing how an agent might select its actions in the reinforcement learning setting, we consider a simpler setting in which the parameters of the POMDP model $m = \{S, A, O, T, \Omega, R, \gamma\}$ are given (known as *planning*). Let the *belief* $b_t(s)$ be the conditional distribution $p(s_t|h_t)$ over the current state s_t given the history h_t. If the parameters of the transition and observation functions T and Ω are known, it is possible to compute the current belief $b_t(s)$ given the previous belief $b_{t-1}(s)$, the current action a_t, and the current observation o_t:

$$b_t(s) = \Omega(o_t|s,a_t) \sum_{s' \in S} \frac{T(s|s',a_t) b_{t-1}(s')}{Pr(o_t|b_{t-1},a_t)}, \tag{1.1}$$

where $Pr(o|b,a) = \sum_{s' \in S} \Omega(o|s',a) \sum_{s \in S} T(s'|s,a) b(s)$. Unlike the most recent observation o_t, the belief $b_t(s)$ captures all the information in the history h_t needed to make predictions about the agent's future [Sondik, 1971]. In this sense, we can think of a partially observable environment as a fully observable environment in this high-dimensional, continuous space of beliefs. It follows that the optimal policy for the POMDP must then lie in the space of functions $\pi(b(s),a) = p(a|b(s))$, and a number of algorithms have been developed for finding near-optimal polices for this representation [Bellman, 1957, Littman et al., 1995, Pineau et al., 2003, Spaan and Vlassis, 2005, Smith and Simmons, 2004, Shani et al., 2007, Kurniawati et al., 2008].

Action-Selection with an Unknown Model: Learning Action-selection is much more challenging in the more general setting in which the parameters of the model m are not known (and thus update rules like equation 1.1 cannot be used to summarize

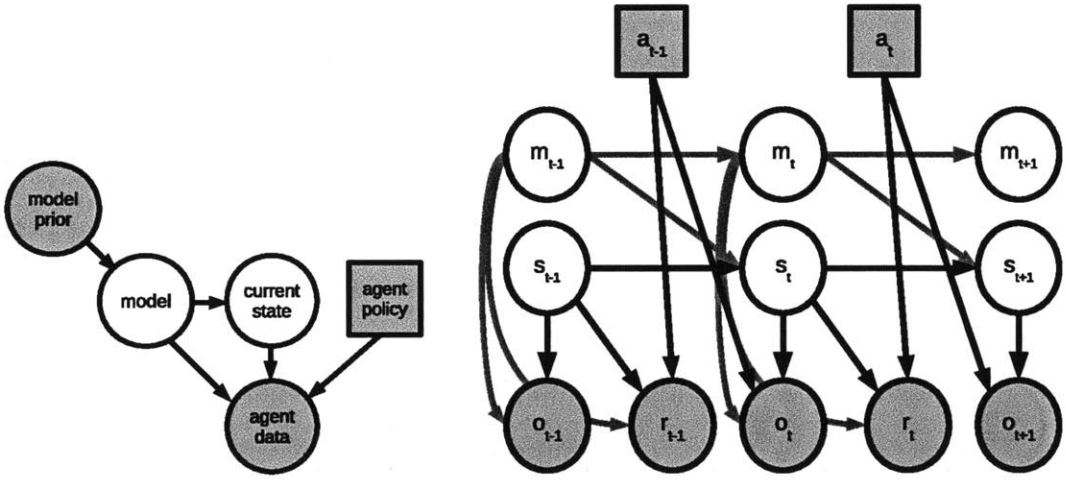
the history). Whether the environment is fully observable or partially observable, reinforcement learning—as opposed to planning—involves learning from data, and histories are inherently noisy. Thus, many repetitions of similar situations may be needed to characterize patterns. Second, it is difficult to assign credit for a reward: a high reward r_t at time t may be the result of the action a_t or some action farther back in the past. Finally, without knowing the model m, the agent must balance time spent learning the dynamics (exploration) and maximizing reward (exploitation).

These factors are particularly challenging in the partially observable setting where the optimal policy $\pi(a|h_t)$ is a function of the entire history h_t. Because the history h_t contains many more parameters than a single state s_t, the policy $\pi(a|h_t)$ will require many more samples to learn. Similarly, without more knowledge about the environment, we are forced to start learning a very general reward function $R(r|h)$ rather than having a simpler functional form $R(r|s)$. Finally, exploration is more difficult when we have to consider visiting histories h rather than states s. At the same time, these factors also make partially-observable reinforcement learning more general than settings in which the domain is fully-observed or fully-specified: we can operate directly in the space of histories and never think about hidden states or beliefs over hidden states.

Approaches to tackling the partially-observable reinforcement learning problem—that is, learning policies based on histories of experience—vary widely. On one end of the spectrum are methods that never directly model the hidden state and instead choose new actions from the history of past actions and observations. Most of these approaches try to group together histories that require similar actions—for example, U-Tree [McCallum, 1993] builds a suffix tree on the history, AIXI [Hutter, 2004] extracts features from the history, and predictive state representations [Singh and James, 2004] learn short-term predictions for a set of test histories. Under relatively mild conditions on the underlying POMDP, history-based approaches can learn near-optimal policies in polynomial time [Even-Dar et al., 2005]. However, even if the sample complexity is polynomial, these algorithms typically require large amounts of experience to convert histories into useful policies. For example, even newer AIXI

implementations, such as that of Veness et al. [2009], require on the order of 10^4 interactions to learn a very simple POMDP (tiger [Littman et al., 1995]), whereas a model-based Bayesian method such as BA-POMDP [Ross et al., 2008a] requires on the order of 10^2 interactions for the same problem.

On the other end of the spectrum are Bayesian methods which explicitly consider beliefs not only over potential hidden states but also over possible models [Poupart and Vlassis, 2008, Jaulmes et al., 2005, Ross et al., 2008a, Strens, 2000, Dearden et al., 1999, Ross et al., 2008b, Doshi et al., 2008, Duff, 2002]. These approaches note that reinforcement learning problem can be thought of as a planning problem in which both the agent's current state s_t and the world model m are hidden. Bayesian methods start out with a prior distribution over models $p(m)$ and then track the joint distribution $b_t(s, m) = p_t(s, m)$ over the current state and the world model m. As with a standard POMDP, the joint belief $b_t(s, m)$ is a sufficient statistic for the history: it encodes everything about the past needed to make future predictions. Thus, the optimal reinforcement learning policy lies in the set of functions $\pi(b(s, m), a)$ which requires solving a larger "model-uncertainty" POMDP (see graphical model in figure 1-4).



(a) Graphical model for the Bayesian RL framework, showing both the model and the current state as hidden nodes

(b) Expansion of the graphical model showing a single time-slice in the model-uncertainty POMDP; gray and black arrows are equivalent (colored only for clarity).

Figure 1-4: Graphical model of the Bayesian RL framework. The model m is considered a hidden variable along with the state s. Usually we assume that the world dynamics are stationary, that is $m_t = m_{t+1}$.

The Bayesian reinforcement learning setting also offers an elegant formulation for incorporating expert knowledge: the starting belief $b_0(s, m)$ can encode one's prior beliefs over what models m are likely. In general, Bayesian methods need fewer samples of experience (that is, fewer histories) to learn good policies. However, actually doing computations on the belief $b(s, m)$—such as applying equation 1.1—can be challenging because the model m itself consists of the many parameters contained in T,Ω,and R. Different Bayesian reinforcement learning methods present different approximations of standard POMDP planning techniques for this more complex space where we maintain distributions over both possible current states and possible models. The cost of the associated computations—having to reason about all the unknown parameters in model—have typically restricted these methods to small problems. Unlike model-free or history-based approaches, another difficulty with these approaches is that the structure of the hidden part of the underlying model, for example, the number of states or sets of factors, must now be specified.

Despite the variety of approaches and continued interest in partially observable reinforcement learning [Hutter et al., 2009], several factors have limited their success. First, much recent work—especially the Bayesian literature—has focused on inferring the dynamics of the true underlying system, rather than predicting how the system will respond to various inputs. While sometimes a reasonable goal, explicitly trying to infer the true system can make the learning problem unnecessarily challenging, especially when accurate predictions are all that are required for decision-making. In these cases, reasoning about extra model parameters is wasted computational effort. Current inference techniques also tend to converge to sub-optimal solutions. Finally, expert information is often incorporated in ways that impose rigid constraints on the model, rather than more targeted use of the agent's own experience.

Bayesian Nonparametric Model Learning This thesis examines how Bayesian nonparametric techniques can address many of the challenges we outlined for partially-observable reinforcement learning. A Bayesian nonparametric model defines a distribution over an infinite-dimensional parameter space [Orbanz and Teh, 2010]. For

example, in this thesis, we consider an extension of the POMDP in which the model m has an infinite number of hidden states s, and thus an infinite number of parameters are needed to describe the transitions $T(s'|s,a)$, observations $\Omega(o|s,a)$, and rewards $R(s,a)$. Like more traditional Bayesian approaches to reinforcement learning, we define an initial prior $p(m)$ over models m and update this prior belief as the agent gathers experience; as before, our use of Bayesian nonparametric methods for reinforcement learning involves casting the partially-observable reinforcement learning problem as solving the model-uncertainty POMDP in which the belief $b_t(s,m)$ is the proxy for the state. In doing so, we inherit many of the benefits of Bayesian approaches to reinforcement learning, including a clear optimization criterion and relatively low sample complexities for learning reasonable policies.

However, unlike more traditional Bayesian reinforcement learning approaches, which try to recover a model of the environment [Ross et al., 2008a, Jaulmes et al., 2005, Poupart and Vlassis, 2008, Dearden et al., 1999, Duff, 2002, MacKay, 1997], our emphasis is simply to be able to make good predictions. Drawing on concepts from Stolcke and Omohundro [1993], Shalizi and Shalizi [2004], and Drescher [1991], using Bayesian nonparametric methods allow us to think of hidden states not as physical aspects of the environment, but as "way-points" that are needed to make the underlying system Markovian. Using an approach that assumes that the world contains an infinite number of underlying states ensures that we will always have enough way-points to explain our observations. However, the prior $p(m)$ ensures that states are instantiated only if the current set of instantiated states do not explain the data well. Models instantiated in this incremental fashion often have fewer instantiated parameters than the "true" model, making them easier to learn and solve. Finally, the same Bayesian nonparametric methods used to keep distributions over models—that is, how the world works—can be used to keep distributions over well-performing policies $\pi(h,a)$—that is, how the agents should behave.

In the context of modeling dynamical systems, one of the core Bayesian nonparametric models used in this work is the infinite Hidden Markov Model (HDP-HMM)

[Beal et al., 2001, Teh et al., 2006],[1] A hidden Markov Model (HMM) is essentially a POMDP without the decision-making component: it does not have actions or rewards, but it still posits that the environment consists of a set of hidden states s that transition according to some transition function $T(s'|s)$ and emit observations according to some observation function $\Omega(o|s)$. The HDP-HMM places a distribution $p(m)$ over HMM tuples $m = \{S, O, T, \Omega\}$ that have a countably infinite number of states s. While there is no closed form expression for $p(m)$, we can draw a sample HMM m from the HDP-HMM prior $p(m)$ by taking the following steps:

1. Draw the mean transition distribution $\bar{T} \sim \mathsf{Stick}(\lambda)$.

2. Draw observation distributions $\Omega(\cdot|s) \sim H$ for each state s.

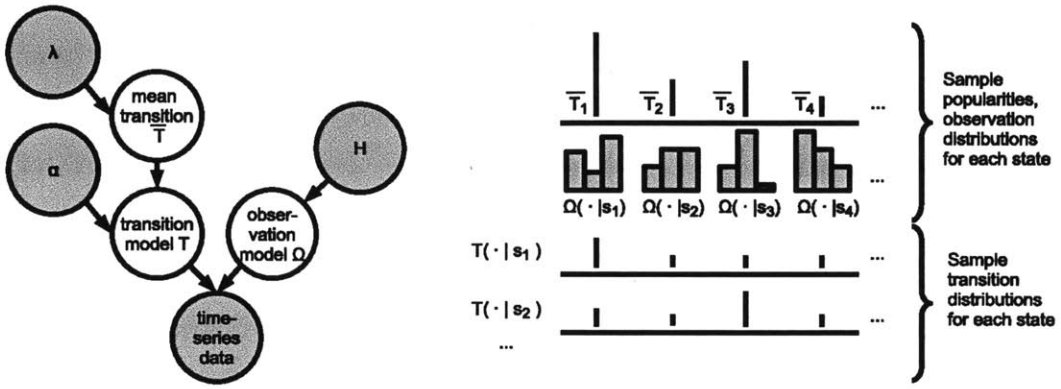
3. Draw transition distributions $T(\cdot|s) \sim \mathsf{DP}(\alpha, \bar{T})$ for each state s.

where Stick() represents a stick-breaking procedure based on the Dirichlet process (DP) [Ferguson, 1973, Teh, 2010], λ is the DP concentration parameter, and H is a prior over observation distributions. For example, if the observations are discrete, then H could be a Dirichlet distribution from which multinomials over the observations are drawn.

This sampling procedure produces HMM models m that have an infinite number of states but whose histories $h_t = \{s_0, o_0, ..., s_t, o_t\}$ typically visit only $\log(t)$ states. The first step, drawing $\bar{T}$, can be thought of as a procedure for assigning a popularity $\bar{T}_k$ to each state k such that the sum of the popularities is one: $\sum_k^\infty \bar{T}_k = 1$. The Dirichlet process provides a bias toward having a few popular states (states k such that $\bar{T}_k > \frac{\lambda}{1+\lambda}$) and and very many unpopular states (states k such that $\bar{T}_k < \frac{\lambda}{1+\lambda}$). By using these popularities $\bar{T}$ as a base distribution, or mean, for the transition distributions $T(\cdot|s)$, we introduce a locality bias so the agent expects to be in the popular states most of the time. However, since the remaining (infinite) states have non-zero popularity, the agent may always transition to somewhere new: a new room

[1]The iHMM models in Beal et al. [2001] and Teh et al. [2006] are formally equivalent [Gael and Ghahramani, 2010].

for a robot, a new set of preferences for a recommender system, a new type of condition for a patient. Figure 1-5 shows the graphical model of the parameters of the HDP-HMM.



(a) HDP-HMM graphical model

(b) Cartoon of some of the HDP-HMM parameters

Figure 1-5: Graphical Model for the parameters of the HDP-HMM.

Our work follows a variety of works in which Bayesian nonparametric models have been used to estimate models of dynamical systems. Many of these [Fox et al., 2010a, 2008, Stepleton et al., 2009, Johnson and Willsky, 2010] are variants of the HDP-HMM. Other approaches [Stolcke and Omohundro, 1993, Shalizi and Shalizi, 2004], while they not explicitly define Bayesian nonparametric models, have a similar flavor to Bayesian nonparametric models in that they automatically infer the size and structure of the underlying model. However, only in rare cases have these models been used for controlling as well as learning systems, and previous work that has used Bayesian nonparametric methods for reinforcement learning [Engel et al., 2005, Deisenroth et al., 2009] has focused largely on continuous domains.

1.2 Contributions of this Research

The core contribution of this thesis is showing how Bayesian nonparametric methods provide a different way about thinking about state in reinforcement learning; indeed, Bayesian nonparametric methods are a natural fit for sequential decision-making problems in which parts of the world can only be learned about once experienced. More

specifically, we present three Bayesian nonparametric models and empirically demonstrate their value for reinforcement learning in partially observable domains.

Infinite POMDPs. Our first model is the infinite POMDP (chapter 4). Building on the HDP-HMM [Beal et al., 2001, Teh et al., 2006], the infinite POMDP posits that the (now controlled) world consists of an infinite number of states. As with the HDP-HMM, the states in an infinite POMDP are no longer identifiable; they do not necessarily correspond to the "true" underlying states of the world. Instead, the concept of a state is a way-point that is useful for making future predictions, something that makes the underlying system more Markovian. We show that learning the state gradually allows us to learn enough to capture the key dynamics of the system, resulting in learning that both requires fewer samples and is computationally more efficient than learning the full "true" model. The next two contributions extend this basic work in two orthogonal directions.

Nonparametric Policy Priors. In chapter 4, the infinite POMDP model is learned only from the agent's own experience. An agent's actions do not provide information about the world; they are simply its policy. However, in some settings, it may be possible to get demonstration trajectories from an expert. As with the agent's data, the expert histories can be used to learn more about the world dynamics T, Ω, and R. However, they provide an additional source of information: the expert's actions are presumably near-optimal. Combining this information with self-exploration is tricky because self-exploration provides direct information about the model, while demonstrations provide direct information about the optimal policy. In chapter 5, we provide a principled way to combine agent experience with expert demonstrations through a model prior that jointly prefers models with fewer states and simpler policies. As expected, combining expert demonstrations with self-exploration results in faster learning for performance.

Infinite Dynamic Bayesian Networks. The infinite POMDP describes the hidden state by a single node or way-point. However, in many applications, it may

make sense to think of the hidden state as consisting of several hidden factors: a robot may have unknown position and velocity; each of a patient's organ systems may be in a different state of health. In chapter 6, we introduce a very flexible prior over factored hidden state spaces, the infinite Dynamic Bayesian Network (iDBN). This prior allows for an infinite number of hidden factors, each of which can take on an arbitrary number of discrete values and have arbitrary inter-node connections. We do not demonstrate the iDBN on a sequential decision-making task, but we do show that it finds interesting, predictive structures compared to other DBN-learning approaches.

1.3 Document Roadmap

This document contains three major sections. First, chapter 2 expands the formalisms introduced in section 1.1, providing the key technical background to read this thesis as a stand-alone text. Chapter 2 also provides pointers to additional papers and tutorials in these fields.

Second, chapters 3 and 7 places the work in the context of other work in reinforcement learning and different types of application domains. Chapter 3 describes related work in partially observable reinforcement learning, focusing on the different notions of state used by different methods. We also describe how the notion of state in Bayesian nonparametric methods relates to these other approaches. Chapter 7 summarizes the empirical results of this thesis, relating these results to what we might expect from a Bayesian nonparametric framework.

Finally, chapters 4, 5, and 6 contain our technical contributions. Many of the contributions in these chapters are already summarized in earlier work [Doshi-Velez, 2009, Doshi-Velez et al., 2010, 2011]. However, chapter 4 in particular has been greatly expanded: not only have we included a thorough evaluation of the iPOMDP in many more domains, we also show the effect of several action-selection strategies on the agent's overall performance. We also provide comparisons between the iPOMDP and a new history-based Bayesian nonparametric model based

on probabilistic-deterministic infinite automata.

More generally, this thesis presents the technical chapters in a more unified manner, with some expanded results, the key additions to this work over the more condensed conference articles are expanded sections on inference and discussion. A special effort has been made to include details and tricks needed to make these models "behave" and to also include where these techniques fail. To this end, each of the three main chapters of the thesis are presented in an identical four-part structure: models, methods, results, and discussion. The models section of each chapter describes the generative process that defines the prior for each model, making explicit key assumptions and discussing how these assumptions manifest themselves in practice. The methods section of each chapter describes the inference techniques needed to derive a posterior over models given data, with a focus on including inference details often glossed over in shorter documents. The results section contains an empirical validation of the approach on various benchmark problems, followed by a discussion on when these techniques work best.

Chapter 2

Background

The work in this thesis combines two areas of machine learning: Bayesian nonparametric statistics and partially-observable reinforcement learning. In this chapter, we provide a brief overview of these areas, as well as pointers to further information. Models for specific applications, such as finite state machines or dynamic Bayesian networks, are described in the chapters in which they appear.

2.1 Reinforcement Learning

The field of reinforcement learning (RL) is characterized by sequential decision-making problems in which the agent's goal is to maximize long-term reward in unknown environments. Specifically, reinforcement learning involves a sequence of exchanges between the agent and the environment. At each time step, the agent interacts with the environment through an action a. The environment sends back an observation o and an immediate reward r (figure 2-1). The agent's goal is to maximize the discounted sum of its expected rewards $E[\sum_t \gamma^t r_t]$, where r_t is the reward that the agent receives at time t, and $\gamma \in [0, 1)$ trades off between the importance of current rewards and future rewards (see Sutton and Barto [1998] for a much more complete introduction).

Three elements differentiate reinforcement learning from related problems in control and optimization. First, we assume that the agent must interact with the envi-

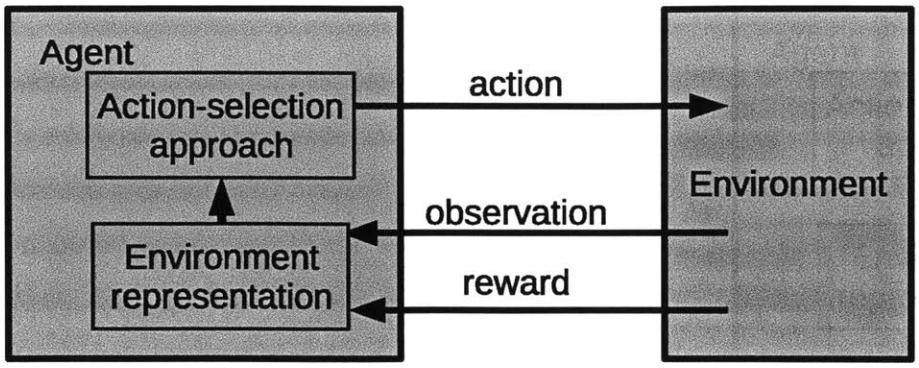


Figure 2-1: General reinforcement learning framework. At each time-step, the agent interacts with the environment with an action and receives an observation and reward. The observation and reward are used to update the agent's representation of the environment and then to select the next action.

ronment over time: all reinforcement learning problems have an element of sequential decision-making. Second, the agent learns through only an immediate reward signal: through trial, error, and inference, the agent must distinguish which actions were responsible for its rewards. Finally, the learning process is sample-based: the agent does not start out with a complete model of the environment and then makes sequential decisions; any and all aspects of the environment must be learned through the agent's experience of experience.

As with most reinforcement learning problems, the partially observable reinforcement learning problem can be divided into two parts [Hutter et al., 2009]: choosing a representation for the environment and deciding how to act given that representation. We emphasize that the representation is internal to the agent: it is how the agent chooses to encode its knowledge about its environment, not the environment itself. In general, the representation will consist of two parts: a part that summarizes the current history h_t and a part that encodes general (usually static) information about the environment. For example, the simple robot from chapter 1 might be simultaneously keeping a distribution over its current location—which summarizes its history h_t—while building a (static) map of its environment. Both of these parts are needed to choose actions.

2.1.1 The Partially Observable Markov Decision Process

In this section, we provide a description of one popular representation, the partially-observable Markov decision process [Sondik, 1971, Kaelbling et al., 1995]. (Other representations are summarized in chapter 3.) A POMDP m is specified by the tuple $\{S,A,O,T,\Omega,R,\gamma\}$, where S, A, and O are sets of states, actions, and observations. The POMDP representation posits that all information needed to make predictions about the environment can be described by a latent variable s that is hidden from the agent. For example, given a robot's current position, its previous history of how it got there is not needed to determine the effect of a movement action.

At each time-step, the transition function $T(s'|s,a)$ gives the probability of transitioning to state s' after taking action a in state s (figure 2-2). The agent then receives an immediate reward $R(s,a)$ for taking action a in state s. What makes the domain partially-observable is that the agent does not also receive the state s. Instead, it only receives an observation o_t which is emitted from the state s_t with probability $\Omega(o|s,a)$. For example, the observation o_t might be a laser scan from the robot's true position s_t. Unlike the current state s_t, the current observation o_t is not sufficient for summarizing the agent's position, and the best action to take at time t may require information from the agent's entire history $h_t = \{a_1, o_1, r_1, ..., a_t, o_t, r_t\}$ of previous actions, observations, and rewards.

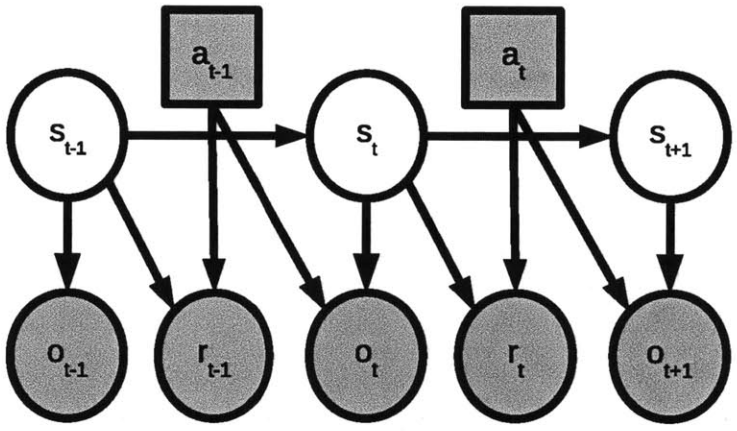


Figure 2-2: Graphical Model for the POMDP. The white nodes s represent the hidden state of the world, and the blue squares a represent the agent's actions. The blue nodes o and r represent the observations and rewards.

As we noted in section 1.1, while POMDPs are not Markovian in the current observation o_t, the current distribution over possible states, called the belief $b_t(s)$, does capture all the information in the history h_t needed to predict future events. In discrete state spaces, the belief at time $t+1$ can be computed from the previous belief, b_t, the last action a, and observation o, by the following application of Bayes rule:

$$b_{t+1}^{a,o}(s) = \Omega(o|s,a)\sum_{s' \in S} \frac{T(s|s',a)b_t(s')}{Pr(o|b,a)}, \tag{2.1}$$

where $Pr(o|b,a) = \sum_{s' \in S} \Omega(o|s',a) \sum_{s \in S} T(s'|s,a)b_t(s)$. The agent starts with some belief $b_0(s)$ that summarizes what states it thinks it may be in before any actions are taken or any observations are received.

Because the POMDP dynamics are Markovian in the space of beliefs $b_t(s)$, the optimal policy is contained in the set of functions $\pi(b(s),a) = p(a|b(s))$ that give the probability of taking action a in belief b. Let $V^\pi(b)$ be the expected long-term reward associated with starting in belief b and then following a policy π. The value of $V^\pi(b)$ can be computed using the Bellman equations [Bellman, 1957]:

$$V^\pi(b) = \sum_a \pi(b,a)(R(b,a) + \gamma \sum_{o \in O} Pr(o|b,a)V^\pi(b^{a,o})) \tag{2.2}$$

where we use the notational shorthand $\pi(b,a) = \sum_s \pi(s,a)b(s)$, and $R(b,a) = \sum_s R(s,a)b(s)$. The term $b^{a,o}$ is the belief obtained after seeing observation o when performing a in b and is computed using equation 2.1.

Solving equation 2.2 gives us the value V^π of a specific policy π. For a belief b, optimal policy π^*, that is, the policy that achieves the highest long-term rewards, can be found by first solving for the value of the optimal policy $V^{\pi^*}(b)$:

$$V^*(b) = \max_{a \in A} Q^*(b,a), \tag{2.3}$$

$$Q^*(b,a) = R(b,a) + \gamma \sum_{o \in O} Pr(o|b,a)V^*(b^{a,o}), \tag{2.4}$$

where $Q(b,a)$ is interpreted as the value of taking action a in belief b and then acting

optimally. The optimal policy is then

$$\pi^*(b) = \arg\max Q^*(b, a). \tag{2.5}$$

The exact solution to equations 2.4 and 2.5 is only tractable for tiny problems, but many approximation methods [Pineau et al., 2003, Spaan and Vlassis, 2005, Smith and Simmons, 2004, Shani et al., 2007, Kurniawati et al., 2008] have been developed to solve POMDPs offline. For even larger problems, forward search techniques can be used to find solutions online [Ross et al., 2008c].

As with all planning and reinforcement learning techniques, an agent using a POMDP as its knowledge representation alternates between two phases when interacting with its environment. Upon receiving an observation, the agent updates its belief over states $b_t(s)$ given the previous belief b_{t-1}, the current action a_t, and the current observation o_t using equation 2.1. This representation update is known as *belief monitoring* or *estimation.* Second, in the *action selection* phase, the agent selects its next action given its current belief $b_t(s)$ and equations 2.4 and 2.5. While different approximations may be used for each of these phases, the phases themselves—incorporating new information (belief monitoring) and then choosing a new action (action selection)—are common to all approaches for acting in POMDPs.

2.1.2 Bayesian Reinforcement Learning

In the reinforcement learning setting, of course, the agent does not have access to the parameters of the model T, Ω, and R; the model m must be learned from the agent's interactions with the world. In the Bayesian reinforcement learning setting, the agent starts out with a prior distribution $p(m)$ over possible models. Given a dataset D of histories h, the agent can compute a posterior over possible models $p(m|D) \propto P(D|m)P(m)$. The model prior $p(m)$ can encode both vague notions, such as "favor simpler models," as well as strong structural assumptions, such as topological constraints among states.

Placing a prior $p(m)$ over possible models is very similar—indeed, mathematically

equivalent—to the initial belief $b_0(s)$ placed over agent states in section 2.1.1. As discussed in Duff [2002], if we think of the prior as an initial belief over models, we can cast the problem of acting in an environment without a known model as a "model-uncertainty" POMDP in which both the model m and the state of the world s are hidden from the agent. We can factor the joint belief $b(s, m)$ as

$$b(s, m) = b(s|m)b(m). \tag{2.6}$$

Conditioned on data D from the agent's histories h, we get

$$b(s, m|D) = b(s|m, D)b(m|D). \tag{2.7}$$

The first term $b(s|m, D)$ can be computed using the belief update in equation 2.1. The second term $b(m|D)$ is simply the posterior over models $p(m|D)$. Describing the reinforcement learning problem as just a very large POMDP implies that now we can use equations 2.4 and 2.5 to determine an optimal policy to maximize expected discounted rewards, even if the dynamics are not initially known.

Of course, having a decision-theoretic formulation of the Bayes-optimal policy does not imply that solving for the policy is computationally tractable. Methods for approximating the optimal policy include sampling a single model m from the posterior belief $b_t(m|D)$ and following that model's optimal policy $pi^*(b, a|m)$ for a fixed period of time [Strens, 2000]; sampling multiple models and choosing actions based on a vote or stochastic forward search [Jaulmes et al., 2005, Doshi et al., 2008, Doshi-Velez, 2009, Ross et al., 2008a]; and trying to approximate the value function for the full model-uncertainty POMDP analytically [Poupart and Vlassis, 2008]. Other approaches [Wang et al., 2005, Kolter and Ng, 2009, Asmuth et al., 2009] try to balance the off-line computation of a good policy (the computational complexity) and the cost of getting data online (the sample complexity).

2.2 Bayesian Nonparametric Statistics: Models and Inference

In statistics, a model is a probability distribution $p(x|\theta)$, where x is some set of data and θ is some set of parameters. The core concept of a Bayesian nonparametric model involves the combination of two ideas: Bayesian models and nonparametric models (see Orbanz and Teh [2010] for a much more detailed overview). A Bayesian model is one in which the parameters of the distribution θ are themselves random variables with some prior distribution $p(\theta)$. We can think of the prior $p(\theta)$ as the distribution over the values that we believe that the parameters θ might take before we have seen any data. The posterior distribution $p(\theta|x)$ captures our uncertainty over the values that the parameters θ might take after seeing the data x. Nonparametric models are models for which the parameter space θ is infinite-dimensional. Thus, a Bayesian nonparametric model is some probability distribution $p(x|\theta)$ in which θ is an infinite-dimensional random variable.

There are two ways in which the prior $p(\theta)$ can be specified. Explicit representations provide a procedure for first sampling the parameters $\theta \sim p(\theta)$ and then the data $x \sim p(x|\theta)$. Having explicit procedures for sampling the parameters and the data ensures that our prior $p(\theta)$ is in fact a probability distribution and that a finite sample of data can be explained by a finite number of parameters. In contrast, implicit representations describe a procedure for generating the data without explicitly sampling the parameters θ first (beneficial because the number of parameters is infinite!). When the prior $p(\theta)$ is never used explicitly, statistical results such as de Finetti's theorem or the Kolmogorov Extension theorem must be applied to ensure that the procedure used to generate the data implies a valid distribution $p(\theta)$. The properties of models are often clearer when the generative process is explicit; inference is often simpler with implicit representations.

Inference, or computing the posterior $p(\theta|x)$, requires special care when the parameter vector θ is infinite-dimensional. The Bayes equation $p(\theta|x) \propto p(x|\theta)p(\theta)$ may not technically exist because the family of all possible posteriors do not dominate,

or overwhelm, the prior. However, conjugacy can allow the posterior $p(\theta|x)$ to be computed given the data x and the prior $p(\theta)$. A properly defined Bayesian nonparametric model has the following properties: First, even though the parameter vector θ is infinite-dimensional, we require that a finite sample of data $x = \{x_1, ..., x_k\}$ can be modeled with only a finite number of those dimensions. Second, asymptotic consistency requires that the effect of the prior $p(\theta)$ disappears with sufficient data (or, alternatively, given infinite data, the parameters will converge to their true values with probability one). A third desirable, but not required, property for these models is exchangeability, which states that the ordering of the data points does not matter when evaluating their probability.

In the remainder of this section, we describe one particular Bayesian nonparametric model, the hierarchical Dirichlet process hidden Markov model (HDP-HMM), that we use extensively in this work. We also provide and overview of several inference techniques that can be used to sample from the posterior over parameters $p(\theta|x)$, which will be necessary for representing the belief over models $b(m)$.

2.2.1 Hierarchical Dirichlet Process Hidden Markov Model

A standard hidden Markov Model (HMM) [Rabiner, 1989] is a model for time-series data which consists of the tuple $m = \{S, O, T, \Omega\}$. The set S is the set of world-states, and the set O is the set of observations accessible to the agent. At each time-step, the current state s_t emits an observation o_t drawn from a distribution $\Omega(\cdot|s)$ and transitions to a new state s_{t+1} drawn from a distribution $T(\cdot|s)$ (figure 2-3). One can think of the HMM as the part of the POMDP that describes how world-states change without the components relevant for decision-making (actions and rewards).

Bayesian approaches to learning HMMs involve putting priors over the transition distributions T and the observation distributions Ω. When the set of states S and the set of observations O are discrete and finite, the multinomial distributions are the most general choice for the transition distribution $T(\cdot|s)$ and the observation distributions $\Omega(\cdot|s)$. As conjugate distributions to the multinomial, Dirichlet distributions are thus a natural choice of priors $p(T(\cdot|s))$ and $p(\Omega(\cdot|s))$.

