Learning approximate representations of partially observable systems

Monica Dinculescu

Master of Science

Reasoning And Learning Lab
School of Computer Science

McGill University

Montréal, Quebec

February 2010

A thesis submitted to McGill University
in partial fulfilment of the requirements of
the degree of Master of Science

Acknowledgments

I want to thank everyone who encouraged and inspired me throughout my academic career. I want to thank my mother for the lifetime of support and encouragement, without which none of my achievements would have been possible. I am forever grateful to my advisor, Doina Precup, for being exceptionally supportive, patient, and for not letting me become Bayesian. She taught me just about everything I know about reinforcement learning, being a grown up, and surviving academia. I could not have asked for a better teacher and mentor. Prakash Panangaden encouraged me to think for myself, to prove theorems, and instilled in me a lasting appreciation for proper curry.

Many people have given me advice and assistance. The members of the Reasoning and Learning Lab at McGill University are a great group and have proven that computer scientists are fun to hang out with. I especially thank Jordan Frank for his helpful input and advice, and Jesse for the infinite supply of coffee. Lastly, I thank Liam, without whom this would not have been as fun.

Abstract

Learning agents that interact with complex environments often cannot predict the exact outcome of their actions due to noisy sensors or incomplete knowledge of the world. Learning the internal representation of such partially observable environments has proven to be a difficult problem. In order to simplify this task, the agent can choose to give up building an exact model which is able to predict all possible future behaviours, and replace it with a more modest goal of predicting only specific quantities of interest.

In this thesis we are primarily concerned with ways of representing the agent's state that allows it to predict the conditional probability of a restricted set of future events, given the agent's past experience. Because of memory limitations, the agent's experience must be summarized in such a way as to make these restricted predictions possible. We introduce the novel idea of history representations, which allow us to condition the predictions on "interesting" behaviour, and present a simple algorithmic implementation of this framework. The learned model abstracts away the unnecessary details of the agent's experience and focuses only on making certain predictions of interest. We illustrate our approach empirically in small computational examples, demonstrating the data efficiency of the algorithm.

Abrégé

L'apprentissage d'agents artificiels confrontés à un environnement complexe est souvent difficile dû à leur incapacité à prédire le résultat de leurs actions et à une description incomplète du système. L'apprentissage d'une représentation interne d'un environnement partiellement observable est particulièrement malaisé. Afin de simplifier cette tâche, l'agent peut, plutôt que de constuire un modèle exact capable de prédire tout comportement futur, chercher à ne prédire que quelque phénomènes en particulier. Dans cet ouvrage, nous nous intéressons à la question de représenter l'état du système de manière à predire la probabilité conditionnelle d'un ensemble restreint d'événements, étant donné l'expérience précédente de l'agent. Dû à une limite quant à la capacité mémoire de l'agent, cette expérience doit être résumée quant à rendre atteignable cet ensemble restreint de prédictions. Nous proposons ici l'idée d'employer des représentations basées sur un historique afin de produire des prédictions conditionnelles à des comportements "intéressants". Nous développons cette idée par le bias d'un algorithme. Nous illustrons notre approche de manière empirique à travers de simples exemples computationnels, démontrant ainsi l'efficacité de l'algorithme quant à la quantitée de données requises.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Reasoning about the consequences of actions in a stochastic domain is a desirable feature of intelligent agents. These environments are usually partially observable, meaning that the agent never observes its actual situation in the world, or *state*, and must infer it through observations received from the environment. Sensors are often noisy or faulty, and thus agents have an incomplete knowledge of the world. In order to be able to interact with their environment, they rely on an internal representation that allows them to predict the outcome of their actions. However, learning the internal representation of such partially observable environments has proven to be a difficult problem.

Consider a robot that navigates around a room trying to predict what it will see next, without having any knowledge of the map of the room. The agent interacts with the environment by taking an action and receiving an observation, without knowing its actual state. In order to be able to make any reliable prediction about the future, it must be able to estimate this state based on its experience with the world.

In complex environments, where the number of actions and possible observations is high, the space of model parameters can be very large and the agent may be computationally incapable of updating them.

Traditionally, the framework of choice for modelling this system has been provided by Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al., 1995). These models assume that the dynamics of the environment can be explained through a number of hidden, or latent states. It is well understood how to plan good courses of action in a POMDP, if the model of the environment is given. If the model is not known, the traditional solution is to use Expectation Maximization (EM) to acquire it from data. This approach has been demonstrated in several practical applications (Shatkay and Kaelbling, 1997), such as robot navigation. However, the empirical evidence to date suggests that this approach only works if one already starts with a good initial model. If the initial model is imprecise, EM typically ends up in a bad, but locally optimal, solution.

In order to simplify the problem, the agent can choose to only be interested in a set of observations at a time. For example, if the robot's battery is low, it could be interested exclusively in finding the power charger. The model that only makes predictions about this observation is much simpler than the full model of the world, and thus the agent needs less data to learn it.

In this thesis we are primarily concerned with ways of representing the agent's state that allows it to predict the conditional probability of future events, given the agent's past experience.

1.1 Predictive State Representations

The state of a POMDP is artificial and requires a prior understanding of the world. From the agent's perspective, the knowledge of its actual x-y coordinate in the world is less important than the ability to know what will happen next. Recent work on predictive state representations (PSRs) (Littman et al., 2002) is aimed at addressing this problem. Their proposed representation is based entirely on predicting the conditional probability of sequences of future observations, conditioned on future sequences of actions and on the past history. Because there are no hidden states in the model, in principle, such a representation should be easier to learn from data. Linear PSRs, which have been explored most, are an exact model of the system in the sense that they can predict the probability of any future, given any past history. Similar exact models, have been provided in other work, for different types of partial observability (Rivest and Schapire (1994), Hundt et al. (2006)).

1.2 Contributions

The common thread in all this work is the idea that one has to give up building an exact model, which is able to predict all possible future behaviours, and replace it with a more modest goal of predicting only specific quantities of interest. The approaches differ greatly in terms of the computational mechanisms involved.

In this thesis, we attempt to provide a unified way of thinking about this problem. First, we insist that the only goal of the learned representation is to maintain particular types of predictions (e.g., predictions about specific observations, about the

total rewards that the agent might obtain, etc.). Secondly, because of memory limitations, the agent's experience must be summarized in such a way as to make these restricted predictions possible. We present two agent state representations that illustrate these ideas: the first requires the agent to be given the "interesting" features of history that are useful for prediction, and the second in which these features are learned automatically, using the agent's recent experience. In both cases we provide a learning algorithm that successfully learns accurate models from small amounts of data, even when the data is generated form a non-Markovian or a continuous-state system.

1.3 Outline

We begin by giving the background on modelling dynamical systems in Chapter 2. We review two methods for representing these systems that are fundamentally different in their representations of the agent's internal state.

In Chapter 3 we formalize the concept of the agent's interest in predicting specific aspects of the future and discuss how we can construct an approximation of the system that only predicts these tests. Furthermore, to abstract over unnecessary details of the agent's experience, we introduce the notion of history representations, a mechanism that computes the "usefulness" of a history in making the predictions of interest.

In Chapter 4 we present our initial approximate agent state representation that requires the agent to have prior knowledge of how the histories representations are constructed, given the tests of interest. We provide two sets of experimental results

that demonstrates the effectiveness of the representation in making the predictions of interest.

In Chapter 5 we expand on the previous representation, by automatically constructing the history representations based on the agent's short term experience. We motivate the construction through the notion of temporal coherence, which suggests that the agent's short term memory is enough to make good local predictions about the future. We also present an initial intuition on how these models can be used in control tasks, to learn an optimal behaviour policy.

Finally, in Chapter 6 we conclude and suggest directions for future work.

Chapter 2
Background

Probabilistic models are necessary for decision making in complex, realistic environments. Agents often cannot predict the exact outcome of their actions due to noisy sensors or incomplete knowledge of the world. In this chapter we will discuss learning a model of a dynamical system from data, and formalize what we mean by *state* from the agent's point of view. We review two methods for modelling dynamical systems that are fundamentally different in their representations of state. Partially Observable Markov Decision Processes (POMDPs) are a popular class of models which assume an underlying set of hidden states. Predictive State Representations (PSRs) instead define the state as a set of statistics about the future. We discuss both of these approaches in the context of making predictions about future events.

2.1 Dynamical Systems

We consider the case of an agent interacting with an environment at discrete time steps, by performing actions from a discrete set $\mathcal{A}$ and receiving observations from a set $\mathcal{O}$, as illustrated in Figure 2.1.

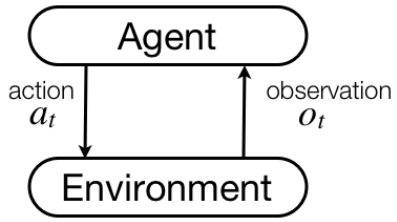

Figure 2.1: An agent interacting with a dynamical system. At time step t, the agent takes an action a_t, to which the environment responds with an observation o_t and a a reward r_t

The dynamical systems on which we focus exhibit the following characteristics:

- *Stochasticity* in both the actions and observations. This means that the agent's actions will not always have the expected outcome (a robot with a broken wheel may attempt to move forward, without actually doing so), and the environment will not always respond to actions in the same way.

- *Partial observability*, which means that the agent does not have complete knowledge of the environment. For example, a robot will rarely observe its actual coordinates in the world, but could observe walls or obstacles. The walls do not convey all the possible information about the agent's situation, and thus cannot be used alone to make decisions - both a kitchen and a living room will have walls, but this does not mean that the agent should behave the same in both. Conversely, in fully observable environments, the agent will know with certainty in which room it is, and thus can make a decision based on its current location alone.

Within this framework, the agent can be thought of as trying to solve one of two problems: understanding how the environment works, and learning how to act within it, once this understanding is formed. We leave the second question for future work. The main goal of this thesis is the first the problem: that of creating the internal representation of the environment. This representation must be grounded in the agent's experience, with little prior information about the environment, and should be able to make predictions about future observations.

2.2 Agent State

What does it mean for an agent to learn a representation of a system? It means that it can give an accurate estimate about what will happen in the future that is consistent with the behaviour of the system in the past. In order to be able to talk about talk about the past and the future from the agent's point of view, we will define a *history* as a sequence of actions and observations that have happened in the past, and a *test* as a sequence that will happen in the future.

Definition 2.2.1. *A* history*, denoted by h_τ, is an action-observation sequence received up to time step τ: $h_\tau = a_0 o_0 a_1 o_1 \dots a_\tau o_\tau$.*

Definition 2.2.2. *An action-observation sequence starting at time $\tau + 1$ is called a* test*, t_τ (Littman et al., 2002).*

Given a test $t_\tau = (a_{\tau+1} o_{\tau+1} \dots a_{\tau+k} o_{\tau+k})$, we denote by $\omega(t_\tau)$ the sequence of observations of the test, $(o_{\tau+1}, \dots o_{\tau+k})$ and by $\sigma(t_\tau)$ the sequence of actions of the test, $(a_{\tau+1}, \dots a_{\tau+k})$. We will refer to the latter as the *skeleton* of a test.