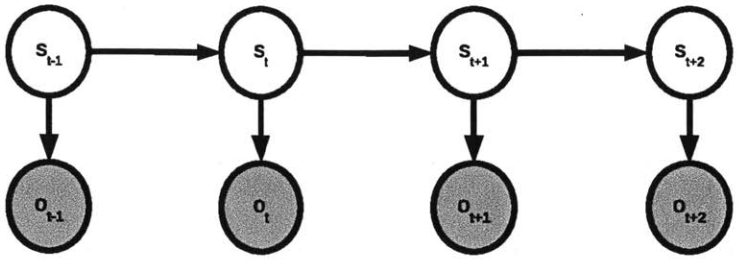


Figure 2-3: Graphical Model for the HMM. The blue observation nodes o are the variables that are observed at each time-step $\{...t-1, t, t+1...\}$; the white nodes s represent the hidden state of world.

The infinite Hidden Markov Model, also known as the hierarchical Dirichlet process Hidden Markov Model (HDP-HMM) [Beal et al., 2001, Teh et al., 2006] places a prior over worlds whose state spaces S are discrete but countably infinite. In the HDP-HMM, the transition distributions T are still multinomials, but over an infinite number of states s. Thus, the natural conjugate prior for a particular transition distribution $p(T(\cdot|s))$ is now a Dirichlet process [Ferguson, 1973, Teh, 2010] rather than a Dirichlet distribution. Using a hierarchical Dirichlet process as a prior for the set of transition distributions T ensures that all the distributions $T(\cdot|s)$ are over the same state space S. It also encodes the prior belief that there are a few popular states to which all states transition.

A formal overview of the HDP-HMM first requires a summary of the Dirichlet process. Recall that a multinomial distribution d with K elements can be encoded by a set of pairs $\{(x_k, \beta_k)\}$ for $k = 1...K$, where β_k is the probability of sampling the element x_k. (Note that $\sum_k^K \beta_k = 1$.) For example, a multinomial distribution over the three colors red, yellow, and blue could be written as { (red, .2) , (yellow , .2) , (blue , .6) }. The Dirichlet process extends the Dirichlet distribution by placing a prior over multinomials $d = \{(x_k, \beta_k)\}$ with a countably infinite number of elements x_k.

The following explicit representation provides the generative procedure to draw a multinomial distribution $d = \{(x_k, \beta_k)\}$ from a Dirichlet process prior $\mathsf{DP}(\lambda, H)$:

1. Draw the samples $x_k \sim H$ for $k = 1...\infty$. The samples x_k define where the probability mass will be placed.

2. Draw the probability β_k of each sample x_k via the following procedure:

   (a) Draw samples $v_k \sim \mathsf{Beta}(1, \lambda)$ for $k = 1...\infty$.

   (b) The probability of x_k under the multinomial distribution d is given by $\beta_k = v_k \prod_{i=1}^{k-1}(1 - v_i)$.

Here, λ is a concentration parameter that governs the ratios of the stick-lengths β_k, and H is some base measure from which we draw samples. We note that since each β_k is the product of more and more variables v_k, the sizes of β_k will decrease exponentially fast in expectation. Thus, a few x_k will be likely and most will will be unlikely.

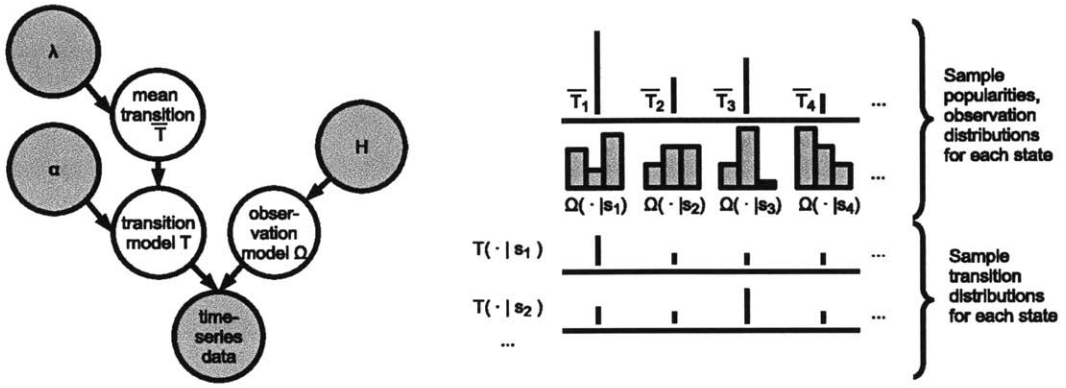
The HDP-HMM uses a pair of Dirichlet processes to create a set of transition distributions over a shared set of states. A draw θ from the HDP-HMM prior consists of the parameters $\theta = \{\bar{T}, \{\Omega(\cdot|s_k)\}, \{T_k\}\}$, where $\bar{T}$ is a mean transition distribution, $\{\Omega(\cdot|s_k)\}$ is the set of observation distributions for each state s_k, and $\{T_k\}\}$ is the set of transition distributions for each state s_k. First, we let the base distribution H be our prior over these observation distributions $\Omega(\cdot|s_k)$. For example, for discrete observations, we can choose a multinomial distribution for Ω and a Dirichlet distribution for H. The process for drawing a sample from the HDP-HMM prior has the following steps:

1. Draw $d \sim \mathsf{DP}(\lambda, H)$. In the context of the HDP-HMM, the atoms $x_k = \Omega(\cdot|s_k)$ describe the observation distributions for state s_k, and sticks $\beta_k = \bar{T}_k$ describe the mean transition probability to state s_k.

2. Draw transition distributions $T(\cdot|s_k) \sim \mathsf{DP}(\alpha, \bar{T})$ for each state s_k.

where λ and α are concentration parameters. A large λ indicates that β_k decays slowly, meaning that we expect many states s_k to have non-negligible visit probabilities. More generally, the exponentially decaying visit probabilities in $\bar{T}$ encodes a prior belief that the agent will spend most of its time in some local region; however, there are an infinite number of states with non-zero visit probabilities. The concentration parameter α determines how closely each transition distribution T_k resembles the

mean transition distribution $\bar{T}$. The graphical model and a cartoon of the generative process is shown in figure 2-4.



(a) HDP-HMM graphical model

(b) Cartoon of some of the HDP-HMM parameters

Figure 2-4: The figure on the left shows the graphical model for the HDP-HMM. On the right is a cartoon of the generative process. First, the stick-breaking process for $\bar{T}$ assigns a mean transition value $\bar{T}_k$ to each state s_k (think of this as a state "popularity"). We also sample observation functions $\Omega(\cdot|s_k)$ for each state s_k. Finally, we sample transition distributions $T(\cdot|s_k) \sim \mathsf{DP}(\alpha, \bar{T})$ for each state s, where α, the concentration parameter for the DP, determines how closely the sampled distribution $T(\cdot|s_k)$ matches the mean transition distribution $\bar{T}$.

2.2.2 Inference

In section 2.2.1, we described the prior for the HDP-HMM as a process for drawing samples θ from $p(\theta)$. Sample-based methods, that is, methods that approximate a distribution $p(\theta)$ with set of particles $\theta_1 ... \theta_n$, are often used for working with Bayesian nonparametric models because an analytical form for the distribution $p(\theta)$ does not exist.[1] In this section, we give an overview of the inference techniques used in this work. These techniques are very general statistical techniques that are particularly well-suited for working with Bayesian nonparametric models.

[1]We also cannot explicitly write down the infinite number of values of the parameters θ; in practice we either set a truncation level for the number of states or only explicitly describe the parameters θ whose posterior value differ from the prior.

Importance Sampling In many situations, we want to draw samples from some distribution $p(\theta)$ but it is much easier to draw samples from some other distribution $q(\theta)$. Importance sampling is a general technique for computing expectations with respect to some distribution $p(\theta)$ with samples from some other distribution $q(\theta)$. Suppose we are trying to compute some expectation

$$E_p[f(\theta)] = \int_\theta f(\theta)p(\theta)d\theta.$$

We can manipulate the integral to be

$$\int_\theta f(\theta)p(\theta)d\theta = \int_\theta f(\theta)\frac{p(\theta)}{q(\theta)d\theta}q(\theta) = E_q[f(\theta)w(\theta)]$$

where

$$w(\theta) = \frac{p(\theta)}{q(\theta)}.$$

The integral can be approximated by the sum

$$E_q[f(\theta)w(\theta)] \approx \frac{1}{N}\sum_{\theta_i \sim q(\theta)} f(\theta_i)w(\theta_i).$$

where N is the number of samples θ_i that we use in approximating the integral. Thus, we can use samples from $q(\theta)$ to compute expectations with respect to $p(\theta)$.

Markov Chain Monte Carlo Markov Chain Monte Carlo (MCMC) describes a set of techniques for drawing samples from $p(\theta)$ in which the previous sample θ_k is used to draw the next sample θ_{k+1} from some transition kernel $t(\theta_{k+1}|\theta_k)$. The samples are correlated, but we can still approximate $p(\theta)$ with a sufficiently long sequence of samples $\theta_1 ... \theta_n$; the length associated for a chain to be representative of a sequence is called the *mixing* time. Furthermore, depending on the starting θ_0, it may take several samples for the sequence to start being representative of the distribution $p(\theta)$. This length is called the *burn-in* time.

One common approach to Markov Chain Monte Carlo is called *Metropolis-Hastings.*

In this method, we first define an arbitrary transition function $q(\theta_{k+1}|\theta_k)$. The sampling procedure for the transition kernel $t(\theta_{k+1}|\theta_k)$ is then described by the following steps:

1. Sample a possible θ^* from $q(\theta_{k+1}|\theta_k)$.
2. Compute the acceptance probability $\alpha = \min(1, \frac{p(\theta^*)q(\theta_k|\theta^*)}{p(\theta_k)q(\theta^*|\theta_k)}$.
3. With probability α, let the next element in the sequence $\theta_{k+1} = \theta^*$. Otherwise, $\theta_{k+1} = \theta_k$.

This procedure ensures that the sequence will represent the distribution $p(\theta)$ under mild conditions. However, the choice of the transition function $q(\theta_{k+1}|\theta_k)$ can have a significant effect on the mixing time.

Using Metropolis-Hastings can be very effective if one has a sense of what might be a good proposal distribution $q(\theta_{k+1}|\theta_k)$. *Gibbs sampling* is special case of Metropolis-Hastings in which the acceptance probability is always one. Given a multi-dimensional vector θ_k of dimension D, the algorithm proceeds as follows:

1. Choose a dimension d to resample. We can choose d by iterating through $1...D$ sequentially or randomly choosing d such that each dimension gets sampled infinitely often in an infinite sequence.
2. Set $\theta_{k+1} = \theta_k$.
3. Set $\theta_{k+1}(d) \sim p(\theta(d)|\theta_k(1)...\theta_k(d-1), \theta_k(d+1)...\theta_k(D))$

A basic extension to the Gibbs sampling scheme is to sample sets of dimensions of θ at one time, instead of only one dimension. For example, if we are sampling parameters in a POMDP model m, we may first resample all of the parameters in the transition function T and the resample all of the parameters in the observation function Ω. This extension is called blocked Gibbs sampling.

Chapter 3

Related Work

Bayesian nonparametric methods provide one alternative for thinking about and learning representations for partially-observable environments. In this chapter, we first provide a formal definition of "state" and describe the different notions of state used in various reinforcement-learning representations. We also provide and overview of the techniques used to learn this state, pointing out their advantages and disadvantages.

3.1 Defining State

In the general reinforcement learning setting, our agent knows its past history $h_t = \{a_1, o_1, r_1, ..., a_t, o_t, r_t\}$. Based on this history, it must make decisions to maximize its expected future rewards. More generally, any predictions that the agent wishes to make about the future may depend on the past history h_t. Following the information-theoretic definition of state described in Shalizi and Klinkner [2004], Tishby et al. [1999], and Wingate [2008], we define the *information* state $s = g(h_t)$ as any statistic of the history h_t which is sufficient to predict the distribution of future rewards and observations.

As a specific example, let us consider the notion of a state in an MDP. Here, the history consists of $h_t = \{a_1, s_1, r_1, ..., a_t, s_t, r_t\}$ where s_t is the state of the world at time t. By construction, the current *world state* s_t captures all the relevant information

about h_t needed for making future predictions; thus the function $g(h_t) = s_t$ satisfies the information-theoretic definition of a state. The POMDP literature continues to use the term "state" to refer to the underlying MDP-state s_t even though s_t is longer a function of history. However, setting the function $g(h_t)$ to the "belief state" $b_t(s)$ satisfies the information-theoretic definition of a state.

More formally, let f_t be the random variable representing the agent's future after time t: $\{a_{t+1}, o_{t+1}, r_{t+1}, ...\}$. The mutual information $I(h_t; f_t)$ captures how much information the past provides about the future. If $s = g(h_t)$ is a sufficient statistic for the history h_t, then $I(h_t; f_t) = I(g(h_t), f_t)$. Another way to state this condition is that if two histories h_t and h'_t have the same statistic s, then $Pr(f_t|h_t) = Pr(f_t|h'_t)$. This information-theoretic definition of state is also sufficient for making optimal predictions from a decision-theoretic perspective [Blackwell and Girshick, 1954].

There exist, of course, many ways of compressing the history h_t into a sufficient statistic. We describe the notion of state used in various reinforcement learning representations below. In some of these cases, the statistic $s = g(h_t)$ is approximate, that is $I(h_t; f_t) > I(g(h_t), f_t)$.

3.2 States from Features of Histories

Strategy One approach to summarizing the history h_t into a statistic $g(h_t)$ is to directly search that history for certain patterns: a subsequence of elements, a suffix, or some more complex feature. For example, noticing that a customer bought the first two books in a trilogy may be sufficient for predicting whether to recommend the third book, regardless of the rest of the customer's purchasing history. The last several moves of a chess player may be sufficient to infer his current strategy, regardless of how he opened the game.

When is it effective? History-based approaches have the advantage of operating directly on the data. There are no hidden variables that need to be defined, inferred, or controlled. Instead, states are simply aspects of the history that are useful for

predicting future rewards or future observations. However, in general it is harder to incorporate direct expert knowledge, such as a sensor calibration, into the learner. Furthermore, because histories are so general, large amounts of experience may be required before the agent discovers what parts of the histories are valuable for making decisions.

3.2.1 States as Windows of History

Representation One of the earliest history-based algorithm was U-Tree [McCallum, 1993]. Let us define h_t as the history from time 0 to time t. The U-Tree algorithm builds a suffix tree of the agents' histories $h_1, h_2, ...h_t$. Each branch of the tree is trimmed to a particular depth, corresponding to how large a window of recent recent history is relevant. The state $s = g(h_t)$ in U-Tree corresponds to which leaf-node the history h_t. If the depth of that leaf-node is d, then all histories h sharing the same d-length suffix will be grouped together into a single state.

Given a particular suffix tree representation, planning is relatively straight-forward: the agent always knows its current state s (that is, which leaf-node is associated with its history h), and that state is always a discrete scalar. The agent can also keep track of how often it has seen various state transitions $T(s'|s, a)$ and rewards $R(s, a)$ to build an MDP model of the dynamics of these node states; once the model is built, choosing actions is simply a matter of solving the MDP.

Learning The tricky part, of course, is how to learn a suffix-tree representation such that the node-state s is a sufficient statistic for the history with respect to the future. The U-Tree algorithm does so by starting with a small tree and then expanding a node—that is, increasing the suffix window—if a statistical test suggests that a split will significantly improve the predictability of future rewards. Thus, we can think of U-Tree as a nonparametric method for automatically determining the necessary suffix depths needed to build sufficient statistics.

However, we find from our own implementations that the choice of both the statistical test and the action-selection approach can dramatically affect the results;

finding robust solutions to these issues is a continuing area of research [Breslow, 1996, Brafman and Shani, 2004, Pchelkin, 2003]. These choices can lead to large numbers of samples being needed for relatively small problems. For example, even in very recent work [Zheng and Cho, 2011], a state-of-the-art implementation of U-Tree require 75,000 interactions with the environment to learn small benchmarks such as the shuttle domain of Chrisman [1992].

Finally, we note a set of related methods that use windows of history as state. Even-Dar et al. [2005] showed that near-optimal behavior in POMDPs can be achieved in a polynomial number of samples: here, the agent resets itself to a specific set of randomized positions and then uses the histories it sees from this position as state (as above, actions are chosen by solving the induced MDP). Dimitrakakis [2010] is a Bayesian work for uncontrolled systems that learns mixtures over models with varying lengths of suffixes. This approach provides an alternative to the statistical tests used in U-Tree, and the authors learn simple, uncontrolled systems with hundreds to thousands of examples. The last approach, also demonstrated on uncontrolled systems, is causal state splitting reconstruction (CSSR) [Shalizi and Klinkner, 2004]. CSSR builds states as collections of suffixes rather than a single suffix to create a more parsimonious representation.

3.2.2 States as History Subsequences: Finite Automata

Learning challenges aside, a fundamental drawback of U-Tree is that the state $s = g(h_t)$ must be a suffix of the history, and sometimes very long suffixes may be needed to capture relatively simple structure. For example, consider a T-intersection that splits into two long hallways that are identical except that they have different rewards at the ends. A suffix-based method that looks only at windows of recent history would require a very long window to realize that the decision of which of corridor to take at the start is all that is needed to predict the final reward well.

Representation An alternative approach to using history suffixes as state is to consider subsequences of the history as the state. One approach to grouping sub-

sequences of the history together is to use finite automata [Rabin, 1963], such a finite state machines [Sunehag and Hutter, 2010], probabilistic-deterministic finite automata [Mahmud, 2010, Pfau et al., 2010], and looping suffix trees [Holmes and Isbell Jr, 2006]. These representations have a set of nodes n; transitions between the nodes are generally deterministic given the action-observation pair (a_t, o_t), and nodes can transition back to themselves.

The nodes and the node transition structure allows the finite automaton to pick out certain patterns such as the initial choice of hallway in the T-intersection, and ignore other information such as the many observations corresponding to traveling down the long hallway. While finite automata are a strictly smaller class of models than POMDPs, they can be made arbitrary close to—and thus their nodes can be sufficient statistics for—any RL environment (similar to uncontrolled automata of Dupont et al. [2005]). Each discrete node can keep track of its transitions and rewards; choosing a policy given a finite automaton simply involves solving the MDP with the nodes as states.

The finite automaton that we described above summarizes the history h_t with a node-state $s = g(h_t)$ that can predict future rewards as well as the history itself, and then that representation is solved to derive a policy. An alternative is to learn the policy directly from the finite automaton, that is, to have each node emit an action. In this form, the finite automaton is usually called a finite state controller [Kim et al., 2000, Charlin et al., 2007, Hansen, 1998]. Policy-based methods that use other features of the history [Aberdeen et al., 2007, Wierstra et al., 2007, Aberdeen and Baxter, 2002] are also closely related in that they seek to learn a representation $s = g(h_t)$ such that s can be used to make near-optimal decisions independently of the history.

Learning Learning finite automata is NP-hard, though several approximation techniques have been developed for learning probabilistic-deterministic finite automata using PAC [Castro and Gavalda, 2008], information-theoretic [Thollard et al., 2000], and Bayesian methods [Pfau et al., 2010]. Each of these methods uses different heuris-

tics to create a bias toward learning compact models while still explaining the data well: whether it is an explicit generalization criterion, an information-gain argument, or Bayesian Occam's razor. We find from our own experience that the heuristics used for learning finite automata can be hard to tune; however, they do have the advantage of learning a compact state-representation directly from the data without the need for any hidden variables. In Mahmud [2010], well-tuned finite automata were used to learn simple grid problems with a few thousand interactions with the environment.

Finally, in the setting where the representation is directly linked to the policy, policy-gradient methods (such as Aberdeen et al. [2007], Wierstra et al. [2007], and Aberdeen and Baxter [2002]) can be used to update the parameters of the representation based on several runs in the environment. The key difficulty with these direct policy-based methods is that knowing what action to take in a particular state s is a much more difficult task than evaluating how well a representation predicts direct rewards. These difficulties often lead to more interactions with the environment being required for even small benchmarks (on the order of 10,000s iterations in Aberdeen et al. [2007]). However, in settings where experts can provide a strong bias toward a good representation, such as in robotics, these direct policy-based methods can do well on complex tasks [Peters and Schaal, 2006].

3.2.3 Predictive State Representations

Representation Finally, another class of history-based methods is the predictive state representation (PSR) [Singh and James, 2004]. PSRs define a dynamical system by a set of *tests* of the form $p(f|h)$ where f is a future of some finite (but possibly variable) length. If we consider all possible sequences of actions and observations for f and h, then it is clear that the set of tests $p(f|h)$ define the system: for any prior history h, we can predict the probability of any future sequence f. A trivial choice of state $s = g(h_t)$ is then the collection of the probabilities of all possible futures $\{p(f|h_t)\}$.

However, this choice of statistic ignores a key aspect of structure found in $\{p(f|h_t)\}$: futures are related to each other in that longer futures are extensions of shorter fu-

tures, and for any length of future T, at least one future $|f| = T$ must occur. If there is a simply underlying process generating the data, such as a finite POMDP, then it can be shown that a finite number of these tests $Q = \{p(f|h_t)\}$ are sufficient for computing the value of any other test $p(\cdot|h_t)$. These tests are known as *core tests* and serve as the state of the PSR $s = Q$. Similar to POMDPs, a set of update equations can be derived to compute Q_t given Q_{t-1}, a_t, and o_t: specifically, we can write $p(f|hao) = \frac{p(aof|h)}{p(ao|h)}$.

Recent work has looked at approximations for this PSR-state [Rosencrantz et al., 2004] with simpler updates, and also methods for near-optimal control given the PSR-state [James et al., 2004, Boots et al., 2011a]. These works have made it possible to scale the use of PSRs to larger environments. However, the PSR-state, expressed as a set of core test probabilities, are already difficult to interpret, and the induced approximations further separate the PSR-state from aspects of the system that are intuitive to human domain experts.

Learning Like the other history-based approaches, a key advantage of PSR learning is that the representation depends directly on statistics of the history: there are no hidden variables. However, even relatively recent work such as Song et al. [2010] tended to have strong constraints when learning PSRs: multiple resets, large numbers of actions, and certain identifiability and ergodicity properties induced by the agent's exploration policy were all needed to learn the parameters of the representation. Very recent work has alleviated some of these concerns [Rosencrantz et al., 2004, Boots et al., 2011a,b] and shown promise on robotics-related problems with only thousands of iterations of experience; it remains to be seen how well these methods scale to problems with other structure.

3.3 States using Hidden Variables

Strategy In the previous section, we described various techniques that derive the state $s = g(h_t)$ directly from the history h_t. Most of the models in this thesis are

derived from an alternative state-space representation which assumes that the environment contains a hidden variables[1] which, if known, would be a sufficient statistic for the history h_t. The partially observable Markov decision process (POMDP) provides the most general formulation of this latent-variable approach, and we use it to introduce several other variants that can be considered other formulations of the same concept.

When is it effective? Historically, the concept of a hidden variable representation in the POMDP derived from application areas in which it was natural to think of the agent being in some "true" world-state that was made ambiguous by inadequate sensing [Sondik, 1971, Kaelbling et al., 1995]. For example, the world-states might correspond to the location of a robot with imperfect GPS or a factory with limited sensors (such as the examples in Pineau et al. [2001] and Kurniawati et al. [2008]). In dialog management applications such as Roy et al. [2000] and Williams and Young [2005], the hidden world-state is generally the task that the user wants the system to perform. This very "grounded" approach to thinking about hidden variables makes the POMDP approach very close to approaches used in the dynamical systems community and systems identification [Maybeck, 1979].

In the setting where the world-state does represent a real, grounded quantity about the world, using a POMDP-like representation allows for model-learning to be split into simpler components: sensors and actuators can be calibrated in laboratory settings where the true world-state, such as a robot's location, can be measured directly. Even when every part of the model cannot be specified independently, if the hidden world-state corresponds to unseen components of a real system, an expert can input the structure of the system into the agent, leaving only a few, very specific parameters to be learned. Incorporating such expert knowledge and calibrations is difficult in the history-based representations described in section 3.3.

In contrast, in applications where sections of the history are highly informative—

[1]We use the term "world-state" for the term "state" as the hidden variable in the POMDP to be consistent with the POMDP literature and still differentiate this use of the world-state from the notion of a "state" as a sufficient statistic of the history with respect to the future.

for example, if an environment has frequent signs or markers—history-based methods may have the advantage. However, as longer histories are required—or more structure is specified—learning a POMDP model may require fewer interactions with the environment than a history-based method. For example, the BA-POMDP [Ross et al., 2008a] learns simple benchmark problems in hundreds to thousands of iterations of experience, compared to the tens of thousands required in some of the history-based methods. However, the BA-POMDP starts out with some knowledge about the size of the underlying state space, which limits its search for models. A more general latent-state method, AIXI, can take two orders of magnitude more interactions to learn similar problems.

3.3.1 Representation

The POMDP model posits that there exists an underlying process that is Markov if the hidden world-states s are known—that is, the world-state is a sufficient statistic for the history. These world-states transition according to some transition function $T(s'|s,a)$ and emit observations according to some observation function $\Omega(o|s,a)$. In this setting, it can be shown that the statistic $b_t(s) = g(h_t)$ is a sufficient statistic for the history h_t [Sondik, 1971]. Moreover, there exists a simple update rule to adjust this sufficient statistic given new experience:

$$b_{t+1}^{a,o}(s) = \Omega(o|s,a) \sum_{s' \in S} \frac{T(s|s',a) b_t(s')}{Pr(o|b_t,a)},$$

The latent world-state in the equation above need not be a discrete scalar. Factored POMDPs are POMDPs in which the world-state is vector-valued quantity, which allows for the encoding of more sophisticated problem structures [Williams and Young, 2005, Guestrin et al., 2001, McAllester and Singh, 1999b, Sim et al., 2008]. For example, world-state may correspond to the location and velocity of a robot. The POMDP representation can also be used to create world-states that encode hierarchical relationships between the hidden variables [Pineau et al., 2001, Toussaint et al., 2008], logic-based systems [Fikes and Nilsson, 1971, Sanner and Kersting, 2010], and

relational variables [Wang and Khardon, 2010]. In all of these cases, the POMDP framework provides a way to construct the sufficient statistic $b(s)$ as distribution over the hidden variables.

The framework also provides a variety of sophisticated solution methods to derive a near-optimal control policy given the model [Pineau et al., 2003, Spaan and Vlassis, 2005, Shani et al., 2007, Smith and Simmons, 2004, Kurniawati et al., 2008, Ross et al., 2008c]. Using a POMDP-based representation to derive the state $b(s)$ does result in more complex update rules than history-based methods based on suffix trees or PDFAs, and the resulting continuous-state MDP is more difficult to solve than the representations in which the statistic s is a discrete scalar. However, modern POMDP solvers, combined with modern computers, can solve even fairly large (10,000s of world-states) POMDPs efficiently.

Finally, there does exist a parallel between defining state as a set of test probabilities, as in the PSR, and defining state as a belief, in the POMDP. The belief $b(s)$ gives a probability distribution over possible hidden states, from which the probability of any future can be derived, while the core test vector q gives the probabilities of various futures happening directly. It can be shown that the dual of the POMDP actually produces a test-based representation that is very similar to the representation used in a PSR [Hundt et al., 2006].

3.3.2 Learning

Unlike the history-based methods, POMDPs cannot simply be learned based on statistics of the histories. Learning POMDPs is a hard problem [Sabbadin et al., 2007], but a variety of methods exist for inferring the transition, observation, and reward functions. The first three methods that we describe, expectation-maximization, (parametric) Bayesian methods, and system identification, all assume that we know the parameters that make up the model (but not their values). These methods are ideal when the hidden world-state corresponds to something tangible, and thus structures and cardinalities can be specified through expert knowledge. For example, if we are building a model for a simple pendulum, we know that the mass and the length are

the parameters that will need to be learned. The last set of methods we describe, nonparametric methods, are suited for more general problems where we may not know the parameters required for the model.

Parametric Methods: Expectation-Maximization, System Identification, and Bayesian Approaches. Learning a POMDP involves inferring the transition, observation, and reward functions T, Ω, and R. Note that if the world-states were known, then this problem is relatively easy: it is no harder than learning the MDP parameters in the history-based methods. The simplest approach for learning the POMDP parameters, expectation-maximization (EM) [Dempster et al., 1977, Rabiner, 1989], alternates between inferring the latent world-states and fitting the model parameters. EM is a greedy optimization approach that is prone to getting caught in local optima. It produces a point-estimate of what model fits the data best, regardless of how much data is available, which can lead to overfitting the model when only a small amount of data is available.

Bayesian approaches attempt to alleviate these issues by placing a prior distribution over the parameters. By computing the posterior over models, rather than a point estimate, they provide their own measure of uncertainty over the parameters. Many works assume that the number of world-states is known and the parameters come from Dirichlet or Gaussian distributions, and they leverage the properties of these distributions to draw samples from, or otherwise approximate, the posterior [Carter and Kohn, 1994, Jaulmes et al., 2005, Doshi et al., 2008, Strens, 2000, Poupart and Vlassis, 2008, Ross et al., 2008a,b]. The AIXI approach [Hutter, 2004, 2007, Veness et al., 2009] computes posteriors over much more general model classes. These approaches tend to require significant computation even to produce approximate posteriors. Also, while keeping posteriors can help the agent know when it does not have enough data to fit the parameters, it does not provide a solution for what the agent ought to do when data is limited.

The last parametric approaches we describe are calibration and system identification, a broad set of techniques used to determine the parameters of a generally

well-defined system model. Calibration can be used in situations in which sensors for observations and actuators for transitions can be individually characterized. When less is known about the system, but one still has strong knowledge about the form of the system—such as a certain environment can be modeled as a set of filters—then system identification can be used to identify the model from well-collected sequences of data.

These parametric approaches have had success in many complex applications in which the world-state and the system are well-understood: then the expert knowledge can be used to learn complex models with relatively little data. However, the grounded-state idea that underlies these approaches is less well-suited for problems where the concept of a hidden world-state is more ambiguous: for example, if the underlying "state" is a patient's health, then one could imagine a variety of definitions of state that could satisfy the required Markov property: ranging from the complete molecular make-up of the patient to the set of all conditions that produce the same clinical phenotype. In other cases, the world-states may "exist" but be difficult and unnecessary to differentiate: for example, all the parts of a long corridor may look very similar, and knowing exactly where one is in the corridor may not be important for traversing it.

These are the kinds of situations in which proponents of history-based methods cite as evidence for not using hidden-variable methods. After all, why introduce a hidden state if it is either ambiguous (as in the healthcare example) or irrelevant (as in the hallway example). History-based methods are designed to pick out predictive features—such as key test result—and compress irrelevant features—such as long stretches of hallway. In the following section, we describe how one can use non-parametric approaches to learn hidden-variable representations. These approaches allow us to keep many of the benefits of hidden-variable approaches, including the ability for experts to bias the learning toward likely structures. However, by turning away from the concept of a world-state as something "real," they are able to learn world-states that, as in history-based methods, are predictive of future events.

Nonparametric Methods: Spectral Techniques and Bayesian Approaches. Most of the history-based methods are nonparametric: they either grow the representation based on the structure in the histories or use various criteria to determine an appropriately-sized representation given a batch of data. We now describe similar techniques for hidden-variable methods (though they are, in general, less commonly used).

Just as with work in the PSR literature, there exist methods to learn the transition and observation dynamics of an HMM using spectral techniques [Song et al., 2010]. The spectral approach allows one to determine an appropriate size for the state space by looking at components in the resulting eigenvalue decomposition. However, this recent work does require certain model assumptions—such as each world-state having a unique observation distribution—and the availability of data from multiple resets. Also, these techniques have not been applied to the case of a controlled system (although the spectral PSR-training approach in Boots et al. [2011b] can be thought of as a generalization of POMDP learning).

Bayesian nonparametric methods provide another alternative to automatically learning the appropriate size for the hidden-variable representation. As with the parametric Bayesian approaches, Bayesian nonparametric approaches place a prior over model parameters. As with all the other POMDP hidden-variable approaches, they posit that there exists a set of underlying world-states that would make the dynamics Markovian. However, closer to the history-based methods, the goal of these world-states is simply to predict future history well (that is, satisfy the Markov assumption) rather than be some representation of what is truly happening in the world. Bayesian nonparametric models and their precursors have been used to find predictive structure in a variety of dynamical systems [Fox et al., 2010a, 2008, Stepleton et al., 2009, Johnson and Willsky, 2010, Stolcke and Omohundro, 1993, Shalizi and Shalizi, 2004, Drescher, 1991].

More specifically, these methods posit that there are, in fact, an infinite number of latent variables that could be used to make the process Markovian (just as a nonparametric history-based method might posit that the entire history might be

needed as a sufficient statistic). However, the Bayesian aspect of the model injects a strong prior bias toward explaining the data with fewer latent states rather than more. These latent states can be thought of as "way-points," features that, if known, would make it possible to predict future events without previous history. In this sense, they bring together some of the key ideas underlying latent-state methods like the POMDP—including advantages such as interpretability and ease of incorporating expert knowledge—with key ideas in history-based methods such as the PSR, whose strength is in building a compact basis to well-predict future data.

In the next three chapters of this thesis, we show several applications of Bayesian nonparametric methods for learning hidden-variable representations. Focusing on problems where the concept of a world-state is not clear, we show that the flexibility of Bayesian nonparametric approaches allows us to learn appropriately-sized models which require both less computation and less data to train than Bayesian parametric models. The structure of the representation still lets us outperform a history-based method using suffix trees as state.

Chapter 4

The Infinite Partially Observable Markov Decision Process

In this chapter, we describe the first of three models that can be used as priors for reinforcement learning in partially-observable domains. Recall from chapter 1 that one of the most common representations used for partially-observable reinforcement learning is the partially observable Markov decision process. The POMDP representation posits that the observation o_t that an agent receives at time t is emitted based on the value of some hidden world-state s_t[1] and the agent's most recent action a_t. The agent's action a_t also causes the world to emit a reward $R(s_t, a_t)$ and transition to a new state s_{t+1}. While very general, the POMDP formulation requires a large number of parameters to specify how observations are emitted $\Omega(o|s, a)$, how states transition $T(s|s, a)$, and how rewards are given $R(s, a)$. When used as a representation for reinforcement learning, all of these parameters must be learned online from the agent's interactions with the environment.