If the system is non-deterministic, the probability of a test occurring is not certain. For example, if an action fails to execute, due to noise or malfunction in the agent, then the observation experienced by the agent will change. Thus, we can say that an agent understands and can model a system if it can predict whether or not a test will succeed given a certain history.

Definition 2.2.3. *The* prediction *for test t given history h, $p(t|h)$, is defined as the conditional probability that $\omega(t)$ occurs, if the sequence of actions $\sigma(t)$ is executed (Littman et al., 2002).*

$$p(t|h) = P(\omega(t)|h, \sigma(t)).$$

In discussing models of dynamical systems, we will refer to the agent's *state* as the knowledge the agent has about the environment at that time, which allows it to make predictions about the immediate future (i.e., the next state). Because the environment is partially observable, the agent does not have access to the true state of the world. This knowledge comes directly from the agent's experience, i.e., the histories it has observed up to that time. One way in which we can then describe the agent's state is as a summary of history.

An important aspect of the agent's state is that it must be a *sufficient statistic* of the history, in the sense that this state alone is enough to make predictions about the future, and no other information is needed. The agent must also be able to *update* the state over time, as it gathers new experience.

There are two general approaches for learning a model that can make such predictions from data:

1. *partially observable Markov decision processes* assume a hidden structure that generates the observations and learn the internal transition probabilities and observation probabilities that match the behaviour of the system

2. *predictive state representations* assume that there is a set of predictions of future action-observation sequences that is sufficient to compute the prediction for all possible action-observation sequences

We now present each of these representations in detail.

2.3 Partially Observable Markov Decision Processes

A partially observable Markov decision process, as illustrated in Figure 2.2, is a general framework for decision making under uncertainty. Formally, a POMDP is a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, b_0)$, where $\mathcal{S}$ is a finite set of hidden (or latent) states in the environment, $\mathcal{A}$ is a finite set of actions the agent can take, $\mathcal{O}$ is a set of observations that the environment emits, and b_0 is the initial distribution over the hidden states.

The probability distribution $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition model; the probability of transitioning from the hidden state s to s' after taking action a, and is given by

$$\mathcal{T}(s, a, s') = \mathbb{P}(s_{\tau+1} = s' | s_\tau = s, a_\tau = a).$$

The probability $\Omega : \mathcal{S} \times \mathcal{A} \times \mathcal{O}$ of receiving an observation from the environment is given by the probability distribution

$$\Omega(s, a, o) = \mathbb{P}(o_\tau = o | s_\tau = s, a_{\tau-1} = a).$$

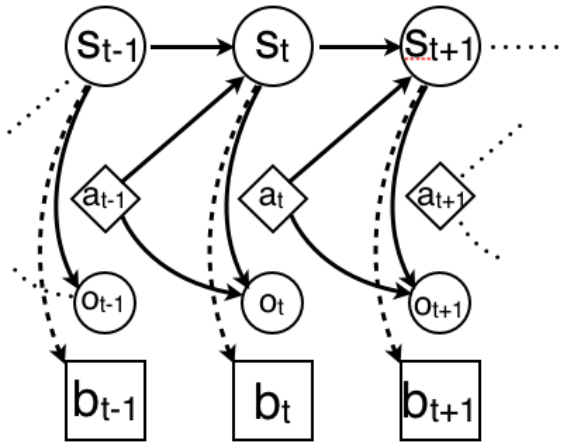

Figure 2.2: Graphical view of a partially observable Markov decision process (POMDP). At each time step τ, the agent executes an action a_τ and receives an observation o_τ. The agent maintains a belief state b_{h_τ} which is a vector of probabilities of being in each state at that time.

The system is partially observable, and thus the agent must maintain a special distribution, known as a *belief state*, that estimates the probability of being in any state at a given time. We denote the probability of an agent being in state s after having observed the history h by $b_h(s)$. After taking a new action a, and observing o, the belief can be updated as follows:

$$b_{hao}(s') = \frac{\Omega(s', a, o) \sum_{s \in \mathcal{S}} b_h(s) \Omega(s, a, s')}{\sum_{s' \in \mathcal{S}} \mathcal{T}(s', a, o) \sum_{s \in \mathcal{S}} b_h(s) \mathcal{T}(s, a, s')},$$

with the initial belief being denoted by b_0.

It has been shown that the belief state is sufficient for optimal decision making (Sondik, 1971). Thus, the agent's state in this case is the belief state vector itself. As seen above, this state can be updated as the agent interacts with the environment. Secondly, this state can be used to make conditional predictions about

the future. The probability of a test t given a history h is:

$$\begin{aligned}
\mathbb{P}(t = a_0 o_0 \dots a_n o_n | h) &= \sum_{s_0 \in \mathcal{S}} b_h(s_0) \times \mathbb{P}(a_0 o_0 \dots a_n o_n | s_0) \\
&= \sum_{s_0 \in \mathcal{S}} b_h(s_0) \times \sum_{s_n} \mathbb{P}(s_n, o_0 \dots o_n | s_0, a_0 \dots o_n) \\
&= \sum_{s_0 \in \mathcal{S}} b_h(s_0) \times \sum_{s_1} \mathbb{P}(s_1, o_0 | s_0, a_0) \sum_{s_n} \mathbb{P}(s_n, o_1 \dots o_n | s_1, a_1 \dots a_n) \\
\dots &\quad \sum_{s_0 \in \mathcal{S}} b_h(s_0) \prod_{i=0}^{n} \sum_{s_i \in \mathcal{S}} \mathbb{P}(s_{i+1} | s_i, a_i) \mathbb{P}(o_i | s_i, a_i) \\
&= \sum_{s_0 \in \mathcal{S}} b_h(s_0) \prod_{i=0}^{n} \sum_{s_i \in \mathcal{S}} \mathcal{T}(s_i, a_i, s_{i+1}) \Omega(s_i, a_i, o_i)
\end{aligned}$$

While decision making and planning using POMDPs has been extensively studied, learning the model itself has not been as thoroughly examined. One approach that assumes little prior knowledge about the system is an extension of the Baum-Welch algorithm, originally developed for hidden Markov models (Rabiner, 1990). Baum-Welch is an Expectation Maximization (Dempster et al., 1977) algorithm that estimates both the transition and observation probabilities from action-observation sequences.