One approach to learning these parameters is to use Bayesian reinforcement learning to convert the reinforcement learning problem into a larger "model-uncertainty" POMDP in which the agent maintains distributions over both the parameters of the

[1]To be consistent with the POMDP literature, we will use the term "state" and the notation s to describe the world-state. We will refer to the information state (described in chapter 3) as the "sufficient statistic for the history."

original POMDP m and the state of the world s (which is hidden). These techniques require that the number of states, at least, is specified to the algorithm in advance. However, as we discussed in chapter 3, if the states are truly hidden, it may be hard to specify how many states a system has, or even what "states" mean. For example, in a medical domain, it may not be clear how many "states" are needed to describe various aspects of the patient's health. Even if the size of the state space is known, just making the agent reason about a large number of unknown parameters at the beginning of the learning process is fraught with difficulties. The agent has insufficient experience to fit a large number of parameters, and therefore much of the model will be highly uncertain. Trying to plan under vast model uncertainty often requires significant computational resources; moreover, the computations are often wasted effort when the agent has very little data. Using a point estimate of the model instead—that is, ignoring the model uncertainty—can be highly inaccurate if the expert's prior assumptions are a poor match for the true model.

We introduce the infinite POMDP, or iPOMDP [2], an alternative prior over representations that posits that the world consists of an infinite number of states. States no longer necessarily correspond to physical aspects of the system; they are abstract entities whose sole function is to render the dynamics of the system Markovian. The iPOMDP places a prior over models of worlds with an infinite number of states. The prior has a bias toward models with a few popular states that the agent is likely to visit over and over and many unpopular states. Thus, during inference, models that explain the agent's data with fewer states will be more probable that models that explain the agent's data with many states. For example, in a robotics domain, if two areas appear and behave similarly, the agent will have a prior bias to inferring that they belong to the same room. Likewise, if two patients with similar symptoms respond similarly to a set of treatments, then the agent will have a bias to inferring

[2]We use the name infinite POMDP and the abbreviation iPOMDP to be consistent with the naming conventions for other Bayesian nonparametric models such as the infinite HMM (iHMM) and the infinite factorial HMM (ifHMM). However, the infinite POMDP model described here should not be confused with the interactive POMDP [Gmytrasiewicz and Doshi, 2004], which is a different model with the same abbreviation.

that they might have had the same condition. However, having an infinite number of states available always allows for something new: whether it is a new room for the robot agent or a new condition for the health-care agent.

From a computational perspective, only states that the agent thinks it has visited must be instantiated with parameters. There is no reason to instantiate parameters for any of the other (infinite) states because there is no data to suggest that the parameter values should have a different distribution than that of the prior. Because of the bias toward explaining the data with fewer visited states, belief-monitoring in the iPOMDP generally involves inference over a relatively small set of model parameters corresponding to agent's limited experience. Additional parameters need to be inferred only as evidence accumulates for the agent observing more states. These small models are often lend themselves to faster planning than a model that tries to explicitly model a large number of parameters.

4.1 Model

The generative process for the infinite POMDP is essentially that of the HDP-HMM described in section 2.2.1. Just as a POMDP can be built from an HMM by adding actions and rewards, the infinite POMDP is built by adding actions and rewards to the HDP-HMM. As review, recall that the specifying an HMM required specifying a transition function $T(s'|s)$ and an observation function $\Omega(o|s)$. In the Bayesian setting, both of these unknown quantities were treated as hidden variables. For each world-state s, the HDP-HMM drew an observation function $\Omega(\cdot|s) \sim H$, where H was some base measure. For example, if the observations o were discrete, drawn from multinomials, H might be a Dirichlet. The transition functions $T(s'|s)$ were drawn from $\mathsf{DP}(\alpha, \bar{T})$, where $\bar{T}$ was a mean transition distribution and the concentration parameter α governs how closely $T(s'|s)$ matches $\bar{T}$. The mean transition distribution $\bar{T}$ was drawn such that the number of world-states was unbounded, but only a few had significant weight.

To convert the HDP-HMM prior into an infinite POMDP prior, we condition the

transition and observation functions on the action a: $T(s'|s,a)$ and $\Omega(o|s,a)$. We only consider settings with a finite number of actions a; in this case we can use the same HDP-HMM prior to now specify transition and observation functions that are conditioned on the action: $T(\cdot|s,a) \sim \mathsf{DP}(\alpha, \bar{T})$ and $\Omega(\cdot|s) \sim H$.[3] In POMDP-based reinforcement learning settings, the reward r is generally treated as a deterministic function of the world-state and current action: $r = R(s,a)$. We relax this convention and assume that the reward is stochastic, where the reward function $R(r|s,a)$ gives the probability of receiving reward r after doing action a in world-state s. Now we can treat the reward as another dimension of the observation o. Just as we placed a prior H over each observation distribution $\Omega(\cdot|s,a)$, we now place a prior H_R over each reward distribution $R(\cdot|s,a)$.

The full process for drawing a model from the iPOMDP prior is then

1. Sample the mean transition distribution $\bar{T} \sim \mathsf{Stick}(\lambda)$. Under the Dirichlet process prior, the popularities of world-state $k = \bar{T}(k)$ decay exponentially. The concentration parameter λ governs this decay: a small value places a bias toward a very few popular world-states; as λ becomes large, many world-states will be likely destinations of a transition.

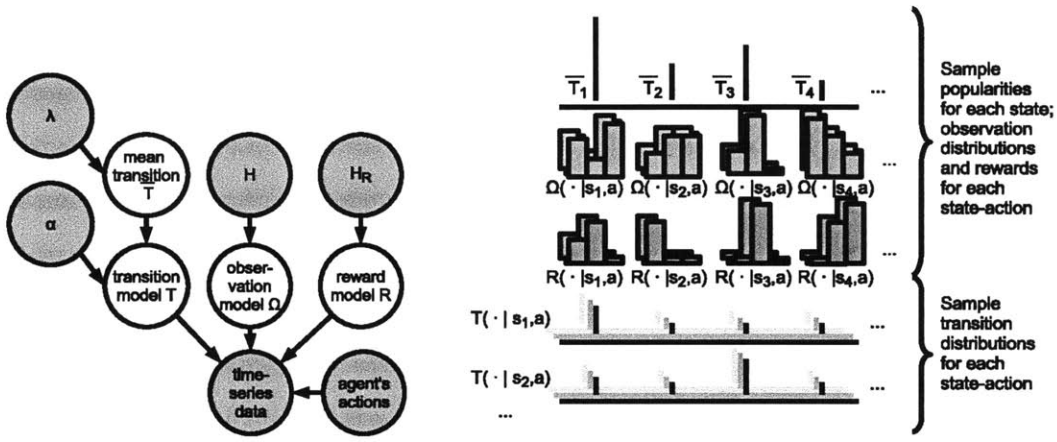
2. For each world-state s and action a, sample a transition function $T(\cdot|s,a) \sim \mathsf{DP}(\alpha, \bar{T})$. The concentration parameter α governs how closely $T(\cdot|s,a)$ matches $\bar{T}$. Large values of α will result in transition functions $T(\cdot|s,a)$ that look almost identical to $\bar{T}$. Small values of α will result in near-deterministic transition functions $T(\cdot|s,a)$ whose mean will still be $\bar{T}$.

3. For each world-state s and action a, sample an observation function $\Omega(\cdot|s,a) \sim H$. The observation function can take any form, and similarly there are no restrictions on the prior H. However, from a computational perspective, it may make sense to choose a prior H that is conjugate to the observation function $\Omega(o|s,a)$.

[3] We use the same base measure H to draw all observation distributions; however, a separate measures H_a could be used for each action if one had prior knowledge about the expected observation distribution for reach action. Likewise, one could also draw a separate $\bar{T}_a$ for each action.

4. For each world-state s and action a, sample a $R(\cdot|s,a) \sim H_R$. Just as there are no restrictions on the observation function, there are no restrictions on the reward function and its prior.

Finally, to complete the specification of the POMDP, we must include a discount factor γ that trades off between current and future rewards. The graphical model is summarized in figure 4-1. We denote the entire POMDP model by the variable $m = \{T, \Omega, R\}$.



(a) iPOMDP graphical model

(b) Cartoon of some of the iPOMDP parameters

Figure 4-1: Graphical Model for the infinite POMDP.

Samples from the iPOMDP prior have an infinite number of states, but fortunately all of these states do not need to be explicitly represented. During a finite lifetime the agent can only visit a finite number of states, and thus the agent can only make inferences about a finite number of states. The remaining (infinite) states are equivalent from agent's perspective, as, in expectation, these states will exhibit the mean dynamics of the prior. Thus, the only parts of the infinite model that need to be initialized are those corresponding to the states the agent has visited as well as a catch-all state representing all other states. In reality, of course, the agent does not know the states it has visited; we address the question of joint inference in the model-state space in section 4.2.

4.2 Methods

Given a prior over POMDP models, we have a starting belief for the joint space of models and states. Acting in this model-uncertainty POMDP requires the same two steps required to act in any POMDP: belief monitoring, to update the distribution over possible states after each action and observation, and action-selection, to choose the next action based on the current belief (see section 2.1.1 for more on these two steps). Each of these steps is described in the context of the infinite POMDP below.

4.2.1 Belief Monitoring

In this section, we describe how to (approximately) compute the joint posterior, or current belief,[4] over states s and models m given a history h of actions, observations, and rewards $\{a_1, o_1, r_1, a_2, o_2, r_2, ..., a_t, o_t, r_t\}$. The posterior $p(s, m|h)$ cannot be represented in closed form. We represent it by first factoring the posterior into $p(s|m, h)p(m|h)$. We represent $p(m|h)$ through a series of samples $\{m\}$. Given a model $m = \{T, \Omega, R\}$, the distribution $p(s|m, h)$ can be computed exactly for discrete spaces using standard belief update equations (see Appendix 2.1.1). Thus, we focus in this section on describing how to draw samples m from $p(m|h)$.

Finally, not only is it intractable to represent the posterior $p(m|h)$ in closed form, it is also impossible to actually "write down" a complete model m because m has an infinite number of states. However, we note that $p(m|h)$ will differ from $p(m)$ only for states s that the agent has visited; the finite history h provides data about only a finite number of states. Thus, it is sufficient to represent the sample m only with the parameters for states that the agent believes it has visited (to be made more formal below). All the other infinite states can be grouped together and having properties that, in aggregate, behave like their base measures.

[4]We will use the words *posterior* and *belief* interchangeably; both refer to the probability distribution over the hidden state given some initial belief (or *prior*) and the history of actions and observations.

Updating $\{m\}$ through Importance Sampling.

The generative process outlined in section 4.1 describes how to draw samples from the prior over models m. One might imagine first drawing a set of samples m_i from the prior (up to some finite number of states K, and then grouping all the other states as a $K+1$ state that acts like the base measures in aggregate), and then never changing those sample points. Instead, we simply update the importance weights on each sample w_i to reflect information from the history h (see Appendix 2.2.2 for background on importance sampling); we know that the weighted set of points will give us an unbiased approximation to the posterior $p(m|h)$.

More generally, suppose we have a set of models m that have been drawn from the correct posterior $p(m|h_t)$ at time t. To get a set of models drawn from the belief at time $t+1$, we can either draw the models directly from the new belief or adjust the weights on the model set at time t so that they now provide an accurate representation of the belief at time $t+1$. Adjusting the weights is computationally most straightforward: directly following belief update equation 2.1, the importance weight $w(m)$ on model m is given by:

$$w_{t+1}^{a,o}(m) \propto \Omega(o|m,a)w_t(m), \tag{4.1}$$

where $\Omega(o|m,a) = \sum_{s\in S} \Omega(o|s,m,a)b_m(s)$, and we have used $T(m'|m,a) = \delta_m(m')$ because the true model does not change.

The advantage of simply reweighting the samples is that the belief update is extremely fast. It is also an effective way of updating the posterior if we do not expect a large change to have happened: usually, a single round of experience from t to $t+1$ is not going to change the agent's beliefs about the model m significantly. However, over time, new experience may render all of the current model samples unlikely. To alleviate this issue, we resample the models m_i from the posterior after every 100 interactions with the environment.

Updating $\{m\}$ through Beam Sampling.

We periodically resample a new set of models directly from the current belief using the beam-sampling approach of van Gael et al. [2008], with a few straight-forward adaptations to allow for observations with different temporal shifts (since the reward r_t depends on the state s_t, whereas the observation o_t is conditioned on the state s_{t+1}) and for transitions indexed by both the current state and the most recent action. The correctness of our sampler follows directly from the correctness of the beam sampler.

Basic Procedure The beam-sampler is an auxiliary variable method that draws samples from the true iPOMDP posterior; we outline the general procedure below. The inference alternates between three phases:

- **Sampling slice variables to limit trajectories to a finite number of hidden states** Given a transition model T and a state trajectory $\{s_1, s_2, \ldots\}$, an auxiliary variable $u_t \sim \mathsf{Uniform}([0, \min(T(\cdot|s_t, a))])$ is sampled for each time t. The final column k of the transition matrix is extended via additional stick-breaking until $\max(T(s_k|s,a)) < u_t.$). Only transitions $T(s'|s,a) > u_t$ are considered for inference at time t.[5]

- **Sampling a hidden state trajectory** Now that we have a finite model, we apply forward filtering-backward sampling (FFBS) [Carter and Kohn, 1994] to sample the underlying state sequence.

- **Sampling a model** Given a trajectory over hidden states, transition, observation, and reward distributions are sampled for the visited states (it only makes sense to sample distributions for visited states, as we do not have information about unvisited states). In this finite setting, we can resample the transitions $T(\cdot|s,a)$ using standard Dirichlet posteriors:

$$T(\cdot|s,a) \sim \mathsf{Dirichlet}(T_1^{sa} + n_1^{sa}, T_2^{sa} + n_2^{sa}, ..., T_k^{sa} + n_k^{sa}, \sum_{i=k+1}^{\infty} T_i^{sa}), \qquad (4.2)\,.$$

[5]For an introduction to slice sampling, refer to Neal [2000].

where k is the number of *active* or used states, T_i^{sa} is the prior probability of transitioning to state i from state s after taking action a, and n_i^{sa} is the number of observed transitions to state i from s after a. The observations and rewards are resampled in a similar manner: for example, if the observations are discrete with Dirichlet priors:

$$\Omega(\cdot|s,a) \sim \mathsf{Dirichlet}(H_1 + n^{o_1 sa}, H_2 + n^{o_2 sa}, ..., H_{|O|} + n^{o_{|O|} sa}). \tag{4.3}$$

As with all MCMC methods, initial samples (from the *burn-in* period) are biased by the sampler's start position; only after the sampler has mixed will the samples be representative of the true posterior. After burn-in, all the samples are drawn from the true posterior and thus have equal weight.

Sampler Notes As with most samplers in practice, the beam sampler in this application tends to find one mode and explore around it, rather than finding multiple modes. Specifically, since the states in the iPOMDP have no meaning, other than as way-points, we would expect that if a sampler were truly mixing, states would occasionally swap meanings. Still, the FFBS sampling procedure does tend to avoid local optima, thus finding better solutions that a basic optimization technique such as expectation-maximization (as will be seen in section 4.4).

To speed up the burn-in period, we "hot-started" the sampler with the most likely sample (based on the importance weights) from the previous round of inference after the first 500 iterations. Using a hot-start too early tended to produce poor results because if the sampler had initially found itself in poor optima (due to limited data), it had trouble escaping it once placed there. However, a hot-start later on, when certain modes had become clearly dominant, sped up the burn-in.

We also experimented with looking at the moving averages and hairiness indices [Brooks, 1998] for various statistics associated with the sampler, including the concentration hyper-parameters λ and α, the number of active states K, and the sum of the active transition parameters T, as measures of mixing. Of these, looking for a

flattening in the moving average for λ proved to be the most effective measure, but in the end we found that a burn-in of 500 samples was almost always sufficient, and a burn-in of 50-100 often worked well if we were using a hot-start.

Computational Notes As mentioned in earlier sections, a full infinite POMDP model m cannot be written down because it has an infinite number of states. Instead, we store only the instantiated values. For example, the transitions are stored as $(K+1)$ x $(K+1)$ x $|A|$ arrays, where K is the number of active states and $|A|$ is the number of actions. The $K+1$ length transition vector associated with each state-action pair is sampled from the Dirichlet distribution in equation 4.2, where the last value represents the probability of visiting any of the infinite uninstantiated states. The transitions from the $K+1$ "state" are simply the Dirichlet parameters from equation 4.2, because we expect, on average, the states to act as the mean.

The rest of the slice-sampling algorithm is relatively straight-forward to implement; using standard vectorization tricks in Matlab speeds up the computation. Finally, it makes sense to cache the random numbers used for the sampling; we empirically found no difference in performance between caching 5000 random numbers and reusing them and always sampling a new random number.

4.2.2 Action Selection

Our belief $b(s,m)$ is a joint distribution over the space of states and models, which we represent by set of sampled models m with exact beliefs over states $b(s|m)$ in each. Reflecting our representation of the belief, we divide our action-selection strategy into two parts: (1) evaluating the value of an action a with respect to some model m and the conditional belief $b(s|m)$, which we write as $Q_m(a, b_m(s))$ and (2) choosing an action given a set of (possibly weighted) models m.

Solving Individual Models The beam-sampling approach outlined in section 4.2.1 alternates between sampling the world-state sequences $s_1 ... s_T$ and the model parameters T, Ω, and R for all visited states. However, there are still an infinite number

of states that the agent does not believe that it has visited. The sampled transition models encode this fact: if the agent believe it has visited K states, then the sum of the transition probabilities $\sum_{k=1}^{K} T(s_k|s,a)$ will be less than 1, reflecting the fact that from any state s, the agent could always transition to somewhere new.

To convert this partially-defined model into something that we can solve, we introduce a new state s^* that represents all of the unvisited states. Collapsing all of the unvisited states into one node is reasonable because we have no data to differentiate these states. We set the transitions into this new state s^* as the "left over probability" $T(s^*|s_i,a) = 1 - \sum_{k=1}^{K} T(s_k|s_i,a)$ for all the visited states s_i, $i = 1..K$. We approximate the remaining parameters for s^* by using mean values from the prior. This approximation ignores the variation in these mean parameters, but it is reasonable because s^* encompasses a large number of states, which, collectively, should act close to their mean. Specifically, we set the transitions from s^*, $T(\cdot|s^*,a)$, to the mean transition function $\bar{T}$, the observations $\Omega(\cdot|s^*,a)$ to their mean from H, and the rewards $R(\cdot|s^*,a)$ to their mean from H_R.

Given a POMDP model m, we can now solve it using standard approximations such as those described in section 2.1.1 (we used point-based value iteration [Pineau et al., 2003]). Many POMDP solution techniques allow us to compute the action-value function $Q_m(a, b_m(s))$, which gives the expected value of performing action a under belief $b_m(s)$ and model m and then following the optimal policy. This action-value function fully takes into account state uncertainty given a model; if there was no model-uncertainty then the action $a = \arg\max_a Q_m(a, b_m(s))$ would be the (near) optimal action for the agent to take. Thus, it follows that as the model uncertainty decreases with more experience, we expect our agent to start behaving near-optimally.

Solving the Model-Uncertainty POMDP In the previous section, we described how we can (approximately) solve an individual model m. Now we describe how we approximate the solution to the full model-uncertainty POMDP, which consists of a set of models, each individually tracking its own state uncertainty. The state space of the model-uncertainty POMDP is too large to apply global solutions methods (such

as PBVI); instead we apply a stochastic forward search in model space which locally approximates the model-uncertainty POMDP solution.

Forward search in POMDPs[Ross et al., 2008c] uses a forward-looking tree to compute action-values. Starting from the agent's current belief, the tree branches on each action the agent might take and each observation the agent might see. At each action node, the agent computes its expected immediate reward $R(a) = \mathbb{E}_m[\mathbb{E}_{s|m}[R(\cdot|s,a)]]$.

From equation 2.4, the value of taking action a in belief $b(s,m)$ is

$$Q(a,b) = R(a,b) + \gamma \sum_o \Omega(o|b,a) \max_{a'} Q(a', b^{ao}) \tag{4.4}$$

where b^{ao} is the agent's belief after taking action a and seeing observation o from belief b, and we can compute the value of $R(a,b)$ with

$$R(b,a) = \sum_m b(m) \sum_s b(s|m) R(s,a) \tag{4.5}$$

where $b(m)$ is simply the weight $w(m)$ on the model m. The update to the conditional belief $b^{ao}(s|m)$ can be computed in closed form using equation 2.1. To update the belief over models $b^{ao}(m)$, we use equation 4.1 to update the belief over models $b(m)$ via the weights $w(m)$. Equation 4.4 is evaluated recursively for each $Q(a', b^{ao})$ up to some depth D.

The number of evaluations $(|A||O|)^D$ grows exponentially with the depth D, so doing a full expansion is feasible only for very small problems. We approximate the true value stochastically by sampling only a few observations from the distribution $P(o|a) = \sum_m P(o|a,m)w(m)$. Equation 4.4 reduces to

$$Q(a,b) = R(a,b) + \gamma \frac{1}{N_O} \sum_i \max_{a'} Q(a', b^{ao_i}) \tag{4.6}$$

where N_O is the number of sampled observations and o_i is the i^{th} sampled observation.

Once we reach a prespecified depth in the tree, we must approximate the value of the leaves $Q(a, b^f)$, where f is the future that corresponds the actions and observations

along the branches from the root b to the leaf. For each model m in the leaves, we can efficiently compute the value $Q_m(a, b_m(s))$ from our approximate solution to the POMDP m. We approximate the value of action a as

$$Q(a, b^f) \approx \sum_m w(m) Q_m(a, b_m^f(s)). \tag{4.7}$$

This approximation is always an overestimate of the value, as it assumes that the uncertainty over models—but not the uncertainty over states—will be resolved in the following time step (the proof follows directly from the fact that equation 4.7 applies the QMDP approximation [Littman et al., 1995] in the space of models). As the iPOMDP posterior becomes peaked and the uncertainty over models decreases, the approximation becomes more exact.

The quality of the action selection largely follows from the bounds presented in McAllester and Singh [1999a] for planning through forward search. The key difference is that now our belief representation is particle-based; during the forward search we approximate expected rewards over all possible models with rewards from the particles in our set. Because we can guarantee that our models are drawn from the true posterior over models, this approach is a standard Monte Carlo approximation of the expectation. Thus, we can apply the central limit theorem to state that the estimated expected rewards will be distributed around the true expectation with approximately normal noise $N(0, \frac{\sigma^2}{n})$, where n is the number of POMDP samples and σ^2 is a problem-specific variance.

4.3 An Alternate Model: Infinite Deterministic Markov Models[6]

In chapter 3, we divided methods for creating a sufficient statistic of the history into two broad categories: those that operated directly on the history and those

[6]**This work was joint with David Pfau and Frank Wood**

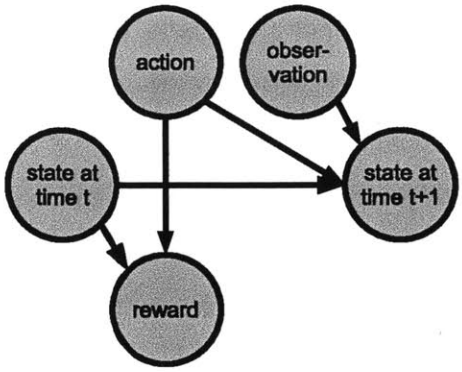


Figure 4-2: Graphical Model showing a time-slice of the PDFA. Note that variables are observed at all times.

that posited the presence of hidden variables. The focus of this chapter has been the infinite POMDP, which uses Bayesian nonparametric methods for learning flexible hidden-variable representations, but in this section, we describe an alternate set of approaches, all based on the probabilistic-deterministic infinite automata (PDIA) [Pfau et al., 2010], that use Bayesian nonparametric methods for a learning flexible history-based representation. Motivated by recent work by Mahmud [2010] that suggested that PDFAs provide a more succinct way to form sufficient statistics of the history than POMDPs, we provide an empirical comparison of the PDIA-based approaches and the iPOMDP approaches in section 4.4.

Mahmud [2010] consider a generalization of the PDFA that includes actions and rewards, also referred to as a Deterministic Markov Model (DMM). Figure 4-2 shows the graphical model for the DMM: at each time step, the node-state s transitions to a new node-state s' given the action a and the observation o. It also emits a reward r. Unlike in the POMDP, the node-state s is always visible: given the most recent action a and the most recent observation o, the transition from s to s' is deterministic. Stochasticity in the transitions arises when the input stream of actions a and observations o are stochastic. However, the DMM does not try to model the stochasticity in the observations; instead, it simply uses them as input when determining transitions. In this way, the DMM can be used to pick out certain patterns in a history without trying to model all aspects of the environment.

Formally, a DMM is a 6-tuple $(S, A, \Omega, \delta, R, s_0)$, where S, A, and Ω are discrete sets of node-states, actions, and observations. The state s_0 is the initial node-state. The deterministic transition function $\delta : S \text{ x } A \text{ x } \Omega \rightarrow S$ outputs the state which follows the current state, action, and observation. The reward function $R(r|s,a)$ gives the probability of receiving reward r after doing action a in node-state s. Inferring a DMM corresponds to finding reward distributions $R(r|s,a)$ and transition mappings $s' = \delta(s,a,o)$ so that rewards can reliably be predicted. The observations indirectly influence model learning through the transition function $\delta(s,a,o)$; an observation model $\Omega(o|s,a)$ is inferred for the purposes of planning is not part of the formal DMM specification.

4.3.1 Model

The Bayesian reinforcement learning approach involves defining a prior over representations. In a recent work, Pfau et al. [2010] proposed a nonparametric prior over PDFA with an unbounded number of states, called the *probabilistic deterministic infinite automata* (PDIA). The prior biases the learned models towards node-state reuse, thus, it can elegantly scale the number of node-states as more complex elements appear in histories, without restricting the space of PDFAs considered[7]. In this section, we extend the PDIA to the infinite Deterministic Markov Model (iDMM) by augmenting it with actions and rewards.

The original PDIA prior, designed for an uncontrolled system, has a generative process based on the hierarchical Pitman-Yor process[8] for the transitions [Teh, 2006]:

$$\mathcal{G} \sim PY(c_0, d_0, H) \tag{4.8}$$

$$\mathcal{G}_o \sim PY(c_1, d_1, \mathcal{G}) \qquad \forall o \in \Omega \tag{4.9}$$

$$\delta(s,o) \sim \mathcal{G}_o \qquad \forall o \in \Omega, s \in S \tag{4.10}$$

[7]In contrast, the prior work of Mahmud [2010] used a uniform prior, but restricted the search space.

[8]The Pitman-Yor process is an extension of the Dirichlet process that allows for the generation of heavy-tailed distributions.

where, as with the HDP-HMM, the base distribution H is the prior over observation distributions $\Omega(\cdot|s)$, and the constants c_0, d_0, c_1, and d_1 are the parameters of the Pitman-Yor process. Here, $\mathcal{G}$ can be thought of a mean transition function and $\mathcal{G}_o$ is the mean transition function for each observation o (encoding the bias that we expect the most recent observation o to be the most informative when predicting the next node-state). Recall that transitions in the PDIA are deterministic; thus given the distribution $\mathcal{G}_o$, we still have to sample a deterministic $s' = \delta(s, o)$ for each state s.

Adding rewards to extend to the PDIA prior to the iDMM is straight-forward: as in previous work with PDFAs [Mahmud, 2010], we assumed that each node s had some reward emission probability $R(r|s, a)$. However, deciding how to incorporate actions requires modeling choices on what biases should be encoded. We considered three different way of incorporating action:

1. **Consider observations then actions** To encode an explicit bias for the most recent observation to be more informative than the most recent action, we can extend the hierarchy for sampling transitions in the following manner:

$$\bar{T} \sim PY(c_0, d_0, H) \tag{4.11}$$

$$\bar{T}_o \sim PY(c_1, d_1, \bar{T}) \quad \forall o \in \Omega \tag{4.12}$$

$$\bar{T}_{oa} \sim PY(c_2, d_2, \bar{T}_o) \quad \forall o \in \Omega, a \in A \tag{4.13}$$

$$\delta(s, a, o) \sim \bar{T}_{oa} \quad \forall o \in \Omega, a \in A, s \in S \tag{4.14}$$

   Here, all of the transition distributions $\bar{T}_{oa}$ will be similar to $\bar{T}_o$, and thus the observation will have the stronger generalizing influence (see figure 4-3 for the graphical model).

2. **Consider actions then observations** In contrast, if we believe that the most recent action is more informative than the most recent observation, we can

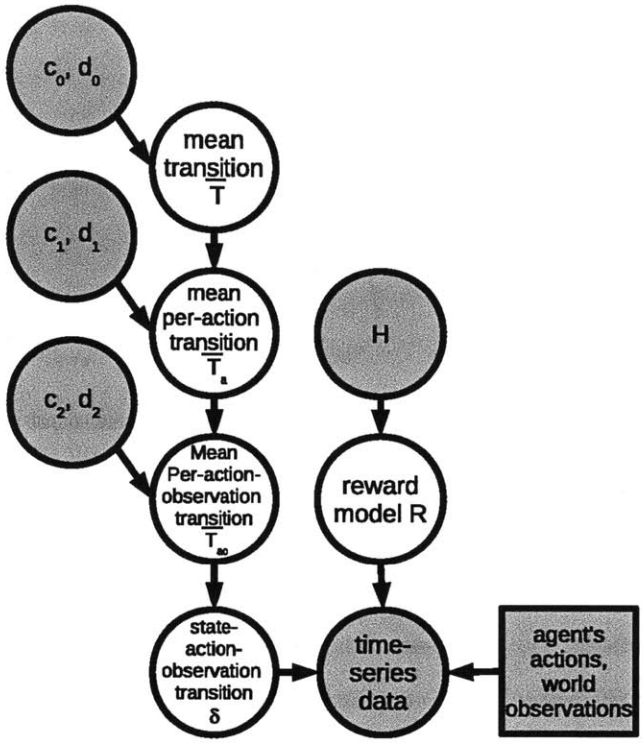


Figure 4-3: Asymmetric iDMM graphical model, in which we believe *a priori* that actions are more informative than observations in determining the next state. Switching the order of actions and observations in the generative process for the transition function would result in an asymmetric iDMM in which observations are more informative than actions.

reverse the ordering of actions and observations in the hierarchy:

$$\bar{T} \sim PY(c_0, d_0, H) \tag{4.15}$$

$$\bar{T}_a \sim PY(c_1, d_1, \bar{T}) \quad \forall a \in A \tag{4.16}$$

$$\bar{T}_{ao} \sim PY(c_2, d_2, \bar{T}_a) \quad \forall a \in A, o \in \Omega \tag{4.17}$$

$$\delta(s, a, o) \sim \bar{T}_{ao} \quad \forall a \in A, o \in \Omega, s \in S \tag{4.18}$$

Here, all of the transition distributions $\bar{T}_{oa}$ will be similar to $\bar{T}_a$, and thus the action will have the stronger generalizing effect.

3. **Consider actions and observations equally** Incorporating the actions by extending the hierarchy forces a bias towards either actions or observations being

more informative than the other (depending on their ordering). Our final model uses a pair of one-level transition models that removes this bias. Specifically, we break the transition function into two stages. In the first stage, the agent takes an action, gets a reward, and updates its node-state (before receiving the observation). In the second stage, the agent receives an observation and updates its node-state based only the observation. Thus the full transition function can be expressed as the composition of two partial-transition functions, $\sigma : SA \to S'$ and $\tau : S'\Omega \to S$, where S' is the set of *intermediate* states, following an action but before an observation. We then use the original PDIA prior for both partial-transition functions:

$$\bar{T} \sim PY(c_0, d_0, H) \tag{4.19}$$

$$\bar{T}_a \sim PY(c_1, d_1, \bar{T}) \quad \forall a \in A \tag{4.20}$$

$$\sigma(s, a) \sim \bar{T}_a \quad \forall a \in A, s \in S \tag{4.21}$$

$$\bar{T}' \sim PY(c'_0, d'_0, H') \tag{4.22}$$

$$\bar{T}'_o \sim PY(c'_1, d'_1, \bar{T}') \quad \forall o \in \Omega \tag{4.23}$$

$$\tau(s', o) \sim \bar{T}'_o \quad \forall o \in \Omega, s' \in S' \tag{4.24}$$

$$\delta(s, a, o) = \tau(\sigma(s, a), o) \tag{4.25}$$

Since node-states in S can only transition to node-states in S', and vice versa, the combined transitions σ and τ form a symmetric graph, hence we refer to this model as a *symmetric iDMM* (see figure 4-4 for the graphical model).

4.3.2 Methods

The belief-monitoring step, which requires performing the inference over iDMMs, follows directly from the Metropolis-Hastings sampler used in the original PDIA work of Pfau et al. [2010], and we do not describe it further here. Given a set of iDMMs, we use a very simple action-selection strategy. First, we choose one of the iDMMs based on its importance weight. Second, we choose the action preferred by that iDMM

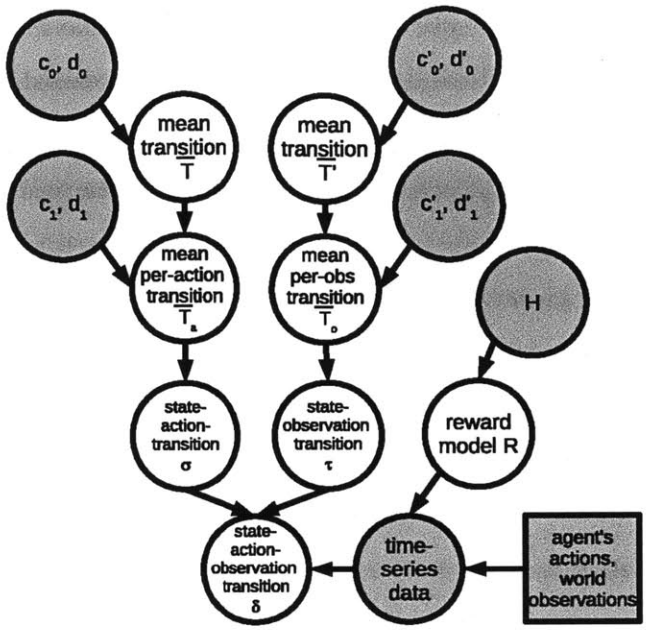


Figure 4-4: Symmetric iDMM graphical model, in which actions and observations are equally informative.

using standard MDP-solution techniques.