We use the following notation when describing the algorithm. Given a history $h = (a_0 o_0, \dots, a_n, o_n)$, we use h_τ to represent the first τ action-observation pairs (i.e., $h_\tau = (a_0 o_0, \dots, a_\tau, o_\tau)$), and $\bar{h}_\tau$ to represent the remaining action-observation pairs of h, starting after time τ (i.e.,$\bar{h}_\tau = (a_{\tau+1} o_{\tau+1}, \dots, a_n, o_n)$).

Given a history $h = (a_0 o_0, \dots, a_n, o_n)$, the Baum-Welch algorithm, visually illustrated in Figure 2.3, updates the POMDP parameters as follows:

1. At every step τ in the history h, we first calculate:

- The probability of being in some state s_i after taking the actions in τ actions of h and seeing the first τ observations of h. This is denoted by

$$\alpha_{h_\tau}(s_i) = b_{h_\tau}(s_i), \forall s_i \in \mathcal{S}$$

- The probability of starting in some other state s_j, taking the remaining $n - \tau$ actions of h and seeing the $n - \tau$ observations of h. This is denoted by $\beta_{\bar{h}_\tau}(s_j)$. For every action a that can be taken in s, this is equal to

$$\beta_{\bar{h}_\tau}(s_j) = \frac{\sum_{s' \in \mathcal{S}} \mathcal{T}(s_j, a, s')\Omega(s', a, o)\beta_{\bar{h}_{\tau+1}}(s')}{\sum_{s' \in \mathcal{S}} \mathcal{T}(s', a, o) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s')\alpha_{h_\tau}(s)}, \forall s_j \in \mathcal{S},$$

  and 0 for all other actions.

2. Calculate the probability that at time τ you have transitioned from state s_i to s_j, and that you have observed the rest of h, as described above. This can be seen as bridging the two probabilities described above, and is denoted by $\gamma_\tau(s_i, s_j)$:

$$\gamma_\tau(s_i, s_j) = \alpha_\tau(s_i)\mathcal{T}(s_i, a_\tau, s_j)\Omega(s_j, a_\tau, o)\beta_{\tau+1}(s_j)$$

3. Re-estimate the state transition probabilities $\mathcal{T}(s, a, s')$ by counting the number of times the agent was in a state s, executed action a, and moved to state s, versus the number of times you were the agent was in the state s and executed action a (i.e., regardless of where the agent ended up).

4. Re-estimate the observation probabilities $\Omega(s, a, o)$ by counting the number of times the agent saw observation o in state s, after taking action a, versus the number of times the agent was in state s (i.e., regardless of what the agent observed)

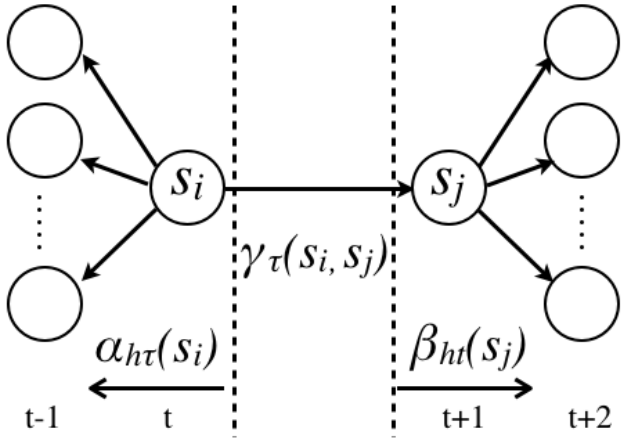

Figure 2.3: Illustration (Rabiner, 1990) of the Baum-Welch algorithm

A second way of learning POMDPs uses Bayesian reinforcement learning (Dearden et al., 1999) to maintain a distribution over the model parameters. This has the benefit of exploring the space of possible models and picking the one that is most likely to have generated the agent's experience. However, it is not always clear what the initial prior over the parameters should be and in practice, learning the representation requires a large amount of data.

2.4 Predictive State Representations

One crucial disadvantage of the previous approach is that learning the POMDP parameters is often sensitive to the accuracy of the initial assumptions (Shatkay and Kaelbling, 1997). If the initial model is imprecise (for example, if the number of hidden states has been underestimated), EM typically ends up in a bad, but locally optimal, solution. This is also true in the case of the Bayesian RL approach, if the

initial prior is very far from the true distribution. Secondly, the state representation requires prior information such as the number of hidden states, that are not necessarily easily specified. For example, in a dialogue manager, it's hard to say what are the underlying states generating answer to questions, however, it is much easier to predict the conditional probability of answer, given a specific question (e.g., the probability of the answer being "It's raining", if the question is "How is the weather?").

An alternative model of a dynamical system is that of Predictive State Representations (PSRs), recently introduced by Littman et al. (2002), that generalizes previous work by Rivest and Schapire (1994). In this case, the internal state is represented as a set of statistics about future tests. This is more desirable than the POMDP formulation, as the representation is constructed from observable data, and not around the artificial notion of hidden state.

System-dynamics matrix

We begin our overview of PSRs by introducing the *system-dynamics matrix* (Singh et al., 2004), which is a conceptual representation of a dynamical system. Let T be the set of all tests and H the set of all histories observed from the system. Given an ordering (e.g. lexicographic) over H and T, one can define the matrix $\mathcal{D}$ (shown in Figure 2.4), whose rows correspond to histories, and whose columns correspond to tests. An entry in the matrix is the prediction of a specific test, given a history.

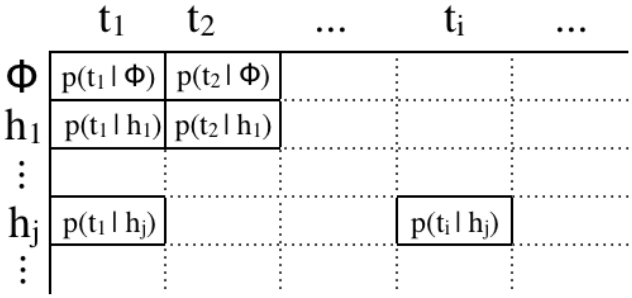

Figure 2.4: The system-dynamics matrix $\mathcal{D}$. The rows correspond to histories, and columns correspond to tests. An entry in the matrix is the prediction of a test given a history

It is important to note that this matrix is not a *model* of the system, but fully specifies the system itself. It contains every possible trajectory that can be observed in the system and can make any conditional prediction about the future.

Singh et al. (2004) have shown that, even though this matrix is infinite, if histories and tests are generated from a POMDP model, the matrix has finite rank. Formally, if the system-dynamics matrix $\mathcal{D}$ has rank k, then there exist k linearly independent columns and rows. The k tests associated with the linearly independent columns are called the *core tests*. Analogously, one can define the *core histories*. The set of core tests is not necessarily unique; however, for simplicity, we can assume that the set with the shortest tests is chosen. We can now form a subset of the system-dynamics matrix, illustrated in Figure 2.5, where the rows are the still the histories in H, but the columns are the k core tests.

Linear PSR

The state representation of given a PSR is the *prediction vector* of a history, which is the vector of predictions for all the core tests. Formally, given a history h and a

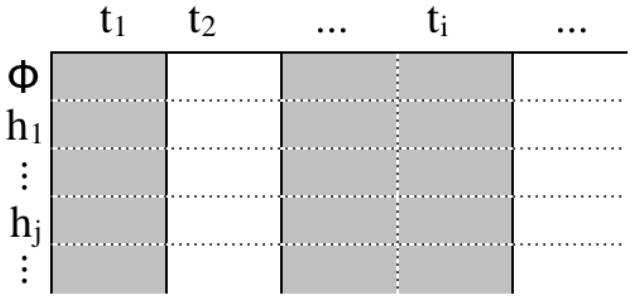

Figure 2.5: The set of core tests Q is a set of linearly independent columns of the system-dynamics matrix. The matrix formed by them is $\mathcal{D}(Q)$, shown in grey

set of core tests Q, the prediction vector is

$$p(Q|h) = [P(q_1|h), P(q_2|h), \ldots, P(q_n|h)].$$

Because the core tests are the linearly independent columns of $\mathcal{D}$, then any column can be written as a linear combination of the columns in $\mathcal{D}(Q)$. Formally, for any test t, and history h, there exists a weight vector m_t such that

$$p(t|h) = p(Q|t)^T m_t.$$

For any history h, the state of the PSR is the set of core tests Q and their predictions $p(Q|h)$, shown in Figure 2.5. These predictions can be thought of as answers to questions of the form "If I take this sequence of actions, will I see that sequence of observations?". This state is a sufficient statistic for the system, as the agent can predict any test as a linear combination of some of the predictions in $p(Q|h)$. Once the agent takes a new action a and receives a new observation o, the prediction for each of the core tests $q_i \in Q$ can be updated according to:

$$p(q_i|hao) = \frac{p(aoq_i|h)}{p(ao|h)} = \frac{p(Q|h)^T m_{aoqi}}{p(Q|h)^T m_{ao}}.$$

From this formulation it is clear that we do not need to re-compute the weights for every possible test in the system: rather, we only need to compute the weights m_{ao} for the one-step tests ($\forall a \in \mathcal{A}, \forall o \in \mathcal{O}$), and$m_{q_iao}$, the weights of the one-step extensions of the core tests ($\forall q_i \in Q$) to be able to make any arbitrary prediction.

These updates can be combined into a single update for Q by defining the matrix M_{ao}, where the ith column is m_{aoq_i} (i.e., the weight of the one-step extensions of the core test q_i). The state update is thus recursively defined as:

$$p(Q|hao) = \frac{p(Q|h)^T M_{ao}}{p(Q|h)m_{ao}}$$

It can be shown (Littman et al., 2002) that the prediction for an arbitrary test t, given a history is

$$p(t|h) = \frac{p(Q|\emptyset)^T m_{ht}}{p(Q|\emptyset^T)m_h},$$

where $\emptyset$ is the empty history.

Finally, Singh et al. (2004) have shown that PSRs are equivalent to POMDPs, in the sense that if a dynamical system can be modelled with a POMDP containing k latent states, then it can also be modelled by a PSR with at most k core tests.

While this approach is theoretically elegant, it is problematic in practice. If the data is not coming from a Markovian underlying system, for example, or if the underlying system has continuous states, the rank for the systems dynamic matrix may be infinite. Even if the environment is Markovian but has many observations and actions, building a sufficient portion of the system-dynamics matrix, to a sufficient degree of accuracy, may not be feasible.