4.4 Experiments

In this section, we describe a series of experiments that we performed using the iPOMDP as our prior. We first provide a set of illustrations that demonstration the properties of the Bayesian nonparametric approach to learning POMDP representations. Next, we compare the iPOMDP to parametric approaches to learning POMDPs, the iDMMs, and U-Tree on several benchmark problems. Finally, we apply a suite of action-selection strategies to three learning techniques on two benchmark domains to analyze the interactions of the learning techniques (which correspond to methods for belief-monitoring), and action-selection.

In all of the experiments, we used the following settings for any applicable priors and inference:

- **Observation Prior** H We used a uniform Dirichlet prior for H with a concentration $H_o = 1$ for each element. Starting with a single pseudo-count per

element provided a bias toward smoother observation distributions.

- **Reward Prior H_R** We assumed that rewards took on discrete values and used a uniform Dirichlet prior for H_R with a concentration $H_o = .1$ for each element. Using a low concentration encoded our prior belief that $R(r|s,a)$ was highly peaked.

- **Updating Models** Beliefs were approximated with a sample set of 10 models. Models were updated after every 100 interactions of experience (after an initial agent run of 250 interactions). When running the beam-sampler to resample models, we used an initial burn-in of 50 iterations and then took every 10th sample as an output. Each round of MCMC was "hot-started" with the last model from the previous round.

- **Evaluating Agent Progress** Following each full update over models using MCMC, we ran 50 "catch" test episodes (not included in the agent's experience) with the new models and policies to empirically evaluate the current value of the agents' policy.

- **Solving Models** Models were solved using PBVI [Pineau et al., 2003]. For each model, 500 beliefs $b(s|m)$ were sampled for running the solver; the number of backups was increased from 10 to 35 linearly with the interactions of experience (so that we would spend more effort trying to solve models in the later stages of the learning process, when they were presumably more accurate).

- **Selecting Actions** We used a forward-search of depth 3.

- **Trial Length** A trial consisted of 7500 interactions with the environment. Within each trial, each episode was capped at 75 interactions of experience.

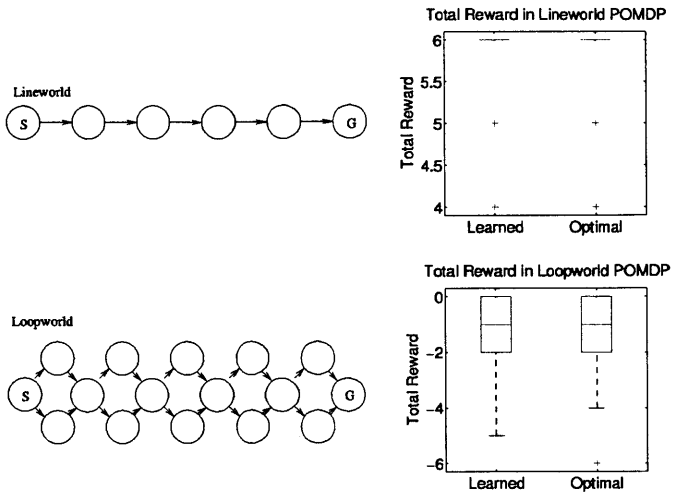
- **Repeated Trials** Each learning trial was repeated 10 times.

4.4.1 Illustrations

We begin with a pair of illustrative examples demonstrating the properties of the iPOMDP. The first, lineworld and loopworld, shows how the iPOMDP learns only the structure that is needed to make predictions. The second, tiger-3, shows how the infinite capacity of the iPOMDP allows it to adapt when "additional state" is added to the environment.

Avoiding unnecessary structure: Lineworld and Loopworld. We designed a pair of simple environments to show how the iPOMDP infers states only as it can distinguish them. The first, lineworld, was a length-six corridor in which the agent could either travel left or right. Loopworld consisted of a corridor with a series of loops (see figure 4-5(a)); now the agent could travel though the upper or lower branches. In both environments, only the two ends of the corridors had unique observations. Actions produced the desired effect with probability 0.95, observations were correct with probability 0.85 (that is, 15% of the time the agent saw an incorrect observation). The agent started at the left end of the corridor and received a reward of -1 until it reached the opposite end (reward 10).

The agent eventually infers that the lineworld environment consists of six states—based on the number of steps it requires to reach the goal—although in the early stages of learning it infers distinct states only for the ends of the corridor and groups the middle region as one state. The loopworld agent also shows a growth in the number of states over time (see figure 4.4.1), but it never infers separate states for the identical upper and lower branches. By inferring states as they are needed to explain its observations—instead of relying on a prespecified number of states—the agent avoided the need to consider irrelevant structure in the environment. Figure 4-5(b) shows that the agent (unsurprisingly) learns optimal performance in both environments.



(a) Cartoon of Models (b) Performance

Figure 4-5: Both lineworld and loopworld have hallways with a start "S," goal "G," and identical middle states (a). In (b), the distribution of rewards from the final iPOMDP episode (left) is almost identical to the that of the optimal policy (right).

Adapting to new situations: Tiger-3. The iPOMDP's flexibility also lets it adapt to new situations. In the tiger-3 domain, a variant of the tiger problem of Littman et al. [1995] the agent had to choose one of three doors to open. Two doors had tigers behind them ($r = -100$) and one door had a small reward ($r = 10$). At each time step, the agent could either open a door or listen for the "quiet" door. Each attempt at listening identified the good door correctly with probability 0.85.

The reward was unlikely to be behind the third door ($p = .2$), but during the first 100 episodes, we artificially ensured that the reward was always behind doors 1 or 2. The improving rewards in figure 4.4.1 show the agent steadily learning the dynamics of its world; it learned never to open door 3. The dip in figure 4.4.1 following episode 100 occurs when we next allowed the reward to be behind all three doors, but the agent quickly adapts to the new possible state of its environment. In this way, the iPOMDP enabled the agent to first adapt quickly to its simplified environment but add complexity when it was needed.

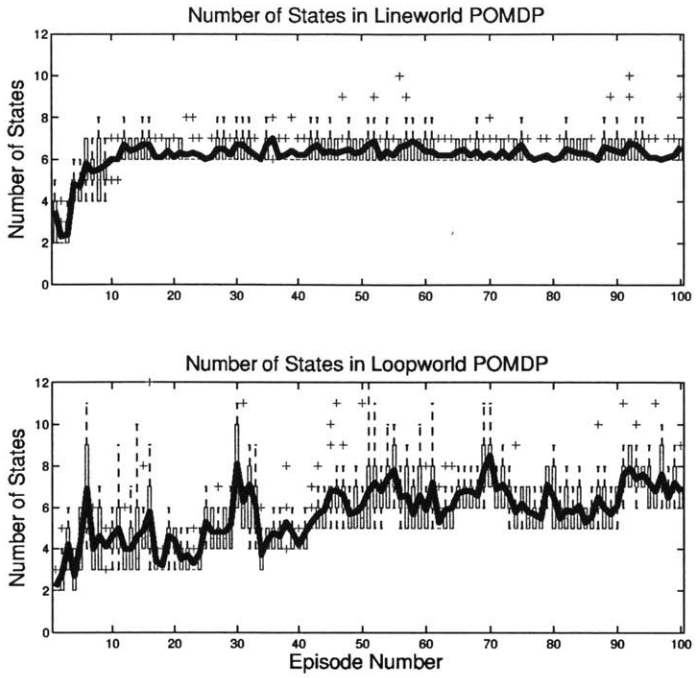


Figure 4-6: The plots show the number of states inferred by the iPOMDP against the number of times the agent has traversed the hallways over 50 repeated trials: the black line shows the mean number of inferred iPOMDP states, and boxplots show the medians, quartiles, and outliers at each episode. Of note is that loopworld infers only necessary states, ignoring the more complex (but irrelevant) structure.

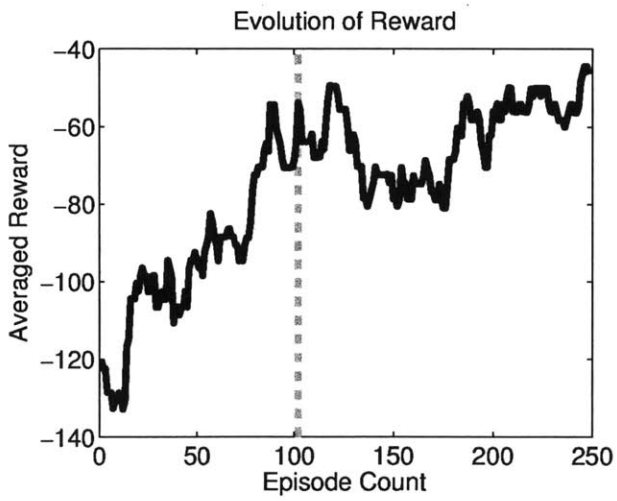


Figure 4-7: Evolution of reward from tiger-3. The agent's performance dips slightly after the third door is introduced, but it is quickly able to learn a representation for the new environment.

4.4.2 Results on Standard Problems

We next completed a set of experiments on POMDP problems from the literature (listed in table 4.1). We compared the accrued rewards for the iPOMDP agent and iDMM variants with five baselines:

- **EM**, an agent that knew the "true" state count K and used expectation-maximization (EM) [Dempster et al., 1977] to train its model. EM is a greedy, hill-climbing approach that quickly finds an optimum in the parameter space m.

- **FFBS**, an agent that knew the "true" state count K and that used the forward-filtering backward-sampling (FFBS) algorithm to sample models m from the posterior over finite models with K states. The prior used the same hyperparameters as the iPOMDP, except that the prior over the transitions T was now a Dirichlet distribution rather than a Dirichlet process. FFBS is used in the inner loop of the iPOMDP beam-sampler to sample state sequences once a finite model has been sliced; thus the only difference between FFBS and iPOMDP was that FFBS considered a class of finite models.

- **EM-Big**, an agent that used EM with ten times the "true" number of states $10K$. This option represented a "safe" strategy if the number of hidden states was not initially known; by guessing too high one could guarantee that the belief-state $b(s)$ would be a sufficient statistic for the history.

- **FFBS-Big**, an agent that used FFBS with ten times the true number of states $10K$, and

- **U-Tree**, an optimized version of the U-Tree algorithm [McCallum, 1993] that used suffixes of the history as a sufficient statistic for predicting future rewards. Specifically, our version of U-Tree did an all-pairs comparison when considering whether to split the nodes. We also searched over different values of the KS hypothesis test threshold α_{KS}, minimum node-size thresholds, and number of

Table 4.1: Summary of iPOMDP Benchmarks

Domain	States	Actions	Observations
Tiger [Littman et al., 1995]	2	3	2
Network [Littman et al., 1995]	7	4	2
Shuttle [Chrisman, 1992]	8	3	5
Cheese [McCallum, 1993]	23	4	8
5x5 Gridworld (adapted from [Littman et al., 1995]	25	4	16
Follow (adapted from [Ross et al., 2008a]	26	5	6
Hallway [Littman et al., 1995]	60	5	21
Beach [Doshi-Velez et al., 2010]	100	5	2
Rocksample [Smith and Simmons, 2004]	257	9	2
Image [Doshi-Velez et al., 2010]	673	5	9
Tag [Pineau et al., 2003]	870	5	30
Harvest [Mahmud, 2010]	896	8	7

MDP backups to find the best performing settings. The tree-depth was limited to 6 for computational reasons.

Figure 4-8 plots the iPOMDP agent's learning curve for one of problems, shuttle. In Figure 4-9, we plot the iPOMDP agent's learning curve, averaged over multiple trials, against all of the other baselines. The iPOMDP and the FFBS agents lead the other methods with very similar learning curves; however, the iPOMDP agent does so without knowledge of the number of hidden variables needed to encode the environment. While the iDMM agents outperform their history-based baseline U-tree, their learning rates are slower than the hidden-variable approaches.

Figure 4-10 shows the rewards from catch tests on several benchmark problems. The problems are organized in order of size: tiger has only two underlying states, while tag and harvest have over 800. All algorithms have difficulty with the larger problems; the variations in performance are most clear in the small and mid-sized problems. The first major difference between the algorithms is that the history-based representations, including the iDMM, tend to learn slower than the hidden-variable representations. This observation is consistent with prior work (see chapter 2 in which model-based (hidden-variable) approaches appear to have lower sample complexity).

However, we also observe the success of a representation depends on the learning technique used to discover it: for example, we optimized several parameters in U-Tree, but the approach had many more knobs than we could tune. Similarly, we used the basic inference of Pfau et al. [2010] for the iDMM; a more sophisticated sampler may have yielded better results.

The impact of the inference techniques is clear when one compares the EM-based approaches with the iPOMDP and FFBS-based approaches. FFBS is a clean, robust inference technique, and the beam-sampler inherits many of its good properties. As a result, these samplers are able to find high-probability samples m and not get caught in local optima (unlike EM). This effect is most clear in tiger, where EM quickly finds the locally optimal greedy strategy of never opening a door—and thus has the worst performance. FFBS and iPOMDP often have similar performance—however, iPOMDP achieves this performance without having to pre-specify the number of states (which would not be available in a real application). The advantage of using the iPOMDP is clear when one observes the less consistent performance of the "big" versions of the algorithms: inferring the number of states (iPOMDP) does as well or better than knowing the true number of states, whereas guessing conservatively leads to poorer performance.

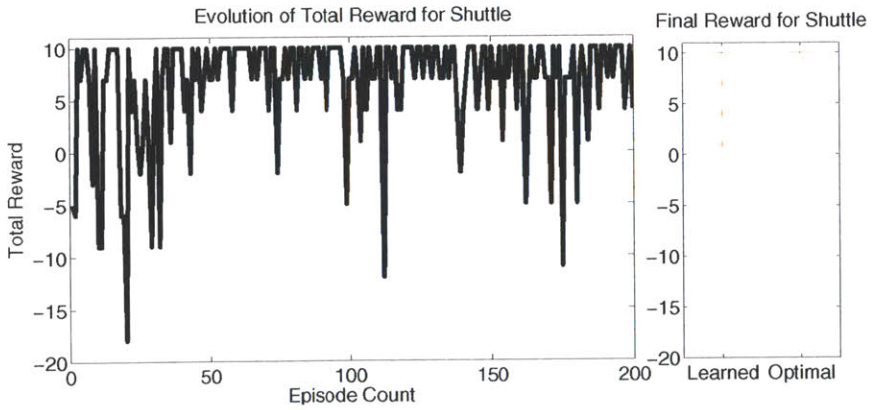


Figure 4-8: Evolution of reward in a single trial for shuttle. During training (left), we see that the agent makes fewer mistakes toward the end of the period. The boxplots on the right show rewards for 100 trials after learning has stopped; the performance distribution for the policy learned by the iPOMDP-agent has all the quantiles are compressed at the optimal value except for a few outliers.

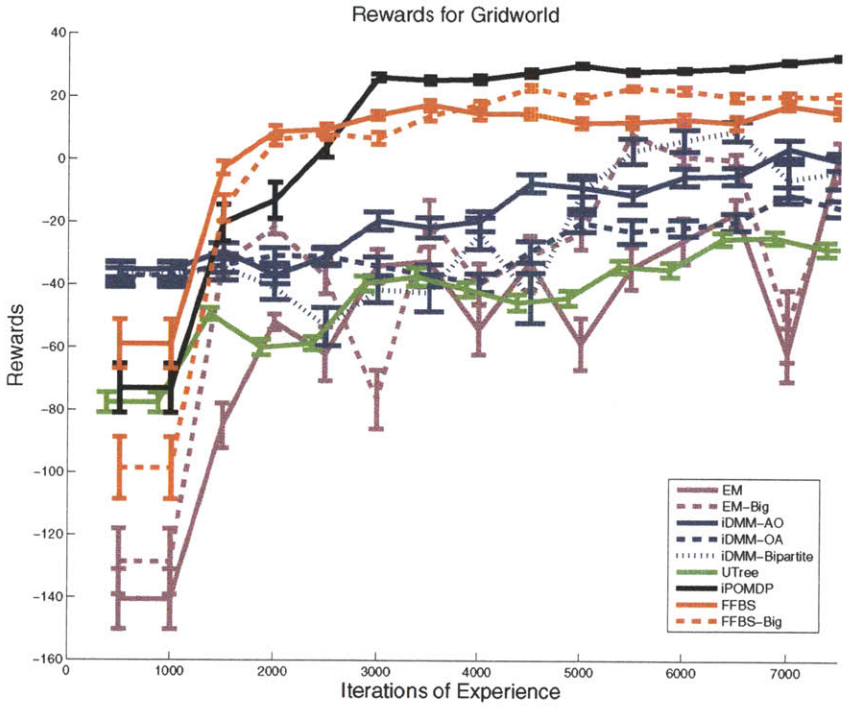


Figure 4-9: Learning rates for various algorithms in the gridworld domain. The iDMM models outperform U-Tree, a simpler history-based learning method, but do not outperform the hidden-variable methods. Among the hidden-variable methods, the Bayesian methods (iPOMDP and FFBS) outperform EM by avoiding local optima.

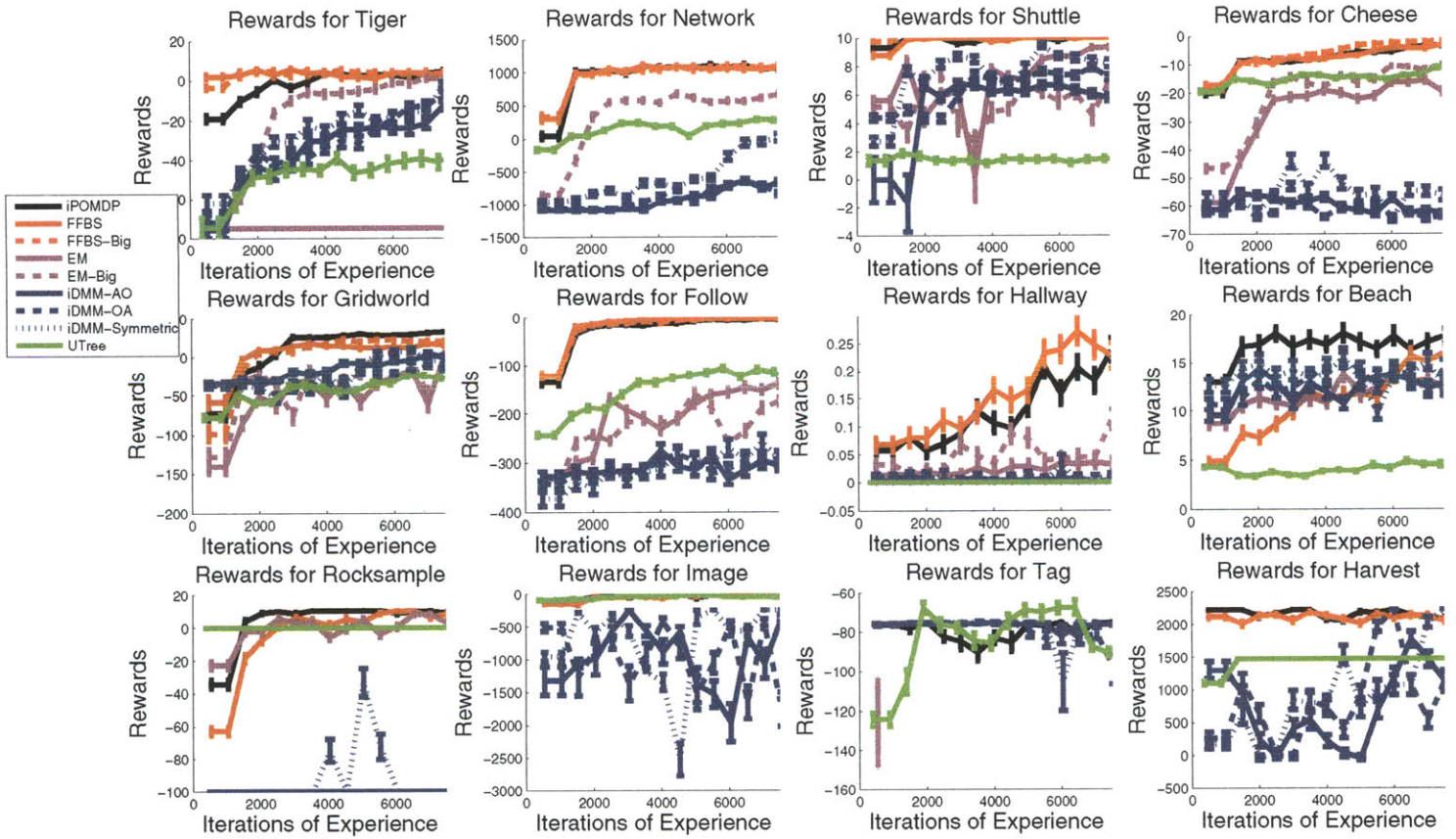


Figure 4-10: Performance of various algorithms on Benchmark Problems. We see that the Bayesian hidden-variable approaches perform best overall, and iPOMDP matches or bests the performance of FFBS with less information about the state space.

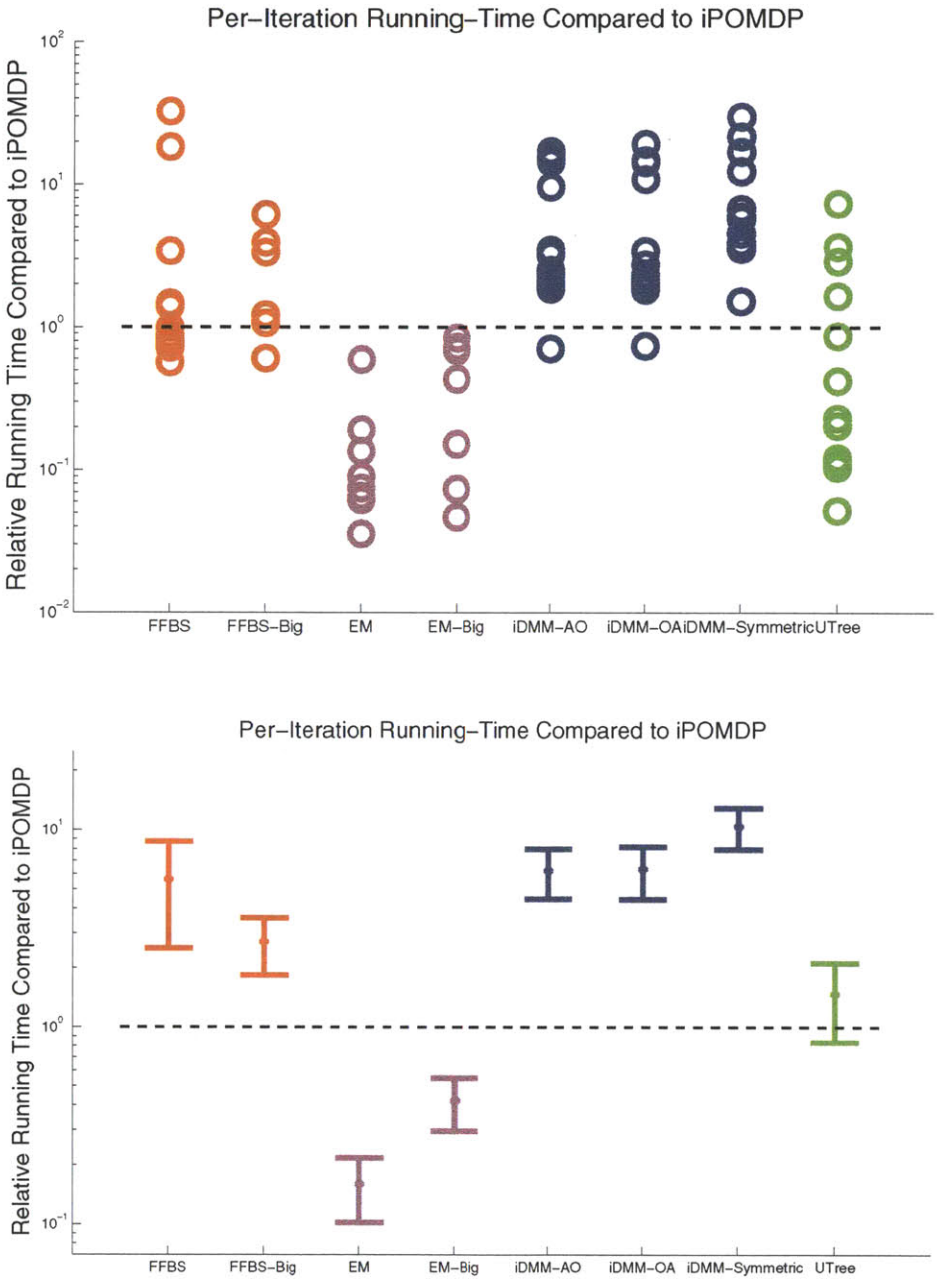


Figure 4-11: Running time of various algorithms on Benchmark Problems. Each circle in the top figure shows the running time of each comparison algorithm compared to the iPOMDP for a particular domain. A value greater than 1 means that the comparison algorithm was slower than the iPOMDP. While simple algorithms, such as EM and U-Tree, generally run faster than iPOMDP, they had significantly worse performance. FFBS-Big appears faster than FFBS because it could only be run on the six smallest domains. The lower figure shows mean relative run-times with standard errors.

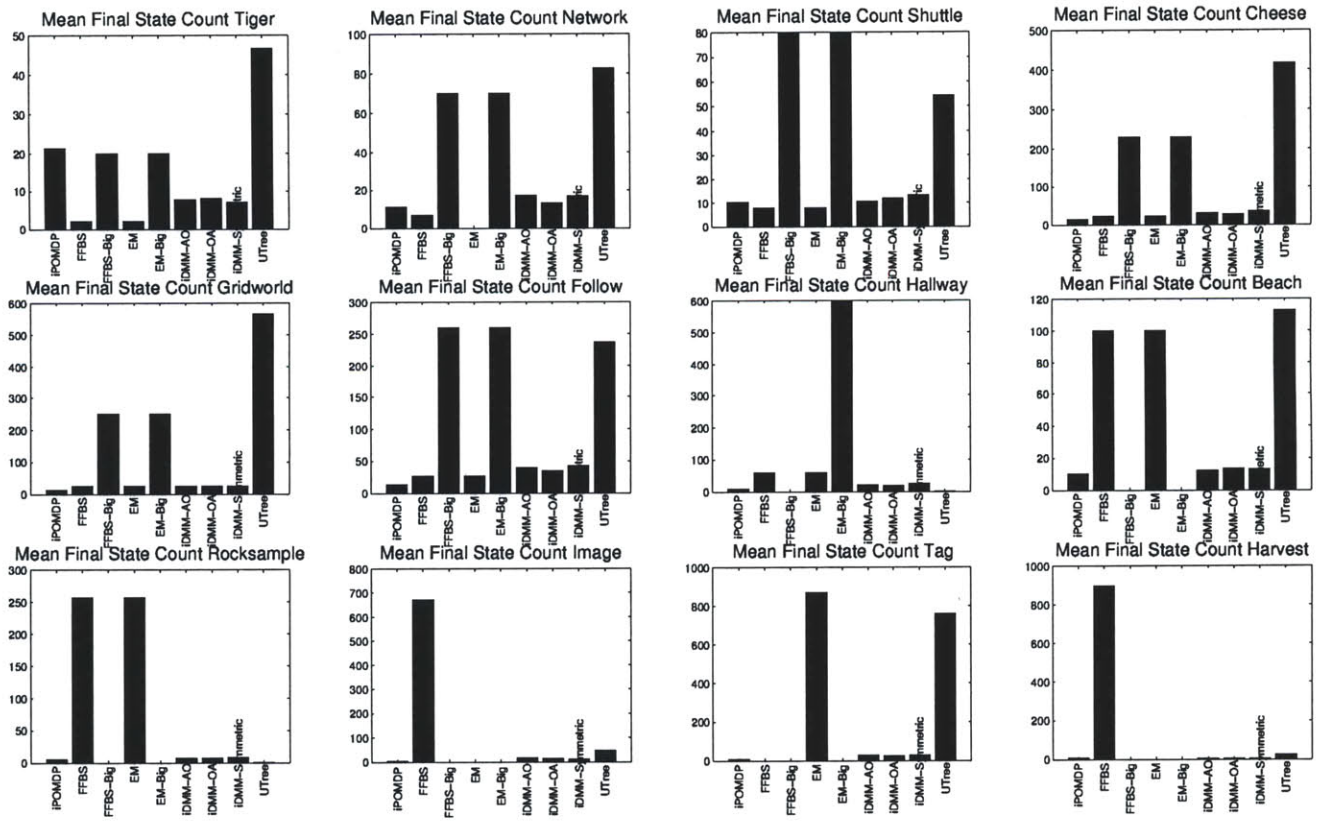


Figure 4-12: Number of variables inferred by various algorithms on Benchmark Problems. The nonparametric approaches, iPOMDP and iDMM, often infer smaller structures to explain the data than the true number of hidden variables (EM/FFBS), resulting in the faster running times seen in figure 4-11.

Another advantage of inferring the size of the state space instead of being conservative is computational. In figure 4-10, the reason that certain algorithms do not appear in later plots is that we were unable to run them on Matlab with 3-GB of RAM. Figure 4-11 shows the running times of all the comparison algorithms relative to the iPOMDP. Because the iPOMDP often infers a smaller number of hidden states than the true count (figure 4-12), ignoring distinctions not supported by the agent's experience, it has faster running times than similar-performing (and predicting) algorithms such as FFBS.

4.4.3 Other action-selection approaches

In chapter 1, we divided the problem of reinforcement learning into two parts: choosing (and learning) a representation, and then selecting actions based on this representation. In section 4.4.2, we saw how the learning mechanism—sampling vs. greedy optimization—can have a large impact on an agent's performance even when the choice of representation is the same. Here, we examine the impact of the choice of action-selection schemes on the performance of the three of three learning algorithms: EM, FFBS, and iPOMDP.

This empirical analysis is motivated in part because stochastic forward search, while guaranteed to be Bayes-optimal in the limit, requires a significant amount of computation to expand the search tree—even to short depths such as five or six. However, while one may be able to plan reasonably well in many *domains* by only considering five actions into the future, it is unclear to what extent the benefits of learning a model can be ascertained by considering such a small amount of future data. In this section, we test several other action-selection heuristics on two domains, tiger and gridworld:

- **Epsilon-Greedy** One of the simplest action-selection algorithms, the QMDP heuristic of Littman et al. [1995] just takes the action with the highest expected reward $\sum_m Q_m(a)b(m)$. This algorithm ignores the future value of information that might be gained from any exploratory action, such as actions that might

help the agent distinguish between models. The epsilon-greedy approach executes the QMDP solution $1 - \epsilon$ proportion of the time, and performs a random action otherwise. We set $\epsilon = .1$ in our experiments.

- **Softmax** Each model maintains an action-value function $Q_m(a)$. Instead of taking the action which maximizes $\sum_m Q_m(a)b(m)$, the softmax algorithms takes action a in proportion to expected future reward: $P(a) \propto \exp(\lambda \sum_m Q_m(a)b(m))$. We can think of softmax as a "softened" version of epsilon-greedy that, instead of occasionally taking a uniformly random action, takes actions based on their expected value.

- **Weighted Stochastic** Weighted stochastic is another very simple but fast heuristic: when selecting actions, we first choose a model m according to its weight $b(m)$. Then we take the action a that m believes is optimal. Choosing a model based on its weight, rather than the most likely model, allows for "potentially useful" actions to be tried some of the time. However, this approach also does not take into account the value of future information.

- **Bayesian Exploration Bonus (BEB)** The original BEB algorithm [Kolter and Ng, 2009], designed for MDPs, inflates rewards for state-action pairs that the agent has rarely tried (hence "exploration bonus"). In the POMDP setting, we no longer have access to the world-state. Instead, we apply the same bonus to state-action visit counts based that the agent *thinks* it has visited. By encouraging the agent to visit regions that it thinks it knows less well, we hope that BEB would help the agent discover more quickly how world-states should be organized.

- **Best of Sampled Set (BOSS)** The original BOSS algorithm [Asmuth et al., 2009] was also designed for MDPs. Given a set of potential models, it uses the most optimistic—or the best of the sample—in each iteration of the iterative process for computing the value of an action. As with BEB, this optimism encourages the agent to either get high rewards or quickly discover that its

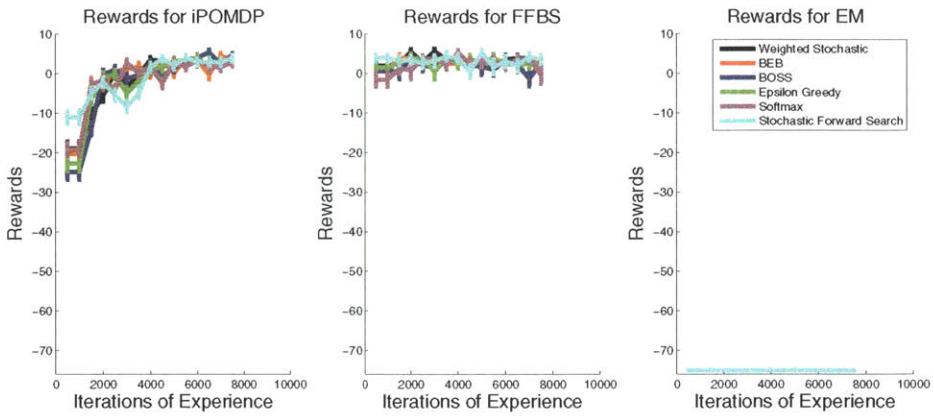


Figure 4-13: Action Selection Comparison on Tiger.

models are too optimistic. While the original paper describes the work in the context of value iteration, which cannot be applied POMDPs, we find it straightforward to implement the concept in the context of a forward-search.