Some learning algorithms have been proposed for PSRs, with mixed success (Singh et al. (2003), Bowling et al. (2006), McCracken and Bowling (2006)). All

these approaches require a lot of data to build a good representation. The reason is that at the heart of PSR construction lies the computation of a "system dynamics" matrix (Singh et al., 2004), which contains conditional probabilities of future action-observation sequences given past sequences (i.e. *histories*). Then, one needs to determine the linearly independent columns of this matrix. However, this operation is not numerically stable, so it only works well if a lot of data has been accumulated, and the estimates of the probabilities are very precise. In environments with many observations and actions, computing all these probabilities is too expensive. Also, since the entries in the matrix are estimated from data, in practice the values are noisy. Noisy columns are often linearly independent, thus resulting in an incorrect set of core test (Jaeger, 1998) This has lead to a wave of recent work on learning approximate predictive representations.

2.5 Other Predictive Models

We now introduce other models of dynamical systems which represent the agent's state as a set of predictions about the future.

2.5.1 Temporal-Difference Networks

Temporal-Difference Networks, introduced by Sutton and Tanner (2005) are similar to PSRs in the sense that they also make predictions of future tests, given past experience. Unlike PSRs, however, predictions are compositional, meaning that the prediction for a test depends on the predictions of smaller tests.

The model is composed of two networks of nodes: the question network, which phrases the questions of interests and the answer network which tells you how the answers are computed. In the question network, nodes represent the *questions* about future events, i.e., the tests that the agent is interested in predicting. For example, a node could contain the question "If the my next action is to move forward, what is the probability that I will hit a wall?". The inputs to these nodes are the current observation, the previous action, and the previous step predictions. A second network, namely the *answer network* determines how the actual predictions as each node are calculated using temporal difference learning.

One drawback of TD-Nets is that the optimal structure for the question network needs to be specified manually, and not much work has been done towards discovering this structure from data. Secondly, in big networks, temporal difference learning can give a non-convergent solution, or can converge to very bad, locally optimal solutions.

2.5.2 Local Models

Local models, introduced by Talvitie and Singh (2008) are a set of models that make predictions about a restricted set of futures. These tests of interest are explicitly specified, with the restriction that every test of interest is a *union tests* (i.e. a set of tests t_i such that no prefix of t_i is a union test). Union tests can be thought of as questions about whether or not any one of a set of observations will occur.

Furthermore, because not all histories are necessary in making accurate predictions of tests of interest, they also introduce the notion of histories of interest. Using these two sets, they can build a *local model* that predicts the tests of interest. A

collection of such local models can be used to answer both more complex questions (by predicting the union of several tests of interest), or finer-grained questions (using intersections of sets of tests of interest).

The main constraint of this approach is that both the histories and tests of interest must be specified explicitly. In Chapter 3 we generalize this notion by providing a formalism that allows histories of interest to be learned from the agent's experience.

2.5.3 Other Approximate Models

Rosencrantz et al. (2004) provide an approximate algorithm for computing approximate predictive representations, based on the idea of maintaining probabilities not over outcomes of tests, but over linear combinations of these outcomes. Their algorithm is efficient, but does not have the same theoretical guarantees as PSRs.

James and Singh (2005) modify the PSR structure to include both memories of the past observations, as well as predictions about the future. The resulting models can be exact, but in practice, they become approximate.

Wolfe et al. (2008) provide approximate PSR models under the assumption that the observations are multivariate and different parts of the observation vector can be assumed to be conditionally independent.

Still (2009) proposes the use of information-theoretic criteria to summarize the system's history in a finite number of states, in such a way as to maximize the predictive information retained.

Chapter 3
Making Predictions That Matter

Building a model that has a perfect understanding about everything in the world is a very difficult task. Due to computational limitations an agent will rarely be able to store its entire experience, and thus will not be able to build enough of the system dynamics matrix to find a correct and accurate linear PSR. For example, in environments with many observations and actions, the space of observable histories and tests is very large. Given that the agent must observe each of the possible history-test pairs enough times to compute an accurate estimate of the corresponding entries in the system dynamics matrix, it is clear that this endeavour may require a lot of data; hence, the agent might have to wait for a long time before it can make reasonably accurate predictions.

In order to scale PSRs to work with large systems and limited memory resources, one must make an approximation, at the cost of either accuracy or completeness. The approach taken in this thesis is that of restricting the set of predictions the agent has to make at any given time. This will allow us to learn a model that is approximate in the sense that it can make accurate predictions about a subset of the system, but makes no predictions about anything else.

In the first part of this chapter we introduce a general framework that allows us to restrict the set of predictions the agent must make, and construct a fragment of the system-dynamics matrix containing only these predictions of interest.

Since we are no longer trying to learn a full model of the world, there is no reason to think that all of the agent's experience is useful in making this limited set of predictions. However, specifying which histories are "meaningful" to the agent is not as straightforward. To address this problem, in section 3.3 we introduce the idea of history representations, a mechanism that allows us to abstract over the irrelevant aspects of the agent's experience.

3.1 Specifying interest in the future

The idea of only wanting to answer questions about a subset of the world at a time is grounded in human behaviour. Humans, who do not suffer from the resource limitations of an agent, tend to answer complex questions by combining the answers of smaller, more detailed questions. For example, "What will happen tomorrow?" is a difficult question to answer, as there are many variables and external factors in the world that may affect the outcome. However, questions such as "Will I go to work?", "Will I buy coffee?", "Will I take the bus?" are concerned with very specific parts of the world, and gathering information about them is easy. Furthermore, combining the answers to these smaller questions gives us a fairly good answer to the abstract question as well, that was obtained much easier and quicker than we would have otherwise. If we think of answers to these questions as predictions of tests, then we

are essentially building small predictive models that are only interested in a subset of future tests at a time.