Of the three alternatives to the basic forward search, epsilon-greedy, softmax, and weighted stochastic do not explicitly consider the value of future information. Instead, they ensure that the agent explores enough by introducing various kinds of stochasticity into the action-selection process. In contrast, the BOSS and BEB expand a similar search tree as the standard forward stochastic search, but they use different optimism-under-uncertainty heuristics to encourage exploration; in principle these heuristics should help compensate for lower search depths.

Figures 4-13 and 4-14 compare the different action-selection strategies on two problems, tiger and gridworld, and three learning approaches, iPOMDP, FFBS, and EM. In all the plots, we see that the action-selection approach seems to make relatively little difference in performance; the very simple, fast heuristics (epsilon-greedy, softmax, weighted stochastic) often do quite well compared to the more computationally intensive approaches basic forward search, BEB, and BOSS. In the tiger problem, we confirm that the poor performance of EM was not just due to our stochastic forward search; none of the action-selection strategies can prevent EM from getting caught in local optima.

We have several hypotheses for why the action-selection strategy seems to have

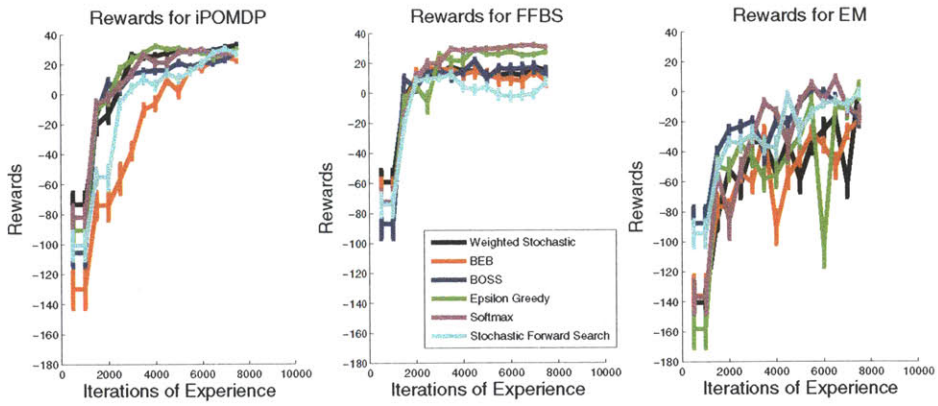


Figure 4-14: Action Selection Comparison on Gridworld. All of the action-selection strategies have similar performance; the main differences in performance come from the inference techniques.

little impact. First, we note that all of the strategies above are only trying to make decisions in the face of model uncertainty $b(m)$. For each sampled model m, we can and do maintain the belief $b(s|m)$ in closed form and compute the value of each action $Q_m(b(s|m), a)$. Thus, if the model uncertainty is small compared to the state uncertainty, it is possible that the models are almost always in agreement; in this situation, all of the action-selection strategies will yield similar results. Looking through the results, we find that all the action-selection strategies chose the greedy action 70-80% of the time, suggesting that the models were often in agreement. If all models m from $b(m)$ are similar, then there is little value in trying to differentiate them.

The model uncertainty may appear to be small either because the posterior $b(m)$ is truly peaked or because of artifacts in the MCMC sampler. We know that Dirichlet priors are fairly smooth, and, in practice, our samplers often give us similar models (because we can rarely mix between modes). We experimented with using multiple restarts to help alleviate potential mixing issues, still finding that we often end up with models that make very similar predictions. Thus, it seems plausible that part of the explanation, at least, may be that there may not be enough information to be discovered in our crude forward search. Models take on the order of hundreds of interactions with the environment to learn; we are only searching to depth of four

or five future actions. It may be that a deeper forward search would result in more optimal learning, but it also seems that the depth-scales for such a forward search may be too large to be practical.

Finally, we might ask if the forward search is not really exploring, how is the learning happening? We hypothesize that all of the domains are relatively "friendly" in the sense that many policies will provide enough information about the domain for the agent to learn a model without explicitly selecting actions to do so. Even action-selection strategies that do not have an explicit stochastic component, such as BOSS or BEB, still rely on sampled models which introduce some randomness into the policy. This randomness, combined with spaces where just acting may provide much information, may be sufficient for simple techniques to perform well.[9]

4.5 Discussion

Recent work in learning POMDP models include Poupart and Vlassis [2008], which uses a set of Gaussian approximations to allow for analytic value function updates in the POMDP space, and Ross et al. [2008a], which jointly reasons over the space of Dirichlet parameter and states when planning in discrete POMDPs. Sampling-based approaches include Medusa [Jaulmes et al., 2005], which learns using state-queries, and Doshi et al. [2008], which learns using policy queries. All of these approaches assume that the number of underlying states is known; all but Doshi et al. [2008] focus on learning only the transition and observation models.

In contrast, the iPOMDP provides a principled framework for an agent to posit more complex models of its world as it gains more experience. By linking the complexity of the model to the agent's experience, the agent is not forced to consider large uncertainties—which can be computationally prohibitive—near the beginning of the planning process, but it still can come up with accurate models of the world when it requires them. As seen in the results, the iPOMDP allows the complexity

[9]We thank David Hsu for an insightful discussion on the topic of action-selection in Bayesian reinforcement learning.

of the model to scale gracefully with the agent's experience. We also showed that it outperforms both basic history-based approaches, such as U-Tree, and our own PDIA alternative.

Past work has attempted to take advantage of structure in POMDPs [Robert et al., 1999, Wolfe, 2006], but learning that structure has remained an open problem. By giving the agent an unbounded state space—but strong locality priors—the iPOMDP provides one principled framework to learning POMDP structure. The Bayesian non-parametric framework also provides some natural directions for extensions: for example, the HDP-based construction described in section 4.1 can be extended to include deeper hierarchies, which can be used to encode structure in the state transitions (for example, clusters of states might behave similarly). In chapter 6, we explore another form of structure, in which the hidden variable representing the world-state is encoded as a discrete vector rather than a single scalar variable.

Another extension would be to use other hierarchical priors than the HDP. In section 4.4.3, we hypothesized that one reason why the choice of action-selection strategies had little effect was that the posterior $b(m)$ was too smooth, making it difficult to discover the value of differentiating models. Posteriors that were peaked around several different candidate hypotheses might make it easier to quickly explore the space of possible models. As an intermediate step toward this goal, we briefly experimented with using hierarchical Pitman-Yor (HPY) processes [Teh, 2006, Pitman and Yor, 1997] as in the iDMM as the transition prior for the iPOMDP. These processes should have introduced a slight bias toward sparser transitions—perhaps one likely next-state for each transition and several unlikely next-states, which we hoped would lead to more differentiable models. In our tests, the performance of the HPY-based iPOMDP did not differ significantly from the iPOMDP (and it was sometimes worse), but we still believe that the HPY or other priors may provide an alternative if we have strong reason to suspect that the environment has a different transition structure than the HDP.

Chapter 5

Nonparametric Policy Priors[1]

In chapter 4, we described how an agent might adaptively build a model from histories of its own experience. This form of data, which we term 'self-exploration,' is the most common form of data used in model-based RL algorithms [Jaulmes et al., 2005, Ross et al., 2008a,b, Doshi et al., 2008] to learn the world's dynamics. However, sometimes other sources of histories are available: for example, a different setting is one in which the the agent has access to histories collected from experts [Abbeel et al., 2006, Ratliff et al., 2009]. In the first setting, self-exploration, the agent's choice of actions do not provide information about the model m; actions are useful only in that they can reveal patterns in the transitions, observations, and rewards. In the second setting, in which histories come from experts, the action choices themselves indirectly reveal additional information about the model by demonstrating a near-optimal policy.

In this chapter, we consider a model for learning when we have access to both types of histories: data from the agent's own self-exploration and data from experts. Data from self-exploration gives information about the dynamics, while expert demonstrations show outputs of good policies and thus provide indirect information about the underlying model. Similarly, rewards observed during independent exploration provide indirect information about good policies. Because dynamics and policies are linked through a complex, nonlinear function, leveraging information about both these

[1]**This work was joint with David Wingate**

aspects at once is challenging. Using both kinds of data improves model-building and control performance.

As usual, we use a Bayesian reinforcement learning approach to this problem, applying Bayes rule to write a posterior over models m given data D as $p(m|D) \propto p(D|m)p(m)$. Previous works [Poupart and Vlassis, 2008, Strens, 2000, Asmuth et al., 2009, Dearden et al., 1999] have generally defined the model prior $p(m)$ as a distribution directly on the dynamics and rewards models, making it difficult to incorporate the policy information encoded in expert trajectories. Placing a prior $p(\pi)$ over what the expert's policy might be, such as in Wilson et al. [2010], is another option, but it ignores the direct information that the agent observes about the environment while also observing the expert's trajectory.

A third option might be to place a prior over both the model parameters $p(m)$ and a prior over the optimal policy $p(\pi)$. However, the interpretation of these disjoint priors is somewhat strange: for example, suppose that the model is known: then we know that there is just one optimal policy. A policy prior that acts as a smoother on this optimal policy will result in less than optimal performance. One of the main contributions of this chapter is to define joint priors over models and policies that make it easier to incorporate information from trajectories in a principled, Bayesian way. In particular, we show how our model can be interpreted as a joint prior that introduces a bias for world models which are both simple and easy to control. This flexibility also allows to consider cases where the expert trajectories may not be completely optimal in a robust manner.

In this chapter, we first lay out a very general approach to modeling and inference over joint model-policy priors in sections 5.1 and 5.2. In section 5.3, we describe the model and the inference for one very specific joint prior that we also use for the empirical tests in section 5.4. Focusing on discrete environments, we use the iPOMDP from chapter 4 as our prior over models $p(m)$. We model policies using (infinite) state controllers. Noting that the graphical model for a state controller is identical to the graphical model for a POMDP, with the roles of the observations and actions reversed, we use the iPOMDP as a prior over policies $p(\pi)$ as well. Because

the iPOMDP has a bias toward models with smaller numbers of instantiated states, this joint prior places a bias toward world models m that explain the agent's self exploration data with relatively few states and whose optimal policy, as characterized by the expert's demonstrations, also requires relatively few nodes.

5.1 Model

We first lay out the notation for our joint model-policy prior. Let m denote the (unknown) world model. For example, if m is the class of POMDPs, then m consists of the transitions T, observations Ω, and rewards R. Let π_{e} be the expert's policy; together with the model m, the expert produces a set of histories $h = a_1 o_1 r_1 ... a_t o_t r_t$ collectively called the expert data D_{e}. Similarly, the agent executes some policy π_{a} to produce self-exploration or agent data D_{a}. The agent has access to all histories, but the world model m and optimal policy π_{e} are hidden.

In general, we can assume that the expert's demonstration provides useful information beyond the robot's self-exploration because the expert knows the world model m, and thus can execute a near-optimal policy. Figure 5-1 shows the two graphical models that can explain the expert's actions. Both graphical models assume that a particular world m is sampled from a prior over POMDPs, $g_m(m)$. In what would be the standard application of Bayesian RL with expert data (figure 5-1(a)), the prior $g_m(m)$ fully encapsulates our initial belief over world models. An expert, who knows the true world model m, executes a planning algorithm $\texttt{plan}(m)$ to construct an optimal policy π_{e}. The expert then executes the policy to generate expert data D_{e}, distributed according to $p(D_{\mathrm{e}}|m, \pi_{\mathrm{e}})$, where $\pi_{\mathrm{e}} = \texttt{plan}(m)$.

However, the graphical model in figure 5-1(a) does not easily allow us to encode a prior bias toward more controllable world models. In figure 5-1(b), we introduce a new graphical model in which we allow additional parameters in the distribution $p(\pi_{\mathrm{e}})$. In particular, we introduce an additional factor $g_\pi(\pi_{\mathrm{e}})$ that depends only on

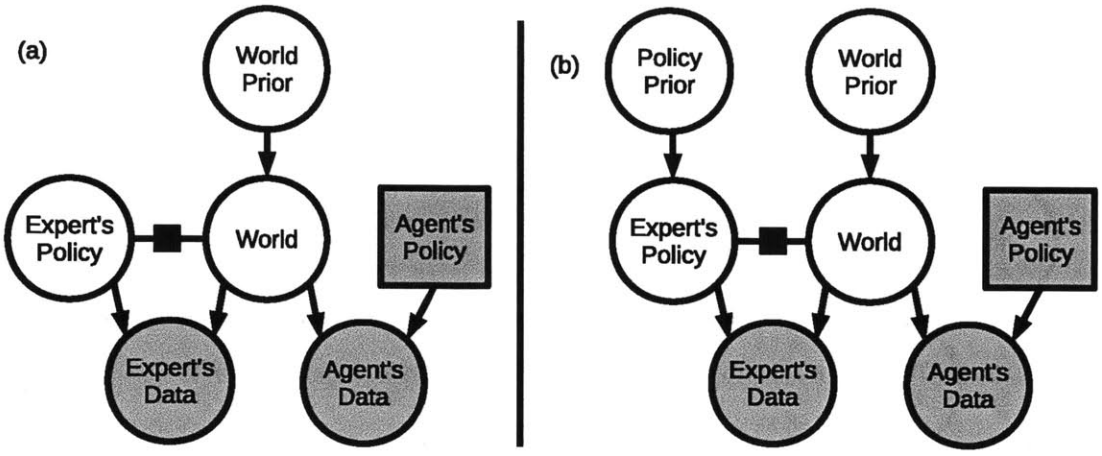


Figure 5-1: Two graphical models of expert data generation. Left: the prior only addresses world dynamics and rewards. Right: the prior addresses both world dynamics and controllable policies. The link between the world and the expert policy can be used to encode relationships such as the expert being optimal or near-optimal.

the properties of the policy:

$$p(\pi_e|m) \propto f_m(\pi_e)g_\pi(\pi_e) \tag{5.1}$$

The factor $g_\pi(\pi_e)$ can be interpreted as the "policy prior." As in the simpler graphical model in figure 5-1(a), the factor $f_m(\pi_e)$ should be a measure how well the expert policy π_e matches the world model m. We can write the distribution over world models as

$$p(m) \propto \int_{\pi_e} f_m(\pi_e)g_\pi(\pi_e)g_m(m)d\pi_e \tag{5.2}$$

If $f_m(\pi_e)$ is a delta function on $\texttt{plan}(m)$, then the integral in equation 5.2 reduces to

$$p(m) \propto g_\pi(\pi_e^m)g_m(m) \tag{5.3}$$

where $\pi_e^m = \texttt{plan}(m)$, and we see that we have a prior over models that provides input on both the world's dynamics and the world's controllability. For example, if the policy class is the set of finite state controllers (as in section 5.3), the policy prior $g_\pi(\pi_e)$ might encode preferences for a smaller number of nodes used the policy, while

$g_m(m)$ might encode preferences for a smaller number of visited states in the world. The function $f_m(\pi_e)$ can also be made more general to encode how likely it is that the expert uses the policy π_e given world model m.

Finally, we note that $p(D_e|m,\pi)$ factors as $p(D_e^a|\pi)p(D_e^{o,r}|m)$, where D_e^a are the actions in the histories D_e and $D_e^{o,r}$ are the observations and rewards. Therefore, the conditional distribution over world models given data D_e and D_a is:

$$p(m|D_e, D_a) \propto p(D_e^{o,r}, D_a|m)g_m(m) \int_{\pi_e} p(D_e^a|\pi_e)g_\pi(\pi_e)f_m(\pi_e)d\pi_e \tag{5.4}$$

The model in figure 5-1(a) corresponds to setting a uniform prior on $g_\pi(\pi_e)$. Similarly, the conditional distribution over policies given data D_e and D_a is

$$p(\pi_e|D_e, D_a) \propto g_\pi(\pi_e)p(D_e^a|\pi_e) \int_m f_m(\pi_e)p(D_e^{o,r}, D_a|m)g_m(m)dm \tag{5.5}$$

In this section, we have described a very general formulation that shows how to encode our prior biases about both the model and the policy into a unified, principled framework. There exist some mild conditions on these independently-specified world and policy priors for the joint prior to be consistent: we require that there exist models for which both the policy prior and model prior are non-zero in the limit of data. Equivalently, it is sufficient for the true policy and the true model to be contained in their respective hypothesis spaces. In the limit of infinite data, the model posterior will peak to a set of dynamics models consistent with the data, while the policy posterior will peak to a set of policies consistent with the expert. As long as the expert was providing optimal trajectories, we can argue that there will exist at least one model and one policy in these sets for which the policy is the solution of the model.

5.2 Methods

As in chapter 4, we describe the belief monitoring and action selection phases for when working with nonparametric policy priors. The belief monitoring phase involves doing

inference in the joint space of models and policies. The action selection phase involves choosing an action given the posterior over models and policies.

5.2.1 Belief Monitoring

The belief now consists of four hidden elements: the model m, the expert policy π_e, the current state of the world s, and the current state of the agent's policy n. For now, we do not assume any specific properties about the state of the policy n or the state of world s; these are simply variables that encode any dynamic aspects of the model or the policy. We assume that the model m itself and the expert's policy π_e do not change over time. Similar to the approach in chapter 4, we write the belief $b(s, n, m, \pi_e)$ as $b(s, n|m, \pi_e)b(m, \pi_e)$. We assume that given the model m and policy π_e, the belief over model and policy states s and n, $b(s, n|m, \pi_e)$, can be computed using a standard approach such as equation 2.1. The focus of this section is computing and updating the joint model-policy belief $b(m, \pi_e)$.

We describe three inference approaches for incorporating knowledge from expert trajectories when sampling from the belief $b(m, \pi_e)$. The samples produced in all of the inference approaches below are unbiased as long as unbiased estimators are used for sampling from $p(\pi_e|D_e)$ and $p(m|D_a, D_e)$. The first approach assumes that the policy prior $p(\pi_e)$ is uniform, corresponding to the case in figure 5-1(a). The second two approaches allow for non-uniform policy priors (corresponding to figure 5-1(b)). The first samples only models m directly, while the second of the two approaches samples both models m and expert policies π_e explicitly.

Uniform Policy Priors (Bayesian RL with Expert Data).

If $f_m(\pi_e) = \delta(\texttt{plan}(m))$ and we believe that all policies are equally likely (graphical model 5-1(a)), then we can leverage the expert's data by simply considering how well that world model's policy $\texttt{plan}(m)$ matches the expert's actions for a particular world model m. Equation 5.4 allows us to compute a posterior over world models that accounts for the quality of this match. We can then use that posterior as part of

a planner by using it to evaluate candidate actions. The expected value of an action[2] $q(a)$ with respect to this posterior is given by:

$$\begin{aligned}\mathbb{E}\left[q(a)\right] &= \int_m q(a|m)p(m|D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}})dm \\ &= \int_m q(a|m)p(D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}|m)g_m(m)p(D_{\mathrm{e}}^{a}|\mathtt{plan}(m))dm \qquad (5.6)\end{aligned}$$

We assume that we can draw samples from $p(m|D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}) \propto p(D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}|m)g_m(m)$. We then weight those samples by $p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})$, where $\pi_{\mathrm{e}} = \mathtt{plan}(m)$, to yield the importance-weighted estimator

$$\mathbb{E}\left[q(a)\right] \approx \sum_i q(a|m_i)p(D_{\mathrm{e}}^{a}|m_i, \pi_{\mathrm{e}}), \quad m_i \sim p(m|D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}).$$

Finally, we can also sample values for $q(a)$ by first sampling a world model given the importance-weighted distribution above and recording the $q(a)$ value associated with that model. (See section 2.2.2 for a review of importance sampling.)

Policy Priors with Model-based Inference.

The uniform policy prior implied by standard Bayesian reinforcement learning does not allow us to encode prior biases about the policy. If we apply a more general prior (graphical model 5-1(b) in figure 5-1), then the expectation in equation 5.6 becomes

$$\mathbb{E}\left[q(a)\right] = \int_m q(a|m)p(D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}|m)g_m(m)g_\pi(\mathtt{plan}(m))p(D_{\mathrm{e}}^{a}|\mathtt{plan}(m))dm \qquad (5.7)$$

if we still assume that the expert uses an optimal policy, that is, $f_m(\pi_{\mathrm{e}}) = \delta(\mathtt{plan}(m))$.

Using equation 5.7 can result in somewhat brittle and computationally intensive inference, however, as we must compute π_{e} for each sampled world model m. It also assumes that the expert used the optimal policy, whereas a more realistic assumption might be that the expert uses a near-optimal policy. We now describe an alternative

[2]We omit the belief over world states $b(s)$ from the equations that follow for clarity; all references to $q(a|m)$ are $q(a|b_m(s), m)$.

link function $f_m(\pi_e)$ that relaxes the hard constraint that the expert produces optimal trajectories $f_m(\pi_e) = \delta(\texttt{plan}(m))$. Instead, we let $f_m(\pi_e)$ be a function that prefers policies that achieve higher rewards in world model m:

$$f_m(\pi_e) \propto \exp\{V(\pi_e|m)\} \tag{5.8}$$

where $V(\pi_e|m)$ is the value of the policy π_e on world m, encoding a bias stating that experts tend to use policies that yield high value.

Substituting this new link function $f_m(\pi_e)$ into equation 5.4, the expected value of an action is

$$\mathbb{E}[q(a)] = \int_{m,\pi_e} q(a|m)p(D_e^a|\pi_e)\exp\{V(\pi_e|m)\}\, g_\pi(\pi_e)p(D_e^{o,r}, Da|m)g_m(m)dmd\pi_e$$

We again assume that we can draw samples from $p(m|D_e^{o,r}, D_a) \propto p(D_e^{o,r}, D_a|m)g_m(m)$, and additionally assume that we can draw samples from $p(\pi_e|D_e^a) \propto p(D_e^a|\pi_e)g_\pi(\pi_e)$, yielding:

$$\mathbb{E}[q(a)] \approx \sum_i q(a|m_i) \sum_j \exp\left\{V(\pi_{ej}|m_i)\right\}, \quad m_i \sim p(m|D_e^{o,r}, D_a), \pi_{ej} \sim p(\pi_e|D_e^a) \tag{5.9}$$

As in the case with standard Bayesian RL, we can also use our weighted world models to draw samples from $q(a)$.

Policy Priors with Joint Model-Policy Inference.

While the model-based inference for policy priors in section 5.2.1 is correct, using importance weights often suffers when the proposal distribution is not near the true posterior. For example, when using the uniform priors as in section 5.2.1, it may be that none of the models sampled just from using the dynamics have optimal policies that are close to that of the expert—and thus they will all look equally (un)plausible. The same problem occurs if we sample sets of policies as in section 5.2.1 and count the number of cases for which a policy is optimal: instead, we are forced to use a very

specific link function that scores a policy based on how well it performs rather than optimality.

In this section, we present a blocked sampler that alternates between resampling the policy based on the model and expert data and then resampling the model based on the policy and the dynamics data (from the expert and the agent's self exploration). Once we have a set of sampled models we can compute the expectation $\mathbb{E}[q(a)]$ simply as the average over the action values $q(a|m_i)$ for each sampled model.

Sampling the Policy given the Model Given a world model, equation 5.5 becomes

$$p(\pi_{\mathrm{e}}|D_{\mathrm{e}}, D_{\mathrm{a}}, m) \propto g_{\pi}(\pi_{\mathrm{e}})p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})f_{m}(\pi_{\mathrm{e}}) \tag{5.10}$$

where making $g_{\pi}(\pi_{\mathrm{e}})$ and $p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})$ conjugate is generally an easy design choice. We then approximate $f_{m}(\pi_{\mathrm{e}}) \approx q(\pi_{\mathrm{e}}, b)$ with a function $q(\pi_{\mathrm{e}}, b)$ in the same conjugate family as $g_{\pi}(\pi_{\mathrm{e}})$, where $q(\pi_{\mathrm{e}}, b) \to f_{m}(\pi_{\mathrm{e}})$ as $b \to 0$. We can then interpret b as a cooling parameter that can be used to initially provide the sampler with more flexibility but does not affect the asymptotic correctness of the inference. The details of how to construct $q(\pi_{\mathrm{e}}, b)$ will depend on the specific choices of priors $g_{\pi}(\pi_{\mathrm{e}})$ and likelihoods $p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})$, and we give an example of one such case in section 5.3.

Sampling the Model given the Policy Next we sample a new world model m given the policy π. Given a policy, equation 5.4 reduces to

$$p(m|D_{\mathrm{e}}, D_{\mathrm{a}}, \pi_{\mathrm{e}}) \propto p(D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}|m)g_{m}(m)f_{m}(\pi_{\mathrm{e}}). \tag{5.11}$$

We use Metropolis-Hastings (MH) to sample new world models (see section 2.2.2 for a summary of MH). For our proposal distribution, we use the posterior over models ignoring the policy information:

$$q(m'|m) = p(D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}}|m)g_{m}(m). \tag{5.12}$$

Thus the acceptance probability α is simply the ratio

$$\alpha = \frac{f_{m'}(\pi_e)}{f_m(\pi_e)}. \tag{5.13}$$

If $f_m(\pi_e)$ is highly peaked, then this ratio will often be zero, so we apply another annealing scheme to allow the sampler more room to move initially. In our case, we want to eventually approximate $f_m(\pi_e) = \delta(\texttt{plan}(m))$. We use the same smoothed approximation as in section 5.2.1, $\hat{f}_m(\pi_e) \propto \exp(b \cdot (V(\pi_e|m) - V(\pi_e^m|m))^2)$. However, instead of simply letting $f_m = \hat{f}_m$, we slowly increase the inverse temperature parameter b. As a result, we eventually only accept a model m' if it produces higher rewards under policy π_e than the current model m. Over time, because the proposal distribution $q(m'|m)$ has full support over the space of models, we will sample models for which π_e is near-optimal.

Using MH can suffer from the same issues as the importance sampling-based approaches described in sections 5.2.1 and 5.2.1: a poor choice of proposal $q(m'|m)$ can result in very few acceptances because the new model m' must perform better than the current model m under the expert policy π_e. However, the annealing provided by the inverse temperature parameter b allows for smoothing early on, letting the sampler explore the space of models without getting caught in local optima. More generally, annealing in both the policy and model inference ensures that one does not quickly become fixed and force the other to simply adjust to match the link constraint $f_m(\pi_e)$.

5.2.2 Action Selection

All the approaches in Sec. 5.2.1 output samples of models $\{m\}$. Since choosing the Bayes-optimal action is intractable, we follow the same process as in chapter 4 and first solve all of the models (each of which is typically small) using standard POMDP planners such as Pineau et al. [2003]. During the testing phase, the internal belief state $b(s|m)$ of the models is updated after each action-observation pair. As in chapter 4, models are also reweighted using standard importance weights so that they continue

to be an unbiased approximation of the true belief $b(m)$

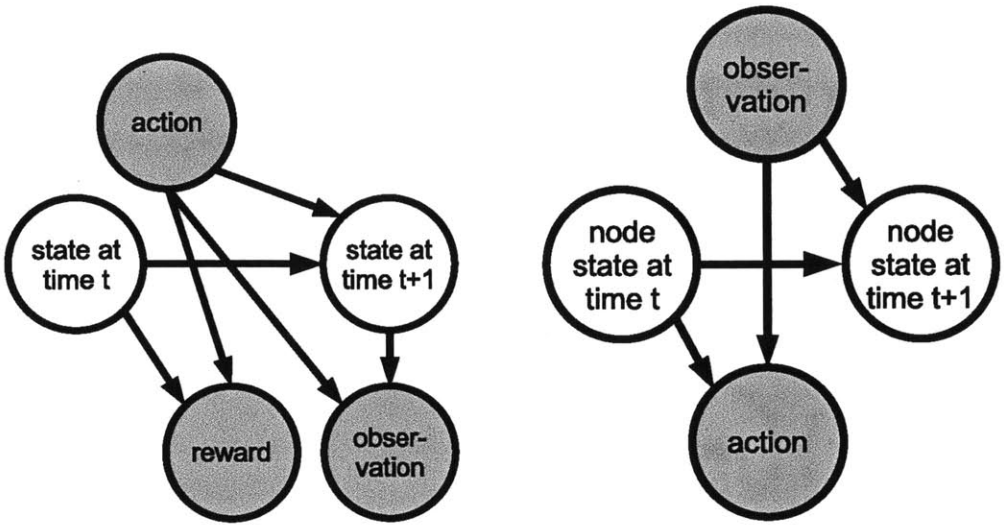
In chapter 4, we found that more sophisticated action-selection strategies, such as stochastic forward search, did not significantly outperform simpler approaches such as just sampling a model and selecting its action. Thus, for the action-selection in this chapter, we simply select a model based on its probability $b(m)$ and then take the action preferred by that model. (This is one of the fastest strategies for action selection.)

5.3 Example: Nonparametric Policy Priors Using the iPOMDP

In this section, we describe one very specific model prior $p(m)$, the iPOMDP, and also describe how it can be used for a policy prior $p(\pi)$ if the policies are encoded as state controllers. We assume that the expert is optimal, that is, the link function $f_m(\pi_e) = \delta(\texttt{plan}(m))$, and provide details valuable for implementing the joint model-policy sampling approach of section 5.2.1 with this choice of priors.

5.3.1 Model

We use the iPOMDP as the prior $p(m)$ over world models m. To place a prior over the policies $p(\pi)$, we first describe how policies can be represented with finite state controllers [Sondik, 1971]. Instead of mapping directly from belief states $b(s, m)$ to actions a, a finite state controller introduces the concept of a node-state n. Each node state is associated with a policy function $\pi(a|n)$ that gives the probability that the agent will perform action a in state n. The controller also has a transition function $\beta(n'|n, o)$ that gives the probability of transitioning to node-state n' if the environment provides an observation o in node-state n. We can imagine that many different histories $h = a_1 o_1 r_1 ... a_t o_t r_t$ will result in reaching a similar set of node-states n; in this way, the node state compresses sets of histories together. All the histories that end up in a particular node-state n will have the same action-selection strategy



(a) Single POMDP time slice

(b) Single FSC time slice

Figure 5-2: Graphical Model for the Finite State Controller.

$\pi(a|n)$.

Formally, a finite state controller consists of the n-tuple (N,A,O,π,β). The sets N, A, and O are sets of nodes, actions, and observations. The set of nodes is discrete and finite, and the node transition function $\beta(n'|n,o)$ and the policy function $\pi(a|n)$ are as described above. Written this in this way, the similarity between the POMDP and the finite state controller is clear: the roles of the observations and actions have simply been reversed. The iPOMDP prior now posits that there is a stochastic state controller with an unbounded number of nodes n, but probable polices use only a small subset of the nodes a majority of the time. We perform joint inference over the node transition parameters $\beta(n'|n,o)$ and the policy parameters $\pi(a|n)$ which are equivalent to the state transition parameters $T(s'|s,a)$ and the observation parameters $\Omega(o|s,a)$. Figure 5-2 shows the graphical model for the finite state controller alongside the graphical model for the POMDP.

Used as a policy prior, we can think of the iPOMDP as a stochastic state controller treating the observations as inputs and the actions as outputs. The iPOMDP does not restrict the number of nodes in the state controller, but it does places a bias toward state controllers that can explain the expert's choices with only a few nodes. The

'policy state' representation n learned is not the world state, rather it is a summary of previous observations which is sufficient to predict actions. Assuming that the training action sequences are drawn from the optimal policy, the learner will learn just enough "policy state" to control the system optimally. Finally, we note that the iPOMDP, as both a model and policy prior, has full support over all possible POMDP worlds and all possible state-controller policies. Thus, it is a consistent prior to use.

5.3.2 Methods

We first note that the beam sampler of van Gael et al. [2008] can be used to sample from the partial posteriors $p(m|D_{\mathrm{e}}^{o,r}, D_{\mathrm{a}})$ and $p(\pi_{\mathrm{e}}|D_{\mathrm{e}}^{a})$. In the limit of infinite samples we will recover the true model and policy posteriors conditioned on their respective data $D_{\mathrm{a}}, D_{\mathrm{e}}^{o,r}$ and D_{e}^{a}. When sampling the models and policies using the approach from section 5.2.1, sampling models m requires no specific considerations. Sampling the policies requires formulating $f_m(\pi_{\mathrm{e}})$ as a function $q(\pi_{\mathrm{e}}, b)$ that is conjugate to the policy prior function $g_\pi(\pi_{\mathrm{e}})$. We describe the process for sampling π_{e} below.

Recall from section 5.2.1 that our policy posterior is given by

$$p(\pi_{\mathrm{e}}|D_{\mathrm{e}}, D_{\mathrm{a}}, m) \propto g_\pi(\pi_{\mathrm{e}}) p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}}) f_m(\pi_{\mathrm{e}}) \tag{5.14}$$

where $g_\pi(\pi_{\mathrm{e}})$ and $p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})$ are conjugate if the policy prior $g_\pi(\pi_{\mathrm{e}})$ is the iPOMDP. Specifically, when using the iPOMDP prior, the transition parameters $\beta(n'|n, o)$ and the policy parameters $\pi(a|n)$ are both multinomials. The beam sampler samples these multinomials given the expert data D_{e}^{a} by combining the iPOMDP prior $g_\pi(\pi_{\mathrm{e}})$ with node-visit counts from the likelihood $p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})$.

To make $f_m(\pi_{\mathrm{e}})$ also conjugate to $g_\pi(\pi_{\mathrm{e}})$, we represent f_m as a set of node-visit counts. Then we can simply combine those counts with the node-visit counts from the likelihood $p(D_{\mathrm{e}}^{a}|\pi_{\mathrm{e}})$ and continue to use our Dirichlet-multinomial formulation to sample the parameters $\beta(n'|n, o)$ and $\pi(a|n)$. Our strategy for performing this approximation takes two steps: first computing a near-optimal stochastic state controller for the given model m (the only policy that would have non-zero weight under

$f_m(\pi_e)$), and then adding a bias toward this stochastic state controller to the prior $g_\pi(\pi_e)$ and the likelihood with respect to the expert data the likelihood $p(D_e^a|\pi_e)$.

Suppose that we have a near-optimal stochastic state controller π^* for model m. All of the parameters of π^* are expressed as multinomial distributions. Recall that the input parameter to a Dirichlet distribution can be parameterized as a vector $a\vec{p}$, where the vector $\vec{p}$ is the multinomial that is the mean of the distribution and a is a concentration parameter describing how sharply peaked the distribution is about its mean. Now, for a certain set of transition function parameters $\beta(\cdot|n,o)$, we have: (1) a mean transition distribution β from the prior $g_\pi(\pi_e)$, (2) a set of node-visit counts $c_{n,o}$ from the likelihood with respect to expert data $p(D_e^a|\pi_e)$, and (3) a desired mean multinomial $\beta^*(\cdot|n,o)$ from the near-optimal policy π^*. We sample $\beta(\cdot|n,o) \sim \mathsf{Dirichlet}(\beta + c_{n,o} + a\beta^*(n,o))$.

If we start with a small temperature parameter a, then the link function f_m—and by proxy, the sampled model m—will play a relatively small role when resampling the expert policy π_e; most of the weight will come from the expert trajectories. As a is increased, we will recover the desired $f_m(\pi_e) = \delta(\texttt{plan}(m))$ because the "fake counts" $a\beta^*(n,o))$ will be much larger than the node-visit counts $c_{n,o}$. The initial approximation allows the expert data to guide the inference without sacrificing asymptotic correctness. An identical approach can be used to sample the parameters of the policy function $\pi(\cdot|n)$;

Our method of directly combining the counts from the expert trajectories with the optimized model π^* implies that nodes in π^* must "match" nodes used in the counts. This condition guides our choice for computing π^*: we compute π^* by applying several rounds of bounded policy iteration [Poupart and Boutilier, 2003] to the current policy π_e to get a policy that is near-optimal with respect to model m. Bounded policy iteration is an iterative process that makes local adjustments to make a policy more optimal. Thus, it preserves the meaning of a node as changes are made.

Practically, this approximation technique requires an implementation of bounded policy iteration and only a very small change to the beam sampler. Once the optimal policy has been computed in a separate module, we simply pass in the $ap_{n,o}$ values

for each node transition function and the ap_n values for each policy function into the beam-sampler, to be added to the base measures β or H before each sampling round.

5.4 Experiments

We first describe a pair of demonstrations that show two important properties of using policy priors: (1) that policy priors can be useful even in the absence of expert data when we assume that $f_m(\pi_e) = \delta(\texttt{plan}(m))$, and (2) that our approach works even when the expert trajectories are not optimal. We compared our inference approaches with two approaches that did not leverage the expert data: expectation-maximization (EM) used to learn a finite world model and the infinite POMDP (chapter 4), which placed the same nonparametric prior over world models as we did.

All of the tests were run with the same set of conditions for resampling updates and inference parameters:

- **Trial Length** All the tasks were episodic, but the number of episodes were a poor metric for the amount of experience because episodes could be of varying lengths. Specifically, experts generally completed the task in fewer iterations. Therefore, we allowed each approach $N = 2500$ iterations, or interactions with the world, during each learning trial. Each episode was capped at 50 iterations.

- **Expert Trajectories** The agent was provided with an expert trajectory with probability $.5\frac{n}{N}$, where n was the current amount of experience. No expert trajectories were provided in the last quarter of the iterations. All agents received expert trajectories, regardless of whether they took advantage of trajectory information.

- **Repeated Trials** Each approach was run for 10 trials (of 2500 iterations each).

- **Updating Models and Policies** Models and policies were updated every 100 iterations. For all of the nonparametric approaches, 50 models and/or policies were sampled, 10 iterations apart, after an initial burn-in of 500 iterations. Sampled models were solved using 25 backups of PBVI [Pineau et al., 2003] with 500

sampled beliefs. When using the joint model-policy inference of section 5.2.1, one iteration of bounded policy iteration [Poupart and Boutilier, 2003] was performed per sampled model to move it toward a more optimal model. When training the finite model, we set the number of states $|S| = \min(25, |S_{true}|)$, where $|S_{true}|$ was the true number of underlying states. Both the nonparametric and finite learners were trained from scratch during each update; we found empirically that starting from random points made the learner more robust than starting it at potentially poor local optima.

- **Evaluating Agent Progress** Following each update, we ran 50 test episodes (not included in the agent's experience) with the new models and policies to empirically evaluate the current value of the agents' policy.

Policy Priors with No Expert Data The combined policy and model prior encodes a prior bias towards models with simpler control policies. This interpretation of policy priors can be useful even without expert data: figure 5-3 shows the performance of the policy prior-biased approaches and the standard iPOMDP on a gridworld problem in which observations correspond to both the adjacent walls (relevant for planning) and the color of the square (not relevant for planning). This domain has 26 states, 4 colors, standard NSEW actions, and an 80% chance of a successful action. The optimal policy for this gridworld was simple: go east until the agent hits a wall, then go south. However, the varied observations made the iPOMDP infer many underlying states, none of which it could train well, and these models also confused the joint model-policy inference from section 5.2.1. The uniform prior approach from section 5.2.1 has no bias toward smaller models; without expert data, it cannot outperform the iPOMDP. By biasing the agent towards worlds that admit simpler policies, the model-based inference with policy priors from section 5.2.1 creates a faster learner.

Policy Priors with Imperfect Experts While we focused on optimal expert data, in practice policy priors can be applied even if the expert is imperfect. Fig-

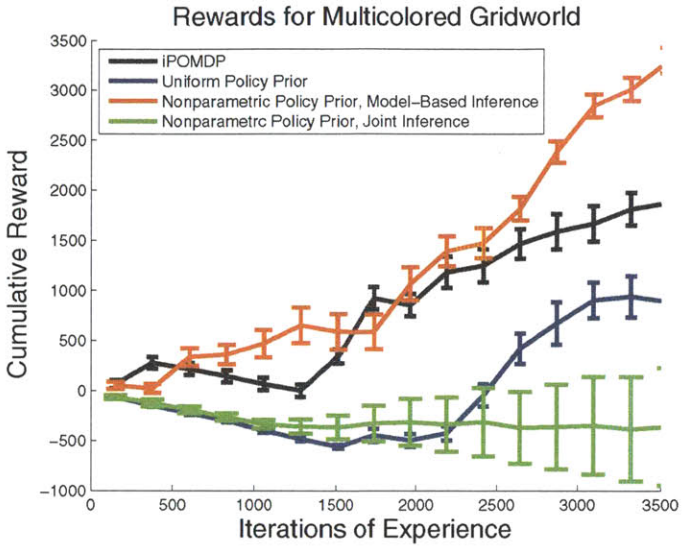


Figure 5-3: Learning curves for different agents on a multicolored gridworld where the colors of the locations were irrelevant.

ure 5-4(a) shows learning curves for a simulated snake manipulation problem with a 40-dimensional continuous state space, corresponding to (x,y) positions and velocities of 10 body segments. Actions are 9-dimensional continuous vectors, corresponding to desired joint angles between segments. The snake is rewarded based on the distance it travels along a twisty linear "maze," encouraging it to wiggle forward and turn corners.

We generated expert data with a rapidly-exploring random tree (RRT), a trajectory-based planner. The RRT is very effective for finding feasible solutions to high-dimensional problems, but it does not provide optimal trajectories. We derived 20 motor primitives for the action space using a clustering technique and interpreted the sequence of clustered actions taken by the RRT as our expert data. Although the trajectories and primitives are suboptimal, figure 5-4(a) shows that knowledge of feasible solutions boosts performance when using the policy-based technique.

Tests on Standard Problems We also tested the approaches on ten standard benchmark POMDP problems: tiger ([Littman et al., 1995], 2 states), network ([Littman et al., 1995], 7 states), shuttle ([Chrisman, 1992], 8 states), a 5x5 version of gridworld

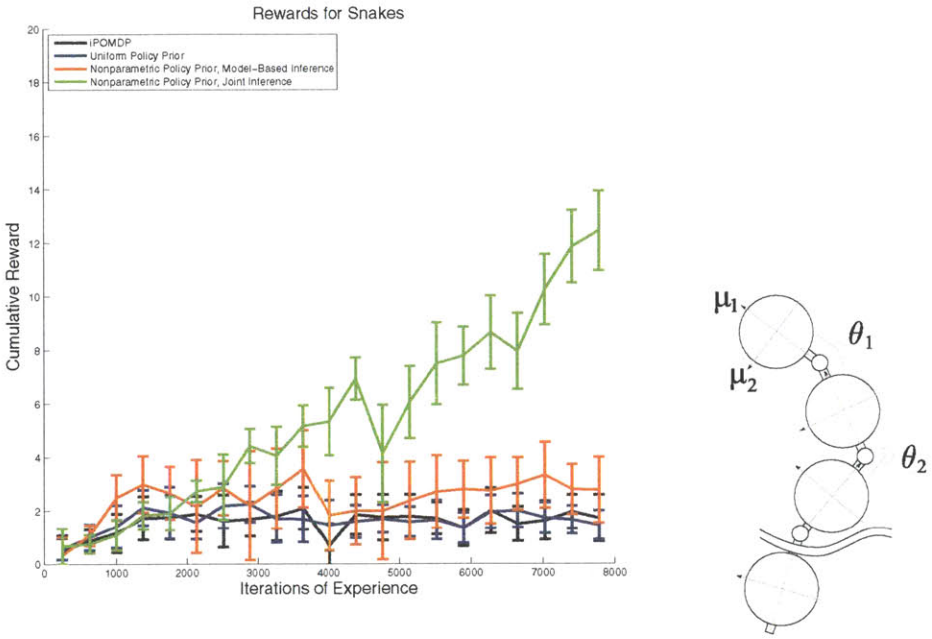


Figure 5-4: Learning curves for the snake. Error bars are 95% confidence intervals of the mean. On the right are three segements of the nine-segmented snake robot.

(modified from [Littman et al., 1995], 26 states), a 1-person version of follow ([Ross et al., 2008a], 26 states) hallway ([Littman et al., 1995], 57 states), beach (100 states), rocksample(4,4) ([Smith and Simmons, 2004], 257 states), tag ([Pineau et al., 2003], 870 states), and an image-search task (16321 states). In the beach problem, the agent needed to track a beach ball on a 2D grid. The image-search problem involved identifying a unique pixel in an 8x8 grid with three type of filters with varying cost and scales.

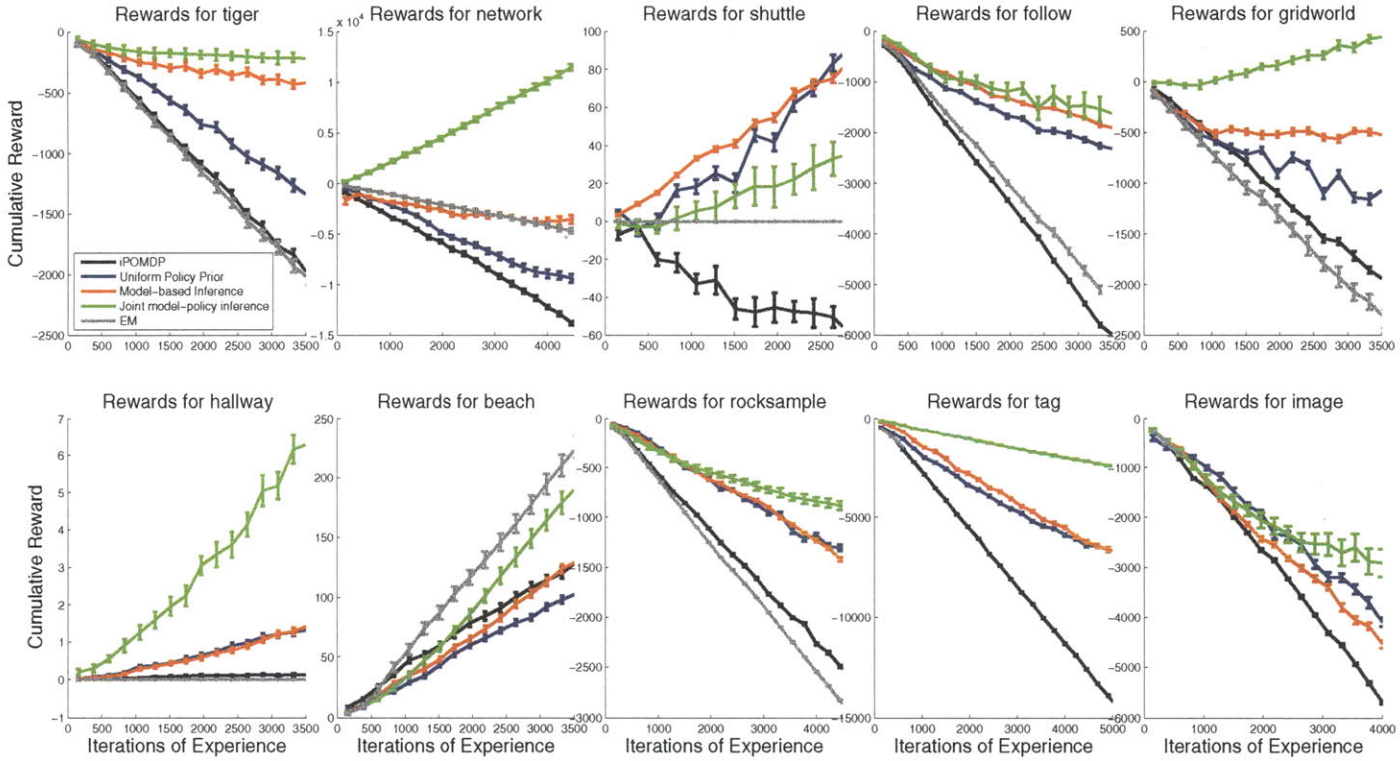


Figure 5-5: Performance on several standard problems, with 95% confidence intervals of the mean.

Figure 5-5 shows the learning curves for our policy priors approaches (problems ordered by state space size). The cumulative rewards and final values are shown in table 5.1. The trials in this chapter contain only 2500 iterations, only a third of the 7500 iterations used to evaluate the iPOMDP in chapter 4. In these shorter trials, the simple iPOMDP agent—which ignores the fact that some of the trials come from expert demonstrations—often does not learn the domain. The policy prior approaches outperform the iPOMDP agent because they used the information that some of the trajectories come from experts (and were thus near-optimal). Even problems with relatively large state-spaces had relatively simple near-optimal policies which could be inferred from the expert data and used to inform what models were probable. Finally, the optimization used in the policy-based approach—recall we use the stochastic search to find a probable policy—was also key to producing reasonable policies with limited computation.

	Cumulative Reward					Final Reward				
	iPOMDP	Uniform Policy Prior	iSC Policy Prior, Model Infernence	iSC Policy Prior, Joint Inference	EM	iPOMDP	Uniform Policy Prior	iSC Policy Prior, Model Infernence	iSC Policy Prior, Joint Inference	EM
tiger	-2.2e3	-1.4e3	-5.3e2	**-2.2e2**	-3.0e3	-2.0e1	-1.0e1	-2.3	**1.6**	-2.0e1
network	-1.5e4	-6.3e3	-2.1e3	**1.9e4**	-2.6e3	-1.1e1	-1.2e1	-4.0e-1	**1.1e1**	-4.7
shuttle	-5.3e1	7.9e1	**1.5e2**	5.1e1	0.0	1.7e-1	3.3e-1	6.5e-1	**8.6e-1**	0.0
follow	-6.3e3	-2.3e3	-1.9e3	**-1.6e3**	-5.0e3	-5.9	-3.1	-1.4	**-1.1**	-5.0
gridworld	-2.0e3	-6.2e2	-7.0e2	**4.6e2**	-3.7e3	-1.3	5.3e-1	1.8	**2.3**	-2.1
hallway	2.0e-1	1.4	1.6	**6.6**	0.0	8.6e-4	7.4e-3	1.4e-2	**1.9e-2**	0.0
beach	1.9e2	1.4e2	1.8e2	1.9e2	**3.5e2**	2.0e-1	1.1e-1	1.4e-1	2.7e-1	**3.4e-1**
rocksample	-3.2e3	-1.7e3	-1.8e3	**-1.0e3**	-3.5e3	-1.6	-5.3e-1	-1.3	**1.2**	-2.0
tag	-1.6e4	-6.9e3	-7.4e3	**-3.5e3**	-	-9.4	-2.8	-4.1	**-1.7**	-9.1
image	-7.8e3	-5.3e3	-6.1e3	**-3.9e3**	-	-5.0	-3.6	-4.2	**1.3e1**	-5.0

Table 5.1: Cumulative and final rewards on several problems. Bold values highlight best performers.

5.5 Discussion

This chapter addresses a key gap in the learning-by-demonstration literature: learning from both expert and agent data in a partially observable setting. Prior work usually tends to either place distributions on—or in some other way directly learn—transition, observation, and reward parameters from the agent's own self exploration [Jaulmes et al., 2005, Ross et al., 2008a,b, Doshi et al., 2008] or learn policies or reward models directly from experts [Abbeel et al., 2006, Ratliff et al., 2009] assuming that the dynamics model (transitions and observations) are known. Our Bayesian approach introduces a joint prior over the world models and policies, connecting information about world dynamics and expert trajectories. Taken together, these priors are a new way to think about specifying priors over models: instead of simply putting a prior over the dynamics, our prior provides a bias towards models with simple dynamics and simple optimal policies. We show with our approach expert data never reduces performance, and our extra bias towards controllability improves performance even without expert data.

In some ways, the work in this chapter is an extension of our previous work [Doshi et al., 2008] that describes how expert knowledge about the policy can be incorporated when given in the form of policy queries, that is, history-action pairs (h, a). While answering single policy queries at first may seem like less work for the expert, incorporating the result of a single query into the prior over models is challenging; the particle-filtering approach of Doshi et al. [2008] can be brittle as model-spaces grow large. The approach also requires the expert to update himself to the agent's internal state given the history rather than just demonstrate a good series of actions. In general, we find empirically that learning from expert trajectories is more robust than from learning from single-action corrections; it appears that using whole trajectories lets the agent generalize better and evaluate models more holistically. By working with models and policies, rather than just models as in Doshi et al. [2008], we can also consider larger problems which still have simple policies.

There are several avenues for extending this into future work. First, the inference

method in section 5.2.1, while relatively robust, is still sensitive to the annealing parameters. Alternate means of sampling from the desired posteriors could make the process even more robust. Next, our policy priors over nonparametric finite state controllers were relatively simple; classes of priors to address more problems is an interesting direction for future work. Finally, in this work, we simply assumed some expert trajectories became available to the agent over time. Targeted criteria for asking for expert trajectories, especially one with performance guarantees such as Doshi et al. [2008], would be an interesting question in active-learning.

Chapter 6

Infinite Dynamic Bayesian Networks[1]

In both chapters 4 and 5, we learned flat models of the environment: that is, the model assumed that a discrete, scalar s completely described the latent state of the environment. This formulation is fairly general, even if we believe the environment might consist of several hidden states. For example, consider a robot whose location (which room) and orientation (north, south, east,or west) are both not directly observable. If we define states to be combinations of locations and orientations, such as kitchen-north or bedroom-west, then we can still encode this domain with a single, discrete variable s for each configuration of the environment.

However, flattening the state-space in this way loses key elements of the structure present in the application domain. In our robot example, certain actions, such as rotations, may only affect the robot's orientations, while other actions, such as movements, may only affect the robot's location. Certain observations, such as the features of a room, may be predominantly a function of the robot's location rather than its orientation. These patterns would be lost if the hidden state were collapsed into just one scalar. This loss of structure is not simply an aesthetic consideration: a flat model, in which we effectively assume that every current part of the state space

[1]**This work was joint with David Wingate**

can affect every future part of the state space and every observation, has exponentially many more parameters to learn than a structured model. Thus, a flat model, while general, may require much more data to learn.

The obvious alternative, of course, is not to flatten the hidden state s. Instead of being a scalar, s is now a vector of *factors* $s^1, s^2, ..., s^K$, where each dimension s^k is a scalar that represents some aspect of the environment (such as location or color). Representing the state with a set of factors not only allows us to encode the structure present in the environment, it can also allow for more efficient inference. Learning can also require fewer samples than a flat model because data for relevant aspects of the environment can be aggregated: for example, one can use all instances of being in the kitchen to learn what kind of floor is likely to be observed, rather than having to learn the observation model separately for the flattened states kitchen-north and kitchen-west.

A common model for representing factored environments is the dynamic Bayesian network (DBN), shown in figure 6-1. Formally, a regular dynamic Bayesian network (DBN) is a directed graphical model in which the state s_t at time t is represented through a set of factors $\{s_t^1, s_t^2, \ldots, s_t^K\}$ (also called nodes). The value of a node—or state—s_{t+1}^k at time $t+1$ is sampled from $T(s_{t+1}^k | Pa_k(s_t))$, where $Pa_k(s_t)$ represents values of the parents of node k at time t. The parents of a node always come only from the previous time slice (there are no intra-slice connections). For example, in our 2-factor robot example, the robot's current orientation may only depend on its previous orientation. However, its current location may depend on both its previous location and orientation.

The state of a DBN is generally hidden; values of the states must be inferred from a set of observed nodes $o_t = \{o_t^1, o_t^2, \ldots, o_t^N\}$. The value of an observation o_t^n at time t is sampled from $\Omega(o_t^n | Pa_n(s^t))$, where $Pa_n(s^t)$ represents values of the parents of observed node o^n at time t. For example an observation corresponding to the type of flooring might only depend on the robot's (hidden) location, while an observation corresponding to whether the robot sees a door might depend on both the robot's current location and orientation. (See Murphy [2002] for very detailed overview of

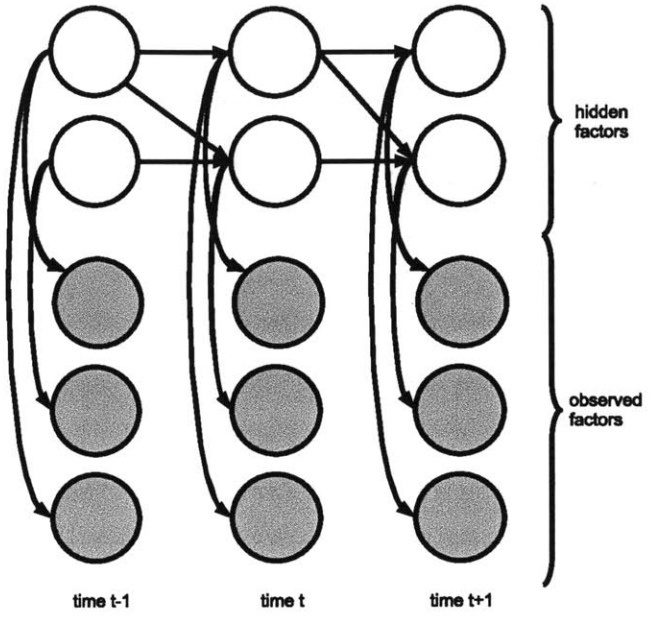


Figure 6-1: Graphical model for the DBN: the hidden states (across the top) are causally connected through time. The visible states (below) are independent of each other and of states at other times given the hidden states at that time step.

DBNs).

In some cases, the structure of the DBN is known beforehand: for example, an engineer might provide a diagram of how unobserved elements of a robotic system might be expected to interact with each other. In other cases, the structure of the DBN may not be known, but all of the nodes (including the states s) are fully observed; work that allows for unseen data [Ghahramani, 1998, Xing-Chen et al., 2007, Peña et al., 2005] still assumes knowledge about the number of hidden nodes and their values. However, in general it may be unclear how many hidden nodes are needed to explain the observed data, how they are connected, or even what values they may take. The core contribution of this chapter is a very general way of learning the hidden structure of a DBN.

Nonparametric extensions of the DBN have tackled various aspects of the DBN structure-learning problem. The Infinite Factored HMM [Van Gael et al., 2009] posits that there are a potentially unbounded number of binary factors that explain the

observed data, while the Infinite Hierarchical HMM [Heller et al., 2009] posits that there are a potentially unbounded number of discrete-valued factors that explain the observed data. Both of these models assume a fixed dependency structure: the iFHMM assumes that each factor evolves independently, while the iHHMM assumes that each factor is affected by itself and a factor one level above it. The Infinite Latent Events Model [Wingate et al., 2009] posits that there are a potentially unbounded number of binary factors that can have time-varying sequences of causes. Finally, the Adaptive Dynamic Bayesian network [Ng, 2007] allows each factor to take on an unbounded number of values but assumes a fixed number of factors.

In this chapter, we describe a generalization of these models, the infinite DBN (iDBN), that allows for a flexible number of factors, factor values, and factor connections. The model allows each factor to take on an arbitrary number of values (learned) as well as be connected in an arbitrary fashion to previous nodes (also learned). Setting various concentration parameters lets designers manage trade-offs between models with more states and models with more factors without the hard model constraints assumed in previous work.

Unlike in previous chapters, where we empirically tested the combination of belief-monitoring and action-selection phases on sequential decision-making problems, we focus solely on the belief-monitoring problem for the iDBN. Since it is the most complex model in this thesis, we focus on demonstrating its properties and the types of structures it finds in a variety of datasets. We do not consider the question of action-selection, but we note that there are several approaches for acting in factored POMDPs that could be used as solvers in this context [Guestrin et al., 2001, McAllester and Singh, 1999b, Sim et al., 2008].[2]

[2]The standard definition of a DBN does not include actions, but actions can easily be incorporated as fully-visible state factors s. In this work, we focus on the structure learning problem without considering actions. However, the techniques directly apply to cases where actions (or any other state factor) are known.

6.1 Model

When motivating the model for the iPOMDP, we argued that if the state of the world is truly hidden from the agent, then the number of possible world states is also likely unknown. We make a similar argument for the iDBN: if the environment has structure that is truly hidden from the agent, then the agent probably does not know the number of factors in the structure or the way these factors interact. Just as with states in the iPOMDP, which represented waypoints rather than physical states in the environment, the structure learned in the iDBN corresponds to a representation that summarizes the data well and is useful for making future predictions, rather than some "truth" about the underlying system.

Our nonparametric DBN model places a prior over DBNs with unbounded numbers of hidden factors. Inference on this infinite structure is tractable only if the prior ensures that only a finite number of hidden nodes will be needed to explain a finite sample of time-series data. More generally, the following properties are desirable in a general nonparametric DBN model:

- A finite dataset should be generated by a finite number of hidden factors with probability one.
- The structure connecting the hidden nodes should be as general as possible (we do not wish to enforce a particular form of connections as the hierarchical or factorial HMM do).
- Each node should be able to take on multiple values (we do not wish to limit ourselves to binary nodes).

Of these desiderata, the first is the most difficult to satisfy in practice. In the case of the iPOMDP, where the hidden structure was a single state, the length of the history automatically ensured that there were only a finite number of hidden states that were relevant: in a history of length t, at most t states could be visited. However, with an infinite factored structure, care must be taken to ensure that inference for any particular hidden node s^k_{t+1} at time $t+1$ does not require knowing the values of

an infinite number of hidden nodes s_t at time t. Specifically, every hidden node s^k and every observed node o^n must either (1) have a finite number of parents or (2) only a finite number of its parents may have changing values. In the latter case, we could imagine there being hidden nodes s^k whose value is always fixed; thus conditioning on s^k does not have any effect.

We chose our prior with an eye toward tractable inference. Our infinite DBN (iDBN) model posits that the world actually has an infinite number of hidden factors s_k^t at any time t. Only a finite number of factors are needed to explain a finite set of observed nodes; however, as we attempt to model more observed nodes, we expect that more hidden nodes will be required to explain the data.

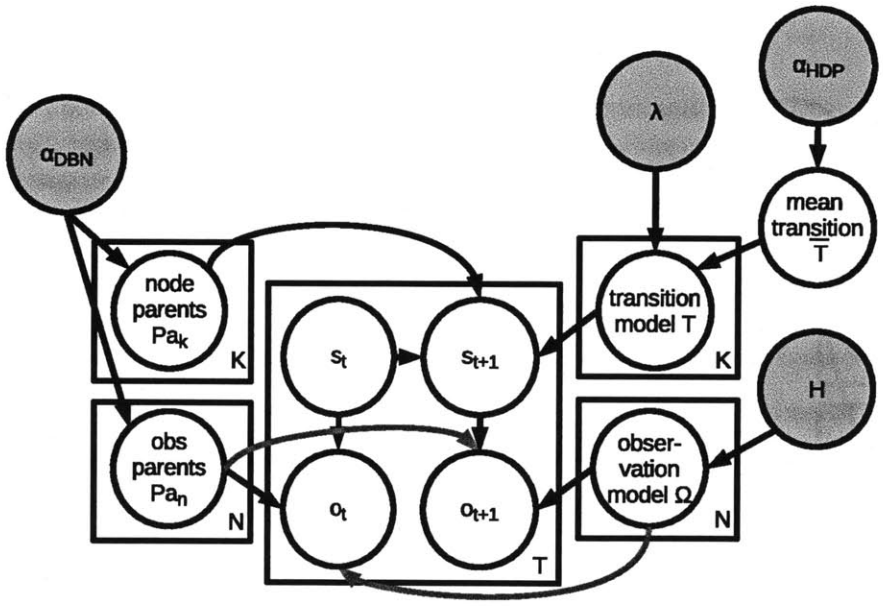


Figure 6-2: Graphical model for the iDBN prior: one concentration parameter, α_{DBN} controls the structure of the connections, while a second, α_{HDP} controls the number of values each hidden node is expected to take in a finite time-series.

We first describe a very general formulation of the iDBN. The general generative process, summarized in figure 6-2, for our iDBN model proceeds as follows: to sample a model m from the iDBN prior, we first, for each observed node, draw zero or more parent hidden factors via a non-parametric process with some concentration

parameter α_{DBN}

$$Pa_n \sim \mathsf{NP}(\alpha_{DBN}) \tag{6.1}$$

where $Pa_n(k) = 1$ if hidden node s^k is a parent of observed node n. Once the observed nodes have chosen parents, all parent hidden nodes choose their own parent nodes via the same process:

$$Pa_k \sim \mathsf{NP}(\alpha_{DBN}) \tag{6.2}$$

where $Pa_k(j) = 1$ if hidden node s^j is a parent of hidden node s^k. This process is repeated for any newly instantiated parents until all hidden nodes that may affect the observed nodes have been instantiated. For example, suppose that there is only one observed node o^1, and it chooses hidden nodes s^1 and s^2 as its parents. Next, nodes s^1 and s^2 would choose their parents: suppose node s^1 chooses only node s^2, but node s^2 chooses itself and a new node s^3. Then we would have to again sample parents for node s^3: suppose it chooses nodes s^1 and s^2. At this point, all nodes' parents are already-instantiated nodes, and we have a finite set of nodes (s^1,s^2,s^3) that are needed to predict observed node s^1.

The process NP should be an exchangeable process with a rich-get-richer property such that (1) nodes choose a finite number of parents with probability one and (2) when a new node is choosing its parents, there is always a finite probability that it not choose any new (uninstantiated parents). Any nonparametric process satisfying (1) and (2) above will ensure number of observed nodes will be explained by a finite number of hidden nodes:

Proposition 1. *If* NP *is a nonparametric process such that the k^{th} node chooses a new (uninstantiated) parent with probability less than some constant c for all k greater than some constant K, then the DBN is guaranteed to have a finite number of nodes with probability one.*

Proof. Once a new node selects no new parents, the process for growing the part of the DBN relevant to the observations is complete. Suppose that the probability that a new (uninstantiated) parent is chosen is always less than c after K nodes have already been instantiated. Then the distribution of number of new parent nodes that

will be added to the DBN is dominated by a geometric distribution with parameter c. Since a geometric distribution outputs a finite value with probability one, only a finite number of nodes will be instantiated with probability one. □

For the application in this chapter, we use the Indian Buffet Process (IBP) [Griffiths and Ghahramani, 2011] as our nonparametric process NP. In the IBP, the n^{th} factor (the "customer") chooses Poisson(α_{DBN}/n) new parents ("dishes"). The probability that the factor chooses no new parents is $\exp(-\alpha_{DBN}/n)$. Figure 6-3 shows how, when using the IBP as NP, the expected number of hidden factors grows logarithmically with the dimensions of the observation.[3]

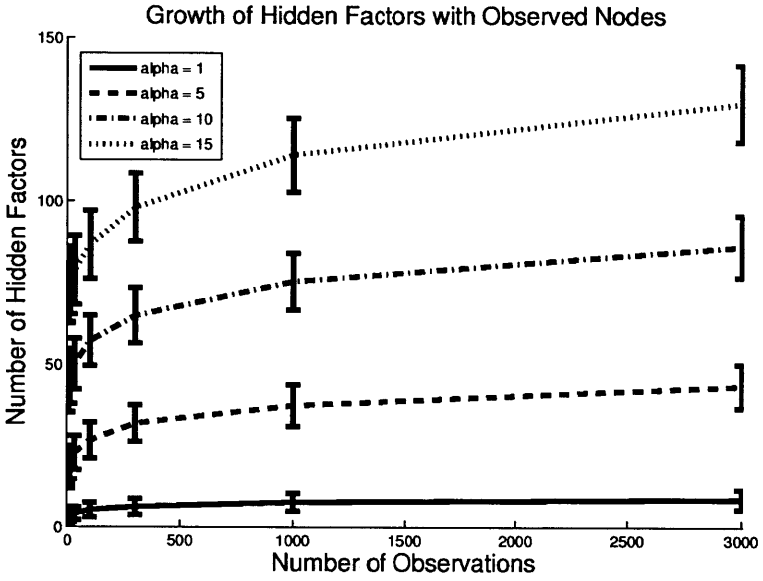


Figure 6-3: Expected number of hidden factors given different numbers of observed nodes, for varying α_{DBN}.

Once the connections of the iDBN are specified, the next step in specifying the iDBN model is describing the prior over transition distributions $T(s^k_{t+1}|Pa_k(s^t))$ and

[3]Our process for sampling inter-factor connections is closely related to the cascading Indian Buffet Process (cIBP) [Adams et al., 2010]. The cIBP uses an IBP used to winnow the number of factors in each layer of a deep belief network, while our iDBN uses the IBP to winnow the number of parents in a two-layer network representing the current and future time slices.

the emission distributions $\Omega(o_t^n|Pa_n(s^t))$. For the emission distribution, we simply specify some base distribution H_n for each observed node. For the transition distributions, we use the hierarchical construction of the hierarchical Dirichlet process HMM (HDP-HMM) [Teh et al., 2006]: we first sample a base, or expected, transition distribution $\bar{T}$ from a Dirichlet process prior, and then use that distribution $\bar{T}$ as the base distribution for each transition distribution $T(s_{t+1}^k|Pa_k(s^t))$.[4]

The complete generative process for the iDBN prior is as follows:

- Sample parents s for all observed nodes o^n according to some nonparametric process $\mathsf{NP}(\alpha_{DBN})$: $Pa_n \sim \mathsf{NP}(\alpha_{DBN})$, where α_{DBN} is the concentration parameter of the nonparametric process.