Much of the recent approximate PSR work has been driven by this idea, with the agent's interest being specified, for example, as subsets of the tests of interest, linear combinations of tests, etc. (Talvitie and Singh, 2008). By limiting the set of predictions, the agent can afford to explore the environment more, as it needs to store a smaller subset of its experience to build a model. Each of these models, accurate with respect to a set of tests of interest, can be combined to answer a more comprehensive set of questions about the world.

We begin by introducing the notion of *probes* that forms the basis of our work.

Definition 3.1.1. *A* **test probe***, f, is a mapping, $f : O^* \to \mathbb{R}$.*

Definition 3.1.2. *The* **probed prediction** *of a test t given a history h and a probe f is defined as the expected value of $f(t)$ given h:*

$$p_f(t|h) = p(t|h)f(\omega(t)),$$

where as before, $\omega(t)$ is the sequence of observations in the test t.

Note that the probe is defined only on the observation sequence, in order to ensure that the prediction can still be properly conditioned on the sequence of actions for the test, $\sigma(t)$ (and thus independent of the agent's policy).

Definition 3.1.3. *The* **prediction for a sequence of actions** $a_{\tau+1} \ldots a_{\tau+k}$ *given a history h and a probe f is defined as the expected value of f over all tests t having the action sequence as a skeleton:*

$$p_f(a_{\tau+1} \ldots a_{\tau+k}|h) = \sum_{t:\sigma(t)=a_{\tau+1}\ldots a_{\tau+k}} p_f(t|h)$$

This is a very general formulation through which we can specify the tests of interest to the agent, while allowing previous work to be derived as special cases. In particular, a full linear PSR will have f equal to 1 for all tests. The test probe as defined above can be used as a discriminative filter: if the agent wants to ignore all observations except the last one in a test, f can be defined as 1 for all tests ending in the desired observation, and 0 for all tests with the same skeleton but not ending in the desired observation.

This approach generalizes in a straightforward way to union tests as well (Talvitie and Singh, 2008). If the observations consist of a vector, parts of which are conditionally independent of others (Wolfe et al., 2008), then multiple test probes can be used, one for each part of the observation vector that can be modelled independently. Similarly, if the agent is only interested in predicting a linear combination of the form $\sum_i w_i p(t_i)$, we can define $f(t_i) = w_i, \forall i$ and $f(t) = 0$ for all other tests; this corresponds to transformed PSRs (TPSRs), defined by Rosencrantz et al. (2004).

An important special case is the one in which the observations contain rewards; in this case, f could be defined as the future sum of discounted rewards, and the predictions associated with action sequences would be akin to value functions for sequences of actions.

3.2 Learning Predictions of Interest

Suppose that instead of containing the values $p(t|h)$, the system-dynamics matrix would contain the *probed predictions* $p_f(t|h)$, where f is a given probe. If two rows

of this matrix, corresponding to histories h_1 and h_2, would be equal, then the corresponding histories yield the same predictions for all tests, and they can be collapsed into one, aggregate class. Similarly, if two tests t_1 and t_2 had the same predictions for all histories, they could be collapsed into an equivalence class.

Figure 3.1 illustrates this idea of equivalent histories and tests. Assuming that the environment is discrete and deterministic, if the agent takes two steps forward, then it will reach the orange wall, and thus could predict that given the history "↑↑" (the observations in between the actions are assumed to be "`white`"), the probability of seeing orange is 1. However, the agent could first turn around 360°, and then take two steps forward, again reaching the orange wall. Thus, given the history "→↓←←↑→↑↑", the probability of seeing orange is also 1. Because these histories make the same predictions, there is no advantage in remembering them both, and thus can be considered identcal. Similarly, once the agent has reached the orange wall, moving forward will not advance its position, and it will continue observing "`orange`".

For the same reason, the tests "↑ `orange`", and "↑ `orange` ↑ `orange`" are also equivalent, as they always occur with the same probability.

Collapsing histories and tests in this way gives rise to a smaller matrix. However, if the matrix is learned from actual data, exact equality in the expectation estimates is unlikely. Hence, we will allow for small errors in these predictions.

We can partition the set of all possible tests T into a set of clusters T', such that for each cluster $T_i \in T'$,

$$\forall t_1, t_2 \in T_i, \forall h \in H, |p_f(t_1|h) - p_f(t_2|h)| \leq \epsilon_f, \tag{3.1}$$

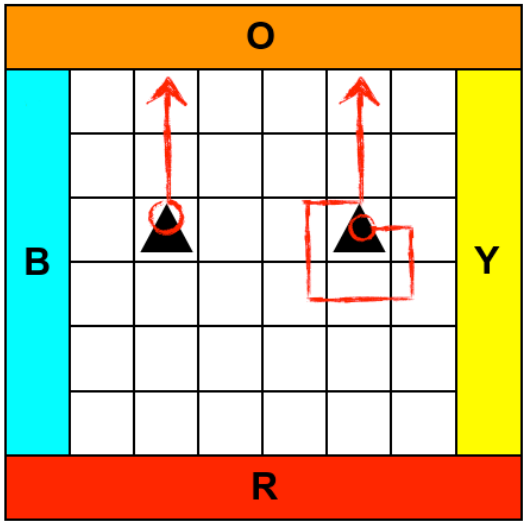

Figure 3.1: An example of equivalent histories and tests. Given that the f probe only considers tests containing `orange`, both histories shown in red are equivalent, as they end up in the same place, relative to the orange wall. Also, once the agent has hit the orange wall, further going forward will not change this, meaning that tests of the form "↑`orange`" are equivalent to tests of the form "↑`orange`(↑ `orange`)*"

where ϵ_f is a small real value (chosen by the experimenter). Note that the mapping of tests to such clusters is not unique. Since every pair of tests in a cluster is similar, any single test within the cluster can be used to uniquely identify a specific cluster.