- While there exist hidden nodes s^k without assigned parents, sample parents for them via the same process $\mathsf{NP}(\alpha_{DBN})$: $Pa_k \sim \mathsf{NP}(\alpha_{DBN})$.

- For each observed node o^n, sample emission distributions $\Omega(o_t^n|Pa_n(s_t)) \sim H_n$ for each setting of the parent variables $Pa_n(s)$.

- Sample a base transition distribution $\bar{T} \sim \mathsf{Stick}(\alpha_{HDP})$, where α_{HDP} is the concentration parameter of the base transition distribution.

- For each hidden node s^k, sample a transition distribution $T(s_{t+1}^k|Pa_k(s_t)) \sim \mathsf{DP}(\bar{T}, \gamma)$, where γ is the concentration parameter for sampling the individual transition distributions.

Besides the properties induced by the nonparametric process, the choices of the concentration parameters adjust the biases of the iDBN prior. The parameter α_{DBN} governs the expected number of hidden nodes that are relevant for predicting the observations. The parameter α_{HDP} governs the number of values, or states, that each hidden node is expected to pass through in any finite sequence. Finally, the parameter γ adjusts how deterministic we expect the transitions to be. Together, these

[4]For simplicity, we used the same base distribution $\bar{T}$ for all hidden nodes k. While it may appear restrictive, evidence from the data still allowed the transitions T to vary; if needed, the hierarchy could easily be extended to sample a private base distribution $\bar{T}_k$ for each hidden node.

three parameters govern the number of factors, then number of values per factor, and the sparsity of the transitions—the three directions which a DBN can be "large" or "non-sparse." Finally, while the iDBN prior ensures that a finite number of hidden nodes will explain a finite number of observed nodes, as time goes on, those hidden nodes may take on new values (as sampled from the HDP prior on transitions) to explain new trends in the observations.

6.2 Methods

As we mentioned at the beginning of the chapter, we focus only on the belief-monitoring aspect of the iDBN, not on control. Thus, in this section we describe how to sample iDBN models m given observations o. Incorporated into a larger decision-making system, the step described here would correspond to drawing samples from the belief $b(s, m)$; an action-selection strategy would be needed to compute an action a given samples from the belief.

In previous chapters, we factored $b(s, m) = b(s|m)b(m)$, noting that in an HMM or POMDP, the conditional belief $b(s|m)$ could be expressed in closed form. The closed form expression allowed us to efficiently compute various quantities needed for control. In factored domains, where s is a vector, the belief $b(s|m)$ grows exponentially with the number of dimensions in s, and thus keeping a closed form expression for $b(s|m)$ is no longer computationally tractable. Instead, we draw samples from $b(s, m)$ directly instead of splitting the belief into two expressions.

Specifically, our inference uses a combination of a blocked Gibbs sampler and Metropolis Hastings (see section 2.2.2 for an overview) to sample combinations of models m and corresponding state sequences s. Throughout this section and for the results, we use the IBP as our nonparametric prior NP because of its straightforward inference properties. We sample potential DBNs from the iDBN posterior by cyclically resampling each of the hidden variables—the values of the hidden factors s, the parent structure for the hidden nodes Pa_k and the observed nodes Pa_n, the base transition distribution $\bar{T}$, and the transition and emission distributions T and Ω—one

at a time conditioned on all of the other variables. The pair of model parameters $m = \{Pa_k, Pa_n, \bar{T}, T, \Omega\}$ and state sequence s from each round of this inferences is a sample (s, m) from the belief $b(s, m)$.

Resampling structure. We separate the process of resampling the structure Pa_n and Pa_k into two parts: resampling connections for already instantiated nodes and changing the number of hidden factors. In the discrete case, given the hidden state sequence s, it is straightforward to integrate out the transition or emission distribution and compute the probability of the hidden state sequence with or without an already-instantiated node as a parent [Heckerman, 1996]. Thus, $p(Pa_n|Pa_k, s, \bar{T}, T, \Omega, o) = p(Pa_n|Pa_k, s, \bar{T})$ and $p(Pa_k|Pa_n, s, \bar{T}, T, \Omega, o) = p(Pa_k|Pa_n, s, \bar{T})$.

To add or delete factors, we use a Metropolis Hastings (MH) birth-death move of the following form:

- Choose whether to attempt adding or deleting a node with probability $p = .5$.
- If attempting to delete a node: only consider nodes whose hidden states are all the same value for all time, that is only delete a node if $s_t^k = c$ for all time t.
- If attempting to add a node: add a node whose hidden state has the same value $s_t^{k+1} = c$ for all times t and connect it to existing hidden and observed nodes with probability p.

Computing the prior probability $p(Pa_k, Pa_n, \bar{T}, T, \Omega)$ of the structure following this MH move is straight-forward because adding or removing a node whose hidden state takes on the same value for all time does not affect the likelihood of the model with respect to the observations, only the probability of the structure. Specifically, recall that the MH acceptance probability is given by:

$$\alpha = \min(1, \frac{P(x')Q(x|x')}{P(x)Q(x'|x)}), \tag{6.3}$$

where x is some random variable. In our specific case the random variable corre-

sponding to x is the probability of the iDBN structure

$$P(x) = P(Pa_k, Pa_n, \bar{T}, T, \Omega) \tag{6.4}$$

Suppose that there are K nodes currently instantiated in addition to the node being added or deleted, and let s^{K+1} denote that special node (note that the indexing is arbitrary, so we can define s^{K+1} to be any node. Let k be the number of nodes—both parents and observations—to which s^{K+1} either was already connected (in the case of node deletion) or to which s^{K+1} is now connected (in the case of node addition). Then, the probability of the transition x to x' in which a node gets added is:

$$Q(x'|x) = .5 * p^k * (1-p)^{(N+K-k)} \tag{6.5}$$

where the .5 comes from the chance of choosing to add a node and the binomial term $p^k * (1-p)^{(N+K-k)}$ represents the probability of selecting a certain set of connections to the instantiated nodes. The probability of a transition x to x' in which a node gets deleted is

$$Q(x'|x) = .5 \tag{6.6}$$

where the .5 comes from the chance of choosing to delete a node. No other terms are needed because there is only one way to delete a node: severing all connections that node has to its parents and observations.

A second approach we use to add factors is to sample draws of states $s^{K+1}...$ that are unconnected to the current structure. Since these states are not connected to the data, the values for these states can be sampled directly from the iDBN prior: they simply represent what some of the other infinite nodes in the infinite dynamic Bayesian network might be doing. Specifically, to use the generative process for the iDBN prior described in section 6.1, we first instantiate one new node s^{K+1}, choose its parents, and then choose parents for any new nodes that get instantiated in this process. Next, we sample the state sequence for any of the new nodes, noting that we only need to instantiate the transition and observation distributions T and Ω

for the settings of the parents that actually occur. These are drawn from $\bar{T}$ and H respectively.

This process results in a state sequences for a new set of nodes $s^{K+1}...$ that do not affect the data. However, in following iterations of the sampler, these new nodes may get connected to observations or become parents of existing nodes if they (just by chance) help explain patterns in the data. We delete these new nodes—and any other nodes that do not either directly or indirectly affect the data—after every full round of the sampler.

Resampling transitions and observations We now turn to resampling the parameters of the transition distributions $p(T|Pa_k, s, \bar{T})$ and the emission distributions $p(\Omega|Pa_n, s, \bar{T}, o)$, as well as the base transition distribution $p(\bar{T}|Pa_k, s)$. The base transition vector $\bar{T}$ is infinite-dimensional; following Teh et al. [2006], we store it as $\{\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_N, \bar{T}_u\}$, where each $\bar{T}_n$ is the base probability for some visited state n. The base probability of visiting any of the (infinite) unvisited states is $\bar{T}_u$. We resample $\bar{T}$ using the restaurant-based sampler of Fox et al. [2010b]. Given the finite representation of $\bar{T}$ and the hidden node sequence s, resampling the transition distributions T is straightforward using Dirichlet-multinomial conjugacy; we can similarly resample the emission distributions Ω given the prior H_n and counts from the observed and hidden nodes. Finally, since each hidden node can take on an infinite number of values, we obviously cannot sample distributions for all parent settings of a particular node. Instead, we only sample distributions for which we have data; additional distributions are instantiated on-demand as the sampler resamples the hidden node sequence.

Resampling states Finally, we must resample the hidden node sequence from the distribution $p(s|Pa_n, Pa_k, \bar{T}, T, \Omega, o)$. While exact inference in DBNs is generally computationally intractable, many approximation algorithms exist for inference over the hidden nodes. We applied the factored frontier algorithm [Murphy and Weiss, 2001], a form of loopy belief propagation with a forward-backward message-passing schedule. Just as with the forward-filtering backard-sampling inference that we used for the state sequence of the iPOMDP, we sampled instead of smoothed on the final

backwards pass. By representing the belief over states at every time step as a product of node marginals, the factored frontier adds one more approximation to our truncated representation of $\bar{T}$ that groups all unvisited states into one extra node. However, we found no empirical difference between this computationally-efficient approximation and inference using a particle smoother [Doucet and Johansen, 2009] that did not require such an approximation to perform inference over the infinite-dimensional state space. The inference over the state sequence is required for structure-learning algorithms for finite DBNs (e.g. Ghahramani [1998]) as well; in our experiments almost 90% of the computational time was spent in this step. Thus, the iDBN prior does not add significant overhead to the inference.

6.3 Experiments

In this section, we describe a set of experiments to demonstrate the capability of the iDBN to find structure in data. We first show some demonstrations that illustrate the properties of the iDBN, and we also test the iDBN on several synthetic datasets. We then apply the iDBN to finding structure in two realworld datasets: precipitation patterns in the United States, and neuronal activity in zebra finches.

For all of the examples, we start out with a time-series of observations o. We randomly choose some of these observations o' to hold-out, or treat as missing, and denote the remaining observations as $o - o'$. For example, if o represents the precipitation over time for various weather stations around the United States, then o' would correspond to stating that certain stations did not report on certain days (even though we have data for them). We use the bulk of the observations to draw samples from the posterior over models m from the posterior $b(s, m|o - o')$. We evaluate these models according to their likelihood $p(o'|m)$ on the held-out data o'. The intuition is that if the model m has characterized the patterns underlying the data well, then it should be able to predict the values of hidden or missing data well.

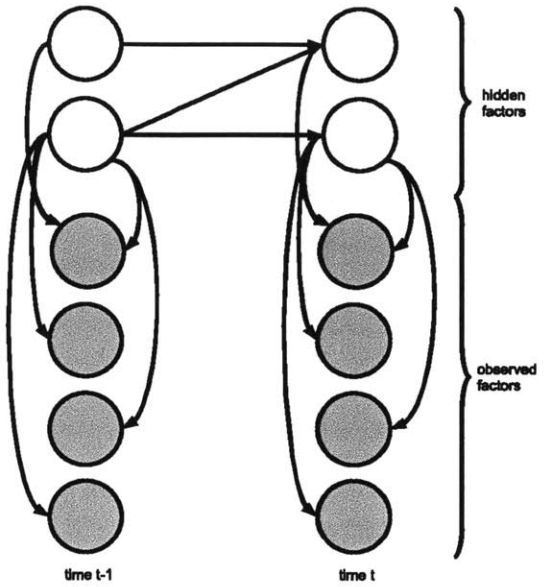


Figure 6-4: Simple DBN with 2 hidden nodes.

6.3.1 Demonstration on a Toy Dataset

We demonstrate various properties of using the iDBN prior using data produced from the simple DBN with 2 hidden nodes and 4 observed nodes shown figure 6-4. Figure 6-5 plots the negative predictive log-likelihood of the finite DBN models and the iDBN on held-out test data, where the predictive likelihoods were computed by holding out 10% of the data from a time-series with 250 time-steps. Error bars show the standard error of the mean averaged from five 50-iteration runs of the sampler. As expected, increasing the number of hidden nodes helps initially because the flat model cannot fully explain the data. However, the larger finite models overfit the training data and thus make larger errors on the held-out test data. The iDBN prior infers a distribution over the number of hidden nodes (right pane of figure 6-5) and node values that generalizes to predict the held-out data well.

Many explanations can exist for a given sequence of observations: for example, suppose the "true" underlying model had 2 hidden nodes which took on 2 and 3 state values, respectively. While it would lose the structure, the model could also

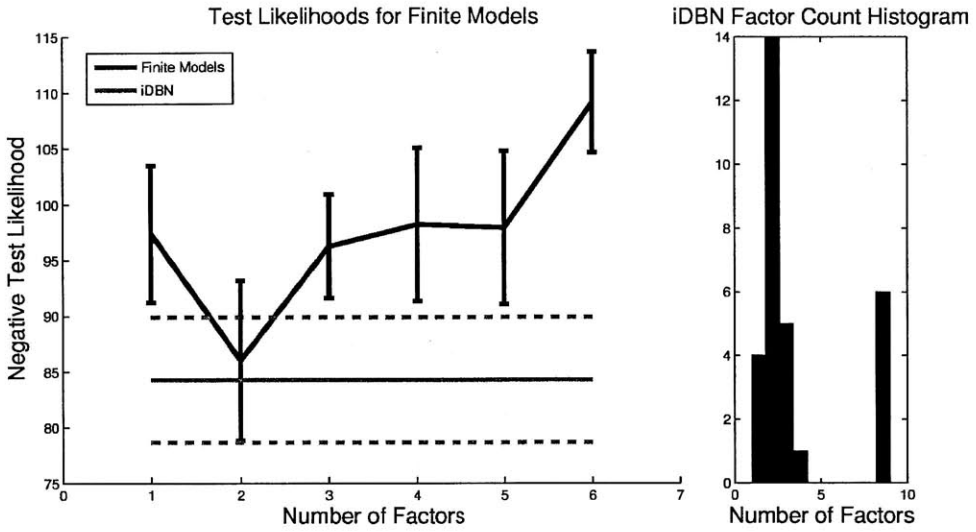


Figure 6-5: Negative log likelihoods for finite models compared to the iDBN. Error bars show the standard error of the mean.

be represented by a flat model with a single hidden node with 6 state values. In figure 6-6, we show how adjusting the α_{DBN} and α_{HDP} in the iDBN prior biases the posterior toward more factors and more states, respectively. As expected, the number of hidden factors in the posterior increases with α_{DBN}, while the number of states the hidden factors taken on increases with α_{HDP} (though with less sensitivity). However, the number of unique factor-state settings in the sequences' posterior stayed within a small range; changing the concentration parameters made biases for different structures but the posterior still captured the core variations in the data.

Overall, we found that good test likelihoods could be obtained over a variety of different concentration parameters. Over the parameter settings, the inter-quartile range for the test likelihoods was 21.8, suggesting that the iDBN could find a variety of likely models based on the biases given by the designer. Moreover, when empirically tested, using the same base distribution $\bar{T}$ for all of the transition distributions did not seem to be overly restrictive: the evidence from the data was able to shape the individual transition distributions to reasonable values.

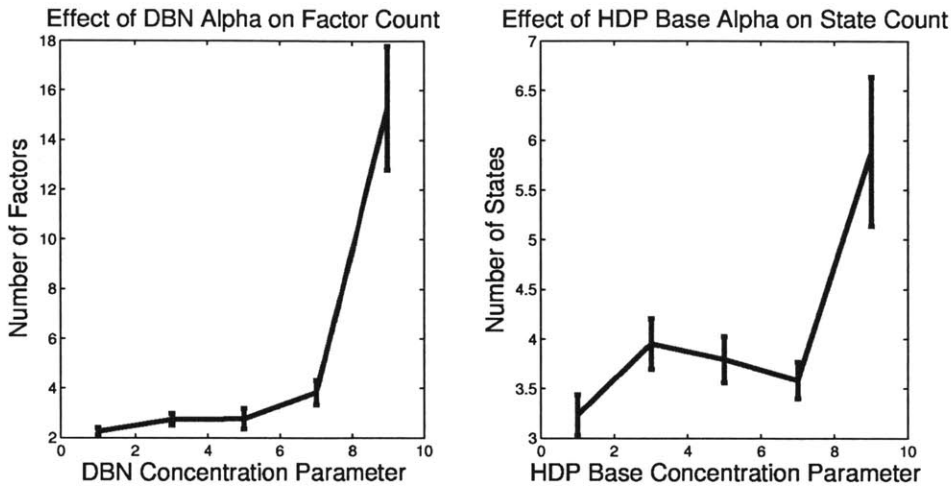


Figure 6-6: Number of hidden factors and number of states discovered by the iDBN; errorbars indicate one standard error of the mean.

6.3.2 Synthetic Datasets

We first show that the iDBN prior generalizes well on several synthetic datasets from the literature. For all of the experiments, we set $\alpha_{DBN} = 1$, $\alpha_{HDP} = 1$, and $\gamma = 3$; the inference was relatively robust to the settings of α_{HDP} and γ; small settings of α_{DBN} definitely helped bias the prior toward smaller models that were more reasonable for the domains we explored. The base emission distribution H_N was set to a uniform distribution with concentration 2. In practice, we found that smaller concentrations—which would have biased the observation distribution toward more peaky observation models—tended to slow down the inference by initially proposing unique states for almost every observation. These states would then have to be merged through the inference process.

We applied the iDBN prior to several datasets from Wingate et al. [2009]. The three network datasets consist of binary observations indicating whether various computers are up or down. Computers crash randomly, and crashes can propagate through the (unknown) network. Each dataset contains an unobserved node which affects the topology of the network. The jungle data set is a synthetic dataset containing a timeseries of noises in a jungle soundscape (where certain animal sounds

cause other animals to also make sounds). Finally, the spike dataset was derived from spike train recordings of hippocampal place cells in a rat while running through a linear track. The data consisted of spike counts that had been passed through a dimensionality-reduction algorithm. The statistics of the datasets are summarized in table 6.1; however note that there are always ways of explaining the data with different numbers of factors or values for factors.

Table 6.1: Description of Datasets

Domain	Factors	Values per Factor	Length
Jungle	6	2	52
Spike	1	45	179
NW-Ring	4	2	1000
NW-Star	5	2	1000
NW-Tree	7	2	1000

We compared the iDBN prior to a finite DBN initialized with the actual number of hidden factors and states from table 6.1 as well as an infinite factorial HMM (iFHMM). The iFHMM (figure 6-7) assumed a specific hidden network structure in which chains of binary hidden nodes evolve independently from each other and every hidden node affects every observed node. We chose these models as comparisons because, like the iDBN, they modeled stationary (non-changing with time) distributions over the hidden states and had somewhat complementary constraints: the DBN fixed the number of nodes but allowed non-binary-valued states, while the iFHMM fixed the number of states per node. The connections for the DBN and the iFHMM were initialized each hidden node with only itself as its parent and connecting to all observed nodes. In the case of the iFHMM, the number of hidden nodes was initially set to the number of observations. To speed up burn-in, the iDBN was initialized with the final iFHMM model; completely random initializations tended to get caught in local optima. All three approaches were run using the same base software with various flags to constrain the numbers of factors and values per factor. A full suite of repeated runs took between 1-4 hours depending on the size of the dataset.

As in section 6.3.1, we randomly held out different subsets of 10% of the observed

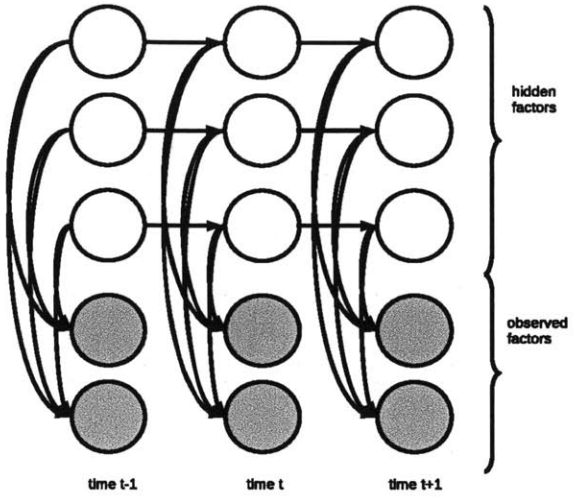


Figure 6-7: Graphical model for the infinite factorial HMM: the hidden nodes (across the top) consist of chains of binary states that evolve independently over time. Each visible state (below) depends on the values of all the hidden states at each time step.

data for 5 runs of the sampler. Each run consisted of 100 iterations, with more complex models initialized from less complex ones. The predictive test-likelihood of each approach was computed over the last 10 iterations of each runs. Table 6.2 summarizes the results: we see that the nonparametric models always outperform the finite model; in all cases the models proposed by the iDBN score either better or comparably to the iFHMM. The DBN—even though it has the "correct" number of states—does less well with limited data due to overfitting. We also emphasize that the structures found by the iDBN are designed to predict the provided data well, not find the "correct"—or even an interpretable—structure: indeed, especially with limited data, there will be many structures that describe the data well. The results show that even though the iDBN is a more flexible prior, it generalizes to unobserved data by finding structure in the model. By allowing for connections between hidden nodes, it can also model structures such as the network topologies better than the more constrained iFHMM.

Table 6.2: Comparison of iDBN approach to other algorithms. Intervals represent the standard error of the mean.

	Negative Test Likelihood			Factors Discovered		
	DBN	iFHMM	iDBN	DBN	iFHMM	iDBN
NW Star	174.0 ± 8.2	165.2 ± 3.0	**156.2 ± 3.0**	5	12.8 ± 0.2	2.4 ± 0.2
NW Tree	255.6 ± 7.1	286.5 ± 2.9	**216.2 ± 10.0**	7	12.0 ± 0.0	4.0 ± 0.4
NW Ring	181.7 ± 16.0	154.3 ± 1.6	**151.4 ± 2.8**	4	9.0 ± 1.2	4.2 ± 1.0
Spikes	142.4 ± 2.7	**133.1 ± 2.1**	136.0 ± 2.8	1	15.9 ± 0.1	18.1 ± 6.2
Jungle	14.8 ± 1.4	**13.9 ± 1.5**	14.2 ± 1.6	6	3.1 ± 0.1	29.5 ± 3.6

6.3.3 Application: Weather Modeling

For this test, we downloaded historical weather data from the US Historical Climate Network[5]. In the first test, we used daily precipitation values for 5 different weather stations (one each in Rhode Island, Connecticut, New Jersey, Delaware and California) for 10 years between 1980-1989, resulting in 3,287 timepoints. Observations were evenly discretized into 7 values.

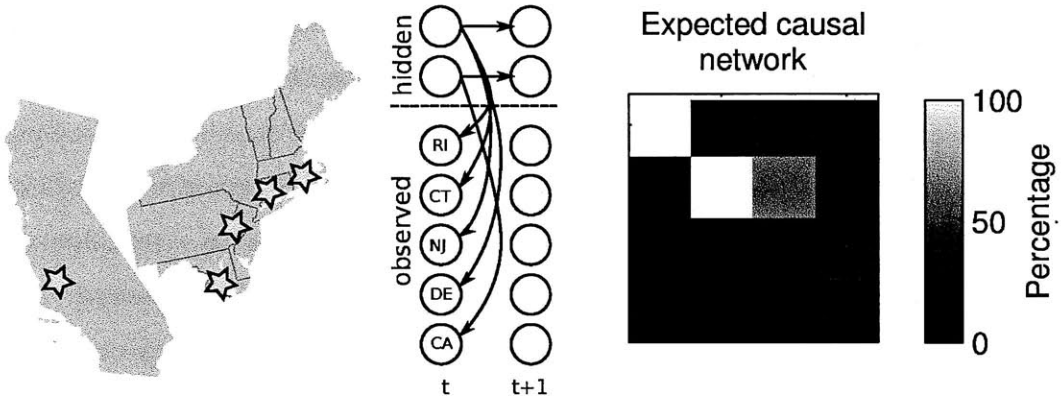


Figure 6-8: Results on the weather dataset. On the left: the weather stations. Middle: the inferred DBN. On the right: the expected causal connections between latent factors.

Figure 6-8 shows the results on this small time-series: on the left is the learned

[5]From ftp://ftp.ncdc.noaa.gov/pub/data/ushcn/daily/

DBN, which identified two independent weather systems for New England and California. This interpretation was stable across many samples from the posterior, as shown in the right hand side. An entry (i, j) in the matrix represents the percentage of samples in which there was a causal connection from parent j to child i (the model occasionally inferred one extra connection (the square 2,3) which did not connect to any observation). Here we see the iDBN naturally picking out the independently evolving latent factors that the iFHMM is designed to model.

Figure 6-9 shows the results of the iDBN applied on a time-series of 500 weather stations across the United States. As before, the algorithm does not have access to the weather station locations; it only sees a time-series of discretized precipitation data. The data can therefore be represented as a matrix with 500 rows (representing stations) and 3,287 columns (representing days). Figure 6-9 shows the results. Not only does the iDBN find geographically-localized clusterings of the observations *without any prior geographical knowledge*, the west-to-east causal links are consistent with U.S. weather patterns (due to the jet stream). A close look at the figure also shows that while most observations connect to a single hidden factor, representing a region, many observations also connect to more than one hidden factor, representing that they lie at the intersection of several regions. The quantitative comparison in figure 6-10 shows that iDBN finds models with lower training and test likelihoods than the iHMM, iFHMM or a flat HMM with up to 100 states.

6.3.4 Application: Discovery of Neural Information Flow Networks

For our final application, we applied the iDBN to analyze neural activity recordings from the auditory pathway of zebra finches. First analyzed in Smith et al. [2006], the dataset corresponds to (possibly misplaced) electrodes put in the cerebral auditory regions of zebra finches. Raw data was discretized into three observations per electrode. The goal of the analysis was to infer functional connectivity between different brain regions given only a timeseries of electrode measurements. We expected factors

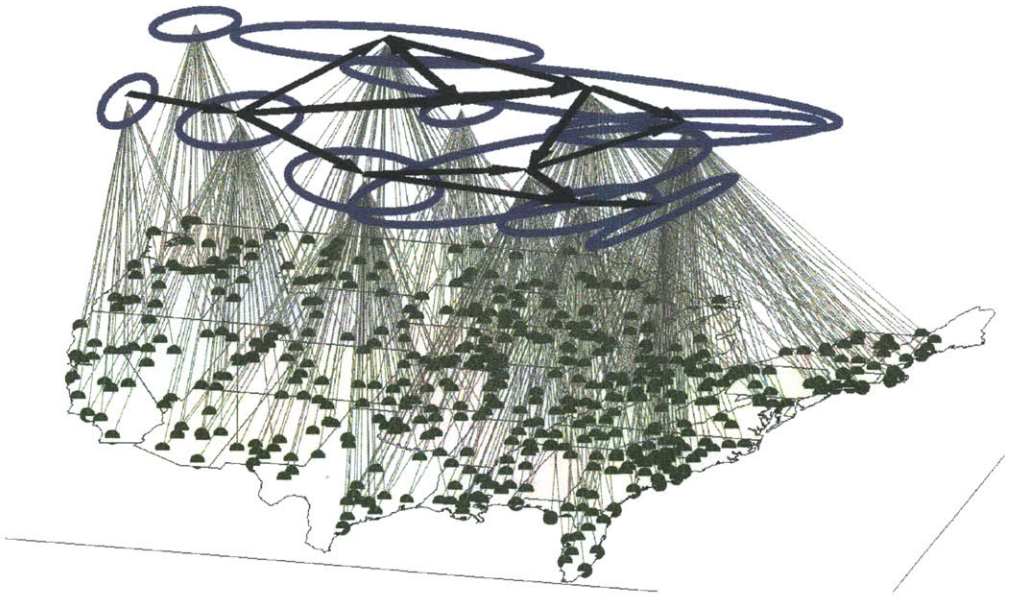


Figure 6-9: Sample network inferred by the iDBN based on 500 weather stations across the United States.

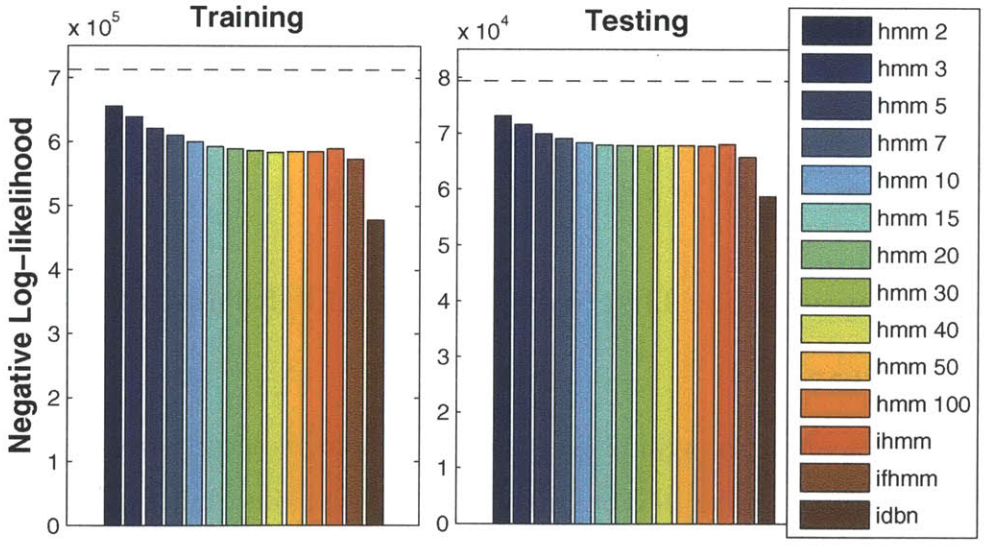


Figure 6-10: Training and test likelihoods for the iDBN, iHMM, iFHMM, and HMM models on the full weather data. Dashed line represents random guessing with the marginal empirical distribution.

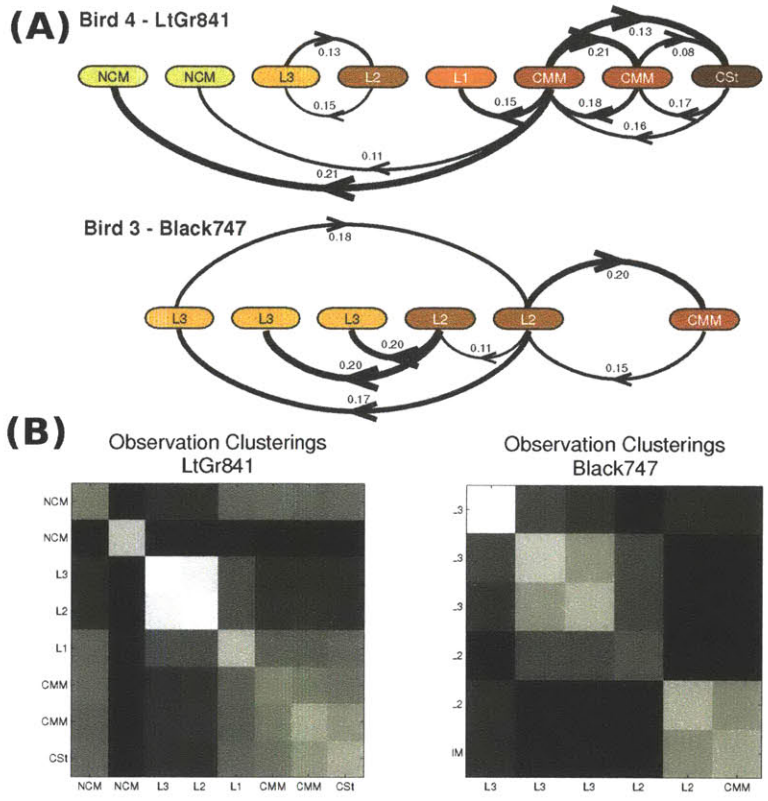


Figure 6-11: Results on the finch dataset. (A) Inferred functional connectivity from Smith et al. [2006]. (B) the iDBN's inferred observation clusterings (*figure (A) courtesy of V. Anne Smith*).

to correspond to functional regions of the brain and causal connections to represent information or processing pathways.

We analyzed data for two birds (Black747 and LtGr841). We first tested the iDBN on temporally-scrambled versions of the datasets. It reliably inferred that no causal connections existed between the hidden factors, suggesting that the temporal connections found in the unscrambled dataset were not a product of chance. Figure 6-11(B) shows clusterings found in the unscrambled time-series: each entry (i, j) in the square represents the frequency with which observation dimensions i and j were connected to the same parent. Over many runs of the iDBN, several observation factors were collapsed into a single state variable—implying that more often than not,

the differences between some observations were not significant enough to justify their own factors.

The groupings are anatomically plausible: for example, in the LtGr841 block, we find that L2 and L3 were often grouped together into a single state variable; similarly, in Black747, we see that CMM and L2 were often grouped together. These observational clusterings correlate strongly with the inferred functional connectivity graphs from the original paper (figure 6-11(A)); the fully-observed-DBN approach of Smith et al. [2006] cannot infer the same collapsing of variables.

6.4 Discussion

In this chapter we described the iDBN, a nonparametric prior over dynamic Bayesian networks that posits that the world contains an infinite number of hidden nodes as well as observed nodes; however, only a finite number of hidden nodes are needed to explain a finite number of observed nodes. By using the iDBN as a prior over hidden nodes, we automatically infer the number of hidden factors—and the number of state values they take on—to explain the observations. Adjusting concentration parameters lets us tune the models to the type of structures we prefer to find. On a variety of datasets, the iDBN finds reasonable structure, ranging from independent chains to highly connected subsets of latent factors. Importantly, this flexibility does not compromise the likelihood of the data, which is on par or better than more structurally constrained models.

Most similar to the iDBN model is the infinite state Bayesian network (ISBN) of Welling et al. [2007]. Like the iDBN, the ISBN uses HDPs as priors over a Bayesian network of latent variables that can take on a countably infinite number of values. Each observation has one or more of these latent variables as its parents. While the ISBN assumes that the Bayesian network structure is given (including the number of latent variables), the iDBN learns both the number of latent variables and their connectivity within the DBN structure.

Still, there are several avenues for further work. Within the basic iDBN model

described in this chapter, we observed in our experiments that sometimes the iDBN is too flexible: for example, suppose that we have seen a set of transitions for factor s^k that cannot be explained by the current data. The form of the likelihood makes it such that, even with different settings of α_{DBN} and α_{HDP}, two very different options—adding a new node s^{K+1} as a parent and letting the parents $Pa_k(s)$ just take a new set of values—are almost indistinguishable. A more structured form for the transition, such as a tree-structured or a linear model, would restrict the flexibility of the prior but also provide the model more direction when choosing to add factors or add values to factors. Developing more sophisticated inference techniques to allow for faster mixing is also an important element to making the iDBN usable as a belief-monitoring step in a larger sequential decision-making system.

More generally, the iDBN provides a very flexible way to model latent structure in observed time-series, and it also raises several interesting questions in non-parametric time-series modeling. For example, even in the general form of the iDBN formulation, the nonparametric process NP which ensures that each child only has a finite number of parents also results in several popular parent factors that influence many parts of the network. Depending on the application, however, it may not be reasonable to think that a few factors tend to affect most other factors. For example, if the hidden factors represent regions on a map, then we might want the prior to be biased toward planar structures (instead of having to infer such a property from the data). The process also enforces that a *fixed*, finite number of hidden factors are needed to explain fixed number of observed nodes, no matter how long the time series. One can also imagine scenarios—such as target tracking—where it might be more reasonable to assume that the longer the time series, the more hidden factors may be needed to explain the data. Developing other nonparametric time-series models—including those that can model non-stationary and relational data—remains an interesting area for future work.

One concrete example of an alternative formulation that satisfies the desiderata in section 6.1 is a prior which allows the connections between the hidden nodes s^k and connections from the hidden nodes s^k to the observed nodes o^n to be completely

arbitrary—that is, nodes could have infinite numbers of parents. However, we place a constraint on the transition function $T(s^k_{t+1}|Pa_k(s_t))$ that required $s^k_{t+1} = s^k_t$ with probability $p_k = 1 - \alpha^k$ for some parameter $\alpha \in (0, 1]$. Intuitively, this constraint means that "higher" factors—factors with larger k—are unlikely to transition. Indeed, it is possible to show in this model that only a finite number of factors will transition from their initial state in a finite period of time (with probability one). As a result, a node may have an unbounded number of parents without violating the desiderata in section 6.1. However, we found inference in this model was less straight-forward than the procedure we used with the iDBN; without a compelling reason to choose this prior, we decided to explore a prior with simpler inference.

Finally, we believe that a factored model will, in general, be needed for control or sequential decision-making problems in partially observable domains. Most interesting domains—robotics, user interfaces, medicine—have large numbers of observed variables and definite structure in the underlying system. However, in many of these domains, we also have some sense of what might be probable sets of hidden variables. The iDBN can incorporate expert knowledge about structures as fake-counts on the Beta variables used as priors for the structure variables Pa_k and Pa_n. However, more work is needed in giving these very flexible structure-learning models flexibility in the "right" places for interesting and useful structures to be inferred.

Chapter 7

Conclusions and Future Work

In this thesis, we presented three examples of Bayesian nonparametric approaches for learning representations in partially-observable domains. The infinite POMDP, chapter 4, placed a distribution over dynamics models that would be Markovian if the value of a discrete, scalar hidden variable was known (but the hidden variable could taken on a potentially unbounded number of values). In chapter 5, we used a similar trick to place a distribution over policies that could explained by a discrete, scalar hidden variable. Finally, in chapter 6, we placed a prior over dynamics models which were Markovian in a vector-valued, discrete hidden variable of unbounded dimensionality.

Following from the Bayesian reinforcement learning framework, the distributions over these hidden variables could now be used as sufficient statistics of the agent's past history when making predictions about the future. The nonparametric aspect of these approaches allowed the size of the sufficient statistic to be scaled as the data required it—we did not have to make any initial assumptions of how complex the distributions over history might be. Besides providing a way to characterize uncertainty, the Bayesian aspect of these approaches governed the trade-off between expanding the sufficient statistic to better explain patterns in the histories while not overfitting to the data at hand. Given this sufficient statistic, the machinery of the Bayesian reinforcement learning framework could then be used to select actions.

As we conclude, we both highlight in what situations the techniques presented in

this work are likely to be useful as well as directions for future work.

7.1 When is this useful?

Now that we have seen a few examples of the uses of Bayesian nonparametric methods to reinforcement learning, we return to the pair of questions that must be answered in any reinforcement learning problem: what representation should one use for the application, and how does one select actions given a representation? This thesis focused on providing one alternative to first question of choosing and learning a representation, and in chapter 3 we described several other approaches to creating sufficient statistics in a partially-observable reinforcement learning setting. When does using a Bayesian nonparametric approach to learning a representation make sense? There are several factors which might make using a Bayesian nonparametric approach attractive:

- **Representations Must Be Built Online.** The iPOMDP in chapter 4 initially learned a small model to explain gross patterns in the data and then refined it. Smaller models are generally computationally easier to solve, leading to computational wins early in the training phase. These computational wins become less significant as the model is refined. The incremental history-based approaches, such as U-Tree, share this property.

- **Training-Time Performance Matters.** By capturing the gross structure in the dynamics initially, the iPOMDP models quickly learn the patterns that allow for reasonable action-selection. Models with more parameters often take longer to pick out patterns. The incremental history-based approaches also share this property. In applications where a large batch of histories is available from the start, or a large training set can be obtained at little cost, this property will have less value.

- **Data is Limited or Fundamentally Sparse.** When gathering experience from the environment is easy, such as with a game-playing AI, just about any

representation will be able to discover reasonable structures: long suffix-trees or many-node PDFAs can be trained without overfitting, statistics can be precisely computed for training PSRs, and sufficient data exists to train the many parameters in a POMDP. While learning an appropriately-sized representation may still be a pertinent question, the computational overhead associated with using a Bayesian nonparametric approach may not outweigh its regularization benefits compared to a simpler model comparison methods such as cross-validation.

In contrast, Bayesian methods are well-suited to situations in which the agent's experience is limited: we observed in chapter 4 that the Bayesian approaches outperformed U-Tree with limited data. Bayesian nonparametric methods go a step farther by automatically scaling the model to the amount of experience—we also saw in chapter 4 that a poorly-sized parametric Bayesian model resulted in slower learning. These generalization properties are also valuable if large amounts of data are available but the data is still fundamentally sparse: for example, new conditions might always be appearing in medical applications, and mapping agents may always continue to explore new spaces. These domains benefit from the combination of having a model with an infinite capacity—we do not need to fix a model-size—with a prior to prevent overfitting.

- **Problem Structure is Poorly Understood or Irrelevant.** All of the benchmark problems that we used in chapters 4, 5, and 6 had one thing in common: even when the domain had a very specific structure—such as the gridworld—none of the structure was given to the agent. In reality, of course, if the domain structure is well-understood, then it makes sense to use a representation and learning technique that can leverage that knowledge. For example, if transitions can be specified, or measurement models can be trained, these simpler learning approaches will perform better than trying to learn everything from scratch.

  There are two situations in which a Bayesian nonparametric approach might make sense even if the domain is well-understood. The first is if there are still parts of the representation that cannot be calibrated using simpler techniques.

In this case, the Bayesian nonparametric approach can be used to learn the "error" or the "extra" parts of the representation that are not already trained. The second is if we believe that much of the domain knowledge may be irrelevant for the predictive task at hand: for example, although medical experts know a lot about anatomy and physiology, this level of detail may not be needed for predicting how a disease may evolve.

- **Predictive Accuracy is the Priority.** Bayesian nonparametric approaches provide a way to tune the sophistication of a Bayesian model to make accurate predictions about the future. As mentioned in chapter 1, the hidden variables should not be thought of as "real" world-states, but rather way-points for making predictions. The discovered structures may be interpretable—such as the regions found in the weather and zebra-finch data in chapter 6—but interpretability is a possible by-product, not a goal. Bayesian nonparametric methods are less appropriate when the primary objective is to learn the "true" system. History-based approaches also share this property.

7.2 Directions for Future Work

We conclude this thesis with a discussion of directions for future work. The examples in this thesis provided a proof-of-concept that Bayesian nonparametric methods can be used to effectively learn Bayesian models. This current work both has several limitations and also suggests specific extensions for future work.

Within the two overall questions of representation choice and action selection, we can divide the question of representation choice into three subproblems. First, we must choose the *form of the representation*, formally the sufficient statistic $s = f(h_t)$. In the reinforcement learning setting, this statistic will be learned from data, and thus the second question is how one intends to *prevent overfitting*: hypothesis tests? Bayesian Occam's razor? Minimum description length? Finally, we must decide upon an *algorithm for the implementation* of the learning scheme. We divide the directions for future work along these three questions regarding the choice of representation as

well as the general question of action-selection.

7.2.1 Choosing a Sufficient Statistic: Bayesian Nonparametric Approaches for History-Based Methods

In this work, we explored the use of Bayesian nonparametric techniques primarily in the context of a learning approach for POMDP-like models. Bayesian nonparametric approaches were a natural fit in this context because a large literature already exists for using Bayesian nonparametric methods to fit time-series data [Fox et al., 2010a, 2008, Stepleton et al., 2009, Johnson and Willsky, 2010], though not necessarily with the objective of control. Using a hidden-variable approach for the representation often makes it easier to incorporate expert knowledge about the underlying dynamics of a system. Finally, although the representations that we learned in this thesis used discrete hidden variable to describe the environment, the "true" underlying state space could be discrete or continuous: our discrete hidden-variable representation creates a discretizaion of environment that is sufficient for making predictions.

However, these techniques can be used to learn other kinds of structure as well: for example, in chapter 4, we explored the use of probabilistic-deterministic infinite automata (PDIA) to directly compress the history. The nonparametric policy prior in chapter 5 used a Bayesian nonparametric approach to learn the expert's policy directly. In the context of history-based methods, the Bayesian nonparametric approach can provide a bias that states that a potentially unbounded length of history (effectively, the entire history) may be needed to summarize the history, however, in most cases, a few features of the history or a small window will do. This bias is exactly the bias introduced by the PDIA in chapter 4, and similar approaches have been used for learning suffix trees in the context of language models [Wood et al., 2011]. In both cases, the Bayesian nonparametric approach provides an alternative to multiple hypothesis-testing or more heuristic model-fitting methods. The inference in these approaches would need to be refined to be used in the context of reinforcement learning, but we hypothesize that given the appropriate algorithms, Bayesian non-

parametric methods could provide similar benefits to history-based approaches—such as learning from limited data—as they have done in hidden-variable approaches.

7.2.2 Fitting and Preventing Overfitting: Improving Priors and Incorporating Outside Knowledge

Improved Priors for Typical Domains The priors used in this work provided a bias toward using fewer parameters rather than more, but they were still extremely vague. For example, the iPOMDP prior (chapter 4) and the infinite state controller prior (chapter 5) simply placed a bias toward using fewer hidden world-states or nodes to explain the data, and the iDBN (chapter 6) had three concentration parameters governing the expected number of hidden factors, their values, and the sparsity of the transitions. Indeed, the only bias actually in the prior is one that is rarely true in reality: all distributions were drawn independently, and every hidden variable had an identical bias toward to returning to a few popular values, regardless of its current value.

While the data was able to overcome these incorrect biases, priors that are more suited to the domain in question will be able to find relevant structure sooner than more general priors. Also, we note that the structures discovered by the iDBN on the synthetic examples in chapter 6 were very different than the kinds of structures we expected to find in those domains. While interpretability is not our primary goal, priors that find structures that are at least somewhat interpretable have the advantage of being easier to analyze.

There are two specific areas in which we believe the priors used in this work could be extended and improved. First, hierarchical priors can allow more sharing between different parts of a structure. For example, with the iDBN, we might expect that if many of a node's parent values have not changed, the node's transition function will be similar to what it was before. Even in the case of iPOMDP, groups of hidden variable settings may behave similarly. Hierarchical models [Heller et al., 2009, Adams et al., 2010] can be used to share statistical strength between shared structure.

Second, all of the priors used in the current work introduced dense connections between variables: the connections in the iDBN induced by the Indian Buffet Process (IBP) is a dense graph in the set of hidden factors[1], and the transitions induced by the HDP-HMM is a dense graph in the set of hidden variable values. Even if the concentration parameters are set such that the prior is biased toward sparse transitions, the Dirichlet distribution and Dirichlet process posteriors will no longer favor sparsity. However, in many domains, connections are sparse: rooms connect to a few other rooms; a few comorbidities can largely explain a medical conditions. Exploring the use of various sparsity-inducing priors—such as negative entropy priors [Brand, 1999], spike-and-slab priors [Ishwaran and Rao, 2005], horseshoe priors [Carvalho et al., 2009]—may result in learning more interpretable models with fewer data, but will introduce significant computational challenges.

Techniques for Making Relevant Predictions Within the reinforcement learning framework, a state that is a sufficient statistic for predicting futures generally captures too much information: we are largely interested in predicting discounted future rewards. More generally, we may only be interested in certain outcomes, such as whether a robot reaches a goal or whether a patient's condition improves, even though many other variables may be measure. The U-Tree algorithm compresses histories based on how well the induced states can be used to predict future rewards; a limitation of current Bayesian approaches, including ours, is that they focus on learning all aspects of an environment as a preliminary step toward choosing a policy. There exists work in reinforcement learning for expanding representations [McCallum, 1993, Geramifard et al., 2011] and compressing representations [Poupart and Boutilier, 2002] based on values; an interesting extension would be to incorporate these kinds of techniques into a Bayesian learning setting.

Incorporating Supervision and Expert Knowledge In chapter 5, we described one approach for incorporating expert knowledge in the form of demonstrations, and

[1] A two-parameter version of the IBP [Ghahramani et al., 2007] can be used introduce some sparsity.

in chapter 6, we noted that prior knowledge about the structure of the hidden nodes in the iDBN could be incorporated through a set of counts. More generally, we can imagine many kinds of expert input during the learning process. In problems where the reward is hard to specify, the expert may instead provide certain quantities that are important to predict well, examples of good trajectories, or labels on the quality of a few of the agent's trajectories. When the problem structure is somewhat well-understood, the expert may wish to provide a bias toward physically, chemically, or biologically plausible models. Understanding how to incorporate these varying forms of information—whether it is in the priors, the inference, or the action-selection—will be key to scaling these (and any techniques) to complex, realworld problems.

7.2.3 Algorithms for Implementation: Inference

The extensions described above will almost certainly add challenges to the inference, and inference is already the toughest part of applying Bayesian nonparametric methods to reinforcement learning problems: the belief monitoring step in Bayesian reinforcement learning is an inner loop within a larger sequential decision-making process. In this work, we optimized the inference using software engineering, and we also took advantage of short-cuts such as importance weighting to reduce the frequency with which an MCMC sampler needed to be run. These techniques will not scale to much larger applications, and we suggest three possible directions for scaling Bayesian nonparametric inference for reinforcement learning.

First, advances have been made in using spectral methods for learning HMMs [Song et al., 2010] and PSRs [Boots et al., 2011a], and recently linear programming relaxations have been used to learn graph structures [Jaakkola et al., 2010]. If it is possible to formulate the form of the Bayesian nonparametric regularizer as a standard optimization problem, then we can at least quickly find a mode of the posterior space. More standard variational or sampling-based approaches could then be used to characterize the uncertainty around this mode.

Second, all of the inference techniques used in this thesis were batch methods. For a system to truly run for long periods of time we must be able to compute

updates sequentially. We expect that computation will be the bottleneck before disk space; thus an interesting set of algorithms might be ones that look at subsets of past data instead of all past data. Very recent work has looked at online inference for various HDP-based structures [Canini et al., 2009, Wang et al., 2011, Rodriguez, 2011], though none of these have yet been applied to a reinforcement learning domain.

Third, as datasets become larger, and structures more sophisticated, understanding the modes in the posterior space becomes more and more important. In the context of sequential decision-making, methods for discovering several relevant modes might be more valuable than knowing the uncertainty around a particular mode. Currently, the standard approach to finding multiple modes is to restart the sampler or the optimization with several different initial values; and interesting area of future work might be iterative smoothing techniques, such as the annealing used in this thesis, to help improve this process. Adapting current algorithms to parallel and distributed computing structures may also help track multiple modes.

7.2.4 Algorithms for Action Selection

This thesis developed and used a fairly sophisticated set of techniques to characterize uncertainty in infinite-dimensional latent structures, and more generally, advances in inference techniques have made it possible to place distributions over a variety of complex structures [Adams et al., 2010, Ross and Pineau, 2008, Heller et al., 2009, Deisenroth et al., 2009, Engel et al., 2005]. However, obtaining a sufficient statistic for the history, of course, is only part of reinforcement learning problem. Given a summary of the history, one still needs to select actions.

In parallel, advanced techniques have been developed for planning in very large problems [Shani et al., 2007, Kurniawati et al., 2008, Silver and Veness, 2010]. However, a question of how to select actions in a Bayesian reinforcement learning framework remains an open question. In chapter 4, we showed that some very simple action-selection strategies performed as well or better than others and hypothesized several explanations: the posterior may be very smooth (have low covering number); with a little randomness, sufficient learning will occur from any reasonable policy;

and our forward-search was simply not deep enough.

These observations raised two theoretical questions with important practical implications. First, when are these very simple action-selection approaches sufficient? Does the Bayesian reinforcement learning setting have certain properties that imply that simple techniques should do well? Second, if these simple strategies work, are there also simplifications that on our representation of state? What information do we need from our characterization of uncertainty—for example, will just knowing a few modes do? The previous directions for future work suggested fairly direct extensions of the key ideas presented in this thesis. However, larger advancements will require a deep understanding of the interplay between representations and action-selection strategies.

Bibliography

Pieter Abbeel, Morgan Quigley, and Andrew Y. Ng. Using inaccurate models in reinforcement learning. In *International Conference on Machine Learning*, pages 1–8. ACM Press, 2006.

D Aberdeen and J Baxter. Scalable internal-state policy-gradient methods for POMDPs. *Proceedings of the Nineteenth International Conference on Machine Learning*, page 3, 2002.

Douglas Aberdeen, Olivier Buffet, and Owen Thomas. Policy-gradients for psrs and POMDPs. In *AI and Statistics*. Society for Artificial Intelligence and Statistics, 2007.

R. P. Adams, H. M. Wallach, and Z. Ghahramani. Learning the structure of deep sparse graphical models. In *AI and Statistics*, 2010.

John Asmuth, Lihong Li, Michael Littman, Ali Nouri, and David Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Uncertainty in Artificial Intelligence*, 2009.

Matthew J. Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. The infinite hidden Markov model. In *Neural Information Processing Systems*, pages 577–584, 2001.

R. Bellman. *Dynamic Programming*. Princeton University Press, NJ, 1957.

D. Blackwell and M. Girshick. *Theory of Games and Statistical Decisions*. Wiley, 1954.

Byron Boots, Sajid Siddiqi, and Geoffrey Gordon. Closing the learning planning loop with predictive state representations. *I. J. Robotic Research*, 30:954–956, 2011a.

Byron Boots, Sajid Siddiqi, and Geoffrey Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proceedings of the 25th National Conference on Artificial Intelligence*, 2011b.

Ronen I. Brafman and Guy Shani. Resolving perceptual aliasing in the presence of noisy sensors. In *Neural Information Processing Systems 17*, 2004.

Matthew Brand. Structure Learning in Conditional Probability Models via an Entropic Prior and Parameter Extinction. *Neural Computation*, 1999.

Leonard Breslow. Greedy utile suffix memory for reinforcement learning with perceptually-aliased states. Technical report, Navy Center for Research Laboratory, 1996.

S. P. Brooks. Quantitative convergence assessment for Markov chain Monte Carlo via cusums. *Statistics and Computing*, 8:267–274, August 1998.

Kevin R. Canini, Lei Shi, and Thomas L. Griffiths. Online inference of topics with latent Dirichlet allocation. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2009.

C. K. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, September 1994.

Carlos M. Carvalho, Nicholas G. Polson, and James G. Scott. Handling sparsity via the horseshoe. *Journal of Machine Learning Research - Proceedings Track*, 5:73–80, 2009.

Jorge Castro and Ricard Gavalda. Towards feasible PAC-learning of probabilistic deterministic finite automata. *Grammatical Inference Algorithms and Applications*, pages 163–174, 2008.

L. Charlin, P.Poupart, and R.Shioda. Automated hierarchy discovery for planning in partially observable environments. In B. Schölkopf, J.C. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.

Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 183–188. AAAI Press, 1992.

Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

Marc Peter Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic moment-based Gaussian process filtering. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 225–232, New York, NY, USA, 2009. ACM.

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*, 39:1–38, 1977.

Christos Dimitrakakis. Bayesian variable order Markov models. In Yee Whye Teh and Mike Titterington, editors, *AI and Statistics*, volume 9, pages 161–168, 2010.

Finale Doshi, Joelle Pineau, and Nicholas Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *International Conference on Machine Learning*, volume 25, 2008.

Finale Doshi-Velez. The infinite partially observable Markov decision process. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 477–485, 2009.

Finale Doshi-Velez, David Wingate, Nicholas Roy, and Josh Tenenbaum. Nonparametric Bayesian policy priors for reinforcement learning. In *Neural Information Processing Systems*, 2010.

Finale Doshi-Velez, David Wingate, Nicholas Roy, and Josh Tenenbaum. Infinite dynamic Bayesian networks. In *International Conference in Machine Learning*, 2011.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In *Handbook of Nonlinear Filtering (eds.* University Press, 2009.

Gary L. Drescher. *Made-up minds: a constructivist approach to artificial intelligence.* MIT Press, Cambridge, MA, USA, 1991.

Michael O'Gordon Duff. *Optimal learning: computational procedures for Bayes-adaptive markov decision processes.* PhD thesis, University of Massachusetts Amherst, 2002.

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.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 201–208, New York, NY, USA, 2005. ACM.

E. Even-Dar, S. M. Kakade, and Y. Mansour. Reinforcement learning in POMDPs without resets. In *IJCAI*, pages 690–695, 2005.

Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

Richard Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *IJCAI*, pages 608–620, 1971.

E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. An HDP-HMM for systems with state persistence. In *Proceedings of International Conference on Machine Learning*, 2008.

E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Sharing features among dynamical systems with beta processes. In *Neural Information Processing Systems 22*. MIT Press, 2010a.

E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Bayesian nonparametric inference of switching linear dynamical systems. Technical Report 2830, MIT Laboratory for Information and Decision Systems, March 2010b.

Jurgen Van Gael and Zoubin Ghahramani. *Inference and Learning in Dynamic Models*, chapter Nonparametric Hidden Markov Models. Cambridge University Press, 2010.

Alborz Geramifard, Finale Doshi, Josh Redding, Nicholas Roy, and Jonathan P. How. incremental feature dependency discovery. In *Proceedings of the 23rd International Conference on Machine Learning*, 2011.

Z. Ghahramani. Learning dynamic Bayesian networks. In *Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks*, 1998.

Z. Ghahramani, T. Griffiths, and P. Sollich. Bayesian nonparametric latent feature models. In *Bayesian Statistics*, volume 8, 2007.

Piotr J. Gmytrasiewicz and Prashant Doshi. Interactive POMDPs: Properties and preliminary results. In *Proceedings of Automonous Agents and Multi-Agent Systems*, pages 1374–1375, 2004.

Thomas L. Griffiths and Zoubin Ghahramani. The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Solving factored POMDPs with linear value functions. In *IJCAI-01 workshop on Planning under Uncertainty and Incomplete Information*, 2001.

E. A. Hansen. An improved policy iteration algorithm for partially observable MDPs. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

David Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, 1996.

K. A. Heller, Y. W. Teh, and D. Görür. Infinite hierarchical hidden Markov models. In *AI and Statistics*, 2009.

M. P. Holmes and C. L. Isbell Jr. Looping suffix tree-based inference of partially observable hidden state. In *Proceedings of the 23rd international conference on Machine learning*, pages 409–416. ACM, 2006.

Christopher Hundt, Prakash Panagaden, Joelle Pineau, and Doina Precup. Representing systems with hidden state. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI Conference on Artificial Intelligence, pages 368–374. AAAI Press, 2006. ISBN 978-1-57735-281-5. URL `http://dl.acm.org/citation.cfm?id=1597538.1597598`.

Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, Berlin, 2004.

Marcus Hutter. Universal algorithmic intelligence: A mathematical top-down approach. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*, Cognitive Technologies, pages 227–290. Springer, Berlin, 2007. ISBN 3-540-23733-X.

Marcus Hutter, William Uther, Pascal Poupart, and Kee Siong Ng. Partially observable reinforcement learning mini-symposium. In *Neural Information Processing Systems*, 2009.

H. Ishwaran and J.S. Rao. Spike and slab variable selection: frequentist and Bayesian strategies. *Ann. Statist.*, 33(2):730–773, 2005.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. *Journal of Machine Learning Research - Proceedings Track*, 9:358–365, 2010.

M.R. James, S. Singh, and M.L. Littman. Planning with predictive state representations. In *International Conference on Machine Learning and Applications*, pages 304 – 311, December 2004.

R. Jaulmes, J. Pineau, and D. Precup. Learning in non-stationary partially observable Markov decision processes. In *European Conference on Machine Learning Workshop*, 2005.

Matthew Johnson and Alan Willsky. The hierarchical Dirichlet process hidden semi-Markov model. In *Uncertainty in Artificial Intelligence*, 2010.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101: 99–134, 1995.

K. Kim, T. Dean, and N. Meuleau. Approximate solutions to factored Markov decision processes via greedy search in the space of finite state controllers. In *Artificial Intelligence Planning Systems*, pages 323–330, 2000.

J. Zico Kolter and Andrew Ng. Near-Bayesian exploration in polynomial time. In *International Conference on Machine Learning*, 2009.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *RSS*, 2008.

M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: scaling up. *International Conference in Machine Learning*, 1995.

David J.C. MacKay. Ensemble learning for hidden Markov models. Technical report, Cambridge University, 1997.

M. M. Hassan Mahmud. Constructing states for reinforcement learning. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning*, pages 727–734, Haifa, Israel, June 2010.

Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, Inc., 1979.

David McAllester and Satinder Singh. Approximate planning for factored POMDPs using belief state simplification. In *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference*, 1999a.

David A. McAllester and Satinder Singh. Approximate planning for factored POMDPs using belief state simplification. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 409–416. Morgan Kaufmann Publishers, 1999b.

Andrew R. McCallum. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 190–196, 1993.

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.

K. P. Murphy and Y. Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Uncertainty in Artificial Intelligence*, 2001.

Radford Neal. Slice sampling. *Annals of Statistics*, 31:705–767, 2000.

B. N. Ng. Adaptive dynamic Bayesian networks. In *Joint Statistical Meetings*, 2007.

P. Orbanz and Y. W. Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer, 2010.

Arthur Pchelkin. Efficient exploration in reinforcement learning based on suffix memory. *Informatica*, 8(2):267–283, 2003.

J. M. Peña, J. Björkegren, and J. Tegnér. Learning dynamic Bayesian network models via cross-validation. *Pattern Recogn. Lett.*, 26, October 2005.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

D. Pfau, N. Bartlett, and F. Wood. Probabilistic deterministic infinite automata. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1930–1938, 2010.

J. Pineau, N. Roy, and S. Thrun. A hierarchical approach to POMDP planning and execution. In *Workshop on Hierarchy and Memory in Reinforcement Learning (International Conference in Machine Learning)*, June 2001.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. *IJCAI*, 2003.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900, 1997.

P. Poupart and N. Vlassis. Model-based Bayesian reinforcement learning in partially observable domains. In *ISAIM*, 2008.

Pascal Poupart and Craig Boutilier. Value-directed compression of POMDPs. In *Neural Information Processing Systems 15*, pages 1547–1554. MIT Press, 2002.

Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In *Neural Information Processing Systems*, 2003.

M.O. Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Nathan Ratliff, Brian Ziebart, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Inverse optimal heuristic control for imitation learning. In *Proceedings AI and Statistics*, pages 424–431, 2009.

Jesse Hoey Robert, Robert St-aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference*, pages 279–288, 1999.

Abel Rodriguez. On-line learning for the infinite hidden Markov model. *Communications in Statistics-Simulation and Computation*, 40(6):879–893, 2011.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Alberta, Canada, 2004.

Stéphane Ross and Joelle Pineau. Model-based Bayesian reinforcement learning in large structured domains. In *Uncertainty in Artificial Intelligence*, pages 476–483, 2008.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive POMDPs. In *Neural Information Processing Systems*, 2008a.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *ICRA*, 2008b.

Stephane Ross, Joelle Pineau, Sebastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32: 663–704, July 2008c.

N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong, 2000.

R. Sabbadin, J. Lang, and N. Ravoanjanahry. Purely epistemic Markov decision processes. In *AAAI Conference on Artificial Intelligence*, pages 1057–1062, 2007.

S. Sanner and K. Kersting. Symbolic dynamic programming for first–order POMDPs. In D. Poole M. Fox, editor, *Twenty–Fourth AAAI Conference on Artificial Intelligence*, Atlanta, USA, July 11 – 15 2010. AAAI Press.

Cosma Rohilla Shalizi and Kristina Lisa Klinkner. Blind construction of optimal nonlinear recursive predictors for discrete sequences. In Max Chickering and Joseph Y. Halpern, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference*, pages 504–511, Arlington, Virginia, 2004. AUAI Press.

Cosma Rohilla Shalizi and Kristina Lisa Shalizi. Blind construction of optimal nonlinear recursive predictors for discrete sequences. In *Uncertainty in Artificial Intelligence*, pages 504–511, 2004.

Guy Shani, Ronen I. Brafman, and Solomon E. Shimony. Forward search value iteration for POMDPs. In *IJCAI*, 2007.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Neural Information Processing Systems*, 2010.

Hyeong Seop Sim, Kee-Eung Kim, Jin Hyung Kim, Du-Seong Chang, and Myoung-Wan Koo. Symbolic heuristic search value iteration for factored POMDPs. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI Conference on Artificial Intelligence, pages 1088–1093. AAAI Press, 2008. ISBN 978-1-57735-368-3.

Satinder Singh and Michael R. James. Predictive state representations: A new theory for modeling dynamical systems. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference*, pages 512–519. AUAI Press, 2004.

T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference*, Banff, Alberta, 2004.

V. A. Smith, J. Yu, T. V. Smulders, A. J. Hartemink, and E. D. Jarvis. Computational

inference of neural information flow networks. *PLoS Computational Biology*, 2(11), 2006.

E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes.* PhD thesis, Stanford University, 1971.

L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. J. Smola. Hilbert space embeddings of hidden Markov models. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

Thomas Stepleton, Zoubin Ghahramani, Geoffrey Gordon, and Tai Sing Lee. The block diagonal infinite hidden Markov model. In *AI and Statistics 12*, 2009.

Andreas Stolcke and Stephen Omohundro. Hidden Markov model induction by Bayesian model merging. In *Advances in Neural Information Processing Systems*, pages 11–18. Morgan Kaufmann, 1993.

M. Strens. A Bayesian framework for reinforcement learning. In *International Conference in Machine Learning*, 2000.

Peter Sunehag and Marcus Hutter. Consistency of feature Markov processes. In *ALT*, pages 360–374, 2010.

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction, 1998.

Y. W. Teh. Dirichlet processes. In *Encyclopedia of Machine Learning.* Springer, 2010.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Y.W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and*

the 44th annual meeting of the Association for Computational Linguistics, pages 985–992. Association for Computational Linguistics, 2006.

Franck Thollard, Pierre Dupont, and Colin De La Higuera. *Probabilistic DFA Inference using Kullback-Leibler Divergence and Minimality*, volume pages, pages 975–982. Morgan Kaufmann, San Francisco, CA, 2000. URL `www.sfs.nphil.uni-tuebingen.de/~thollard/Recherches/Icml2k/icml2k.html`.

N. Z. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37th Allerton Conference on Communication, Control, and Computing*, 1999.

Marc Toussaint, Laurent Charlin, and Pascal Poupart. Hierarchical POMDP controller optimization by likelihood maximization. In *Proceedings of the 24th Annual Conference on Uncertainty in Artificial Intelligence*, pages 562–570, Arlington, Virginia, 2008. AUAI Press.

J. van Gael, Y. Saatci, Y. W. Teh, and Z. Ghahramani. Beam sampling for the infinite hidden Markov model. In *International Conference in Machine Learning*, volume 25, 2008.

Jurgen Van Gael, Yee W. Teh, and Zoubin Ghahramani. The infinite factorial hidden Markov model. In *Advances in Neural Information Processing Systems 21*, 2009.

Joel Veness, Kee Siong Ng, Marcus Hutter, and David Silver. A monte carlo aixi approximation. *CoRR*, abs/0909.0801, 2009.

Chenggang Wang and Roni Khardon. Relational partially observable MDPs. In *AAAI Conference on Artificial Intelligence*, 2010.

Chong Wang, John William Paisley, and David M. Blei. Online variational inference for the hierarchical dirichlet process. *Journal of Machine Learning Research - Proceedings Track*, 15:752–760, 2011.

Tao Wang, Daniel Lizotte, Michael Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *International Conference on Machine Learning*, 2005.

Max Welling, Ian Porteous, and Evgeniy Bart. Infinite state bayes-nets for structured domains. In *Advances in Neural Information Processing Systems*, 2007.

D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2007.

J. Williams and S. Young. Scaling up POMDPs for dialogue management: The "summary POMDP" method. In *Proceedings of the IEEE ASRU Workshop*, 2005.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. Bayesian policy search for multi-agent role discovery. In *AAAI Conference on Artificial Intelligence*, 2010.

D. Wingate, N. D. Goodman, D. M. Roy, and J. B. Tenenbaum. The infinite latent events model. In *Uncertainty in Artificial Intelligence*, 2009.

David Wingate. *Exponential Family Predictive Respresentations of State.* PhD thesis, University of Michigan, 2008.

Alicia Peregrin Wolfe. POMDP homomorphisms. In *Neural Information Processing Systems RL Workshop*, 2006.

F. Wood, J. Gasthaus, C. Archambeau, L. James, and Y. W. Teh. The sequence memoizer. *Communications of the Association for Computing Machines*, 54(2): 91–98, 2011.

H. Xing-Chen, Q. Zheng, T. Lei, and S. Li-Ping. Research on structure learning of dynamic Bayesian networks by particle swarm optimization. In *Artificial Life*, 2007.

Lei Zheng and Siu-Yeung Cho. A modified memory-based reinforcement learning method for solving POMDP problems. *Neural Processing Letters*, 33:187–200, 2011. ISSN 1370-4621.