We associate with each cluster T'_i a *representative test* $t'_i \in T'_i$, which can be chosen arbitrarily (e.g., the lexicographically shortest member). The set T' can then be considered to contain only the representative tests from each cluster. The prediction for a cluster T_i and history h is:

$$p_f(T_i|h) = \frac{1}{|T_i|} \sum_{t \in T_i} p_f(t|h)$$

Note that because of the way in which the clusters are constructed, all elements of a cluster will be within ϵ_f of this prediction.

We define the *probed prediction vector of a history* to be the vector of probed predictions for all the tests in the set of interest, T':

$$p_f(T'|h) = [p_f(t'_1|h), p_f(t'_2|h), \ldots, p_f(t'_k|h)],$$

where k is the number of clusters in T'.

Similarly, we can partition the set of all possible histories H into a set of clusters H', such that for each cluster $H_i \in H'$

$$\forall h_1, h_2 \in H_i, \forall t \in T, |p_f(t|h_1) - p_f(t|h_2)| \leq \epsilon_g, \tag{3.2}$$

where ϵ_g is a parameter. As before, a *representative history* $h'_i \in H'_i$ can be chosen, with H' being the set of all clusters. Again, by definition, all histories in a cluster will be within ϵ_g of this value.

The *probed prediction for a history cluster* H'_i is the average value of the probed prediction vectors of the histories in the class:

$$p_f(T'|H'_i) = \frac{1}{|H'_i|} \sum_{h_i \in H'_i} p_f(T'|h_i).$$

The vector defined above gives the probed predictions for the representative tests in T'. Given a new history that can be mapped to a cluster H'_j, we can get the probed prediction for a test of interest t'_i by looking at the ith column of the vector $p_f(T'|H'_j)$. The simple averaging is taken here because without any a priori information, all histories should be considered equally likely. If the agent were to behave in a way biased by a particular policy, the averaging could be done based on the data received, so more likely histories would naturally be given more weight. We can collect all the probed predictions for H' in a set denoted $p_f(T'|H')$.

One possibility for obtaining T' and H' is to construct a fragment of the system dynamics matrix, then use Equations 3.1 and 3.2 to collapse its rows and columns. The following theorem gives an error guarantee for this approach.

Theorem 3.2.1. *Suppose T' respects Equation 3.1 and H' respects Equation 3.2. Then the maximum prediction error will be at most $2(\epsilon_f + \epsilon_g)$.*

Proof: Figure 3.2 depicts the main idea of the proof. The large dots represent the histories, and the small dots are the predictions for specific tests. Within each circle, i.e. each cluster of tests, all predictions are ϵ_f away from the center, and thus the difference in prediction is at most the diameter of the circle, which is $2\epsilon_f$. Similarly, histories are only collapsed in the same history cluster H_i if their predictions of the tests of interest are ϵ_g away from the center. So, within each cluster of histories, any two predictions will have a total error of at most $2(\epsilon_f + \epsilon_g)$ $\square$.

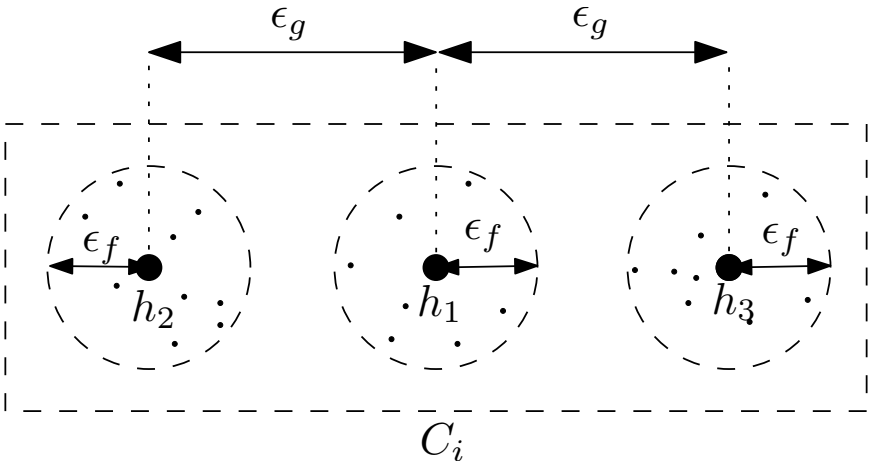

Figure 3.2: The prediction error of the approximate system-dynamics matrix is at most $2(\epsilon_f + \epsilon_g)$

3.3 History Representations

In the rest of the paper, we will assume that f is given, and the goal is to learn the probed predictions p_f from data. Given the definition, the simplest approach is to use *discriminative learning*, where the history (or the observation-action sequence) is treated as an input and the output to be predicted is p_f. Existing results (Ng and Jordan, 2002) suggest that discriminative learning may have advantages compared to generative models (like those that might be learned by EM) in terms of the quality of the solution obtained. Intuitively, this should be especially true for temporal predictions, in which small errors in the model estimates may cause predictions to drift considerably for long sequences of observations.

With this view, the problem of learning a predictive representation becomes a traditional supervised learning problem, which can be solved in a straightforward way *if a mapping of histories H into a finite set of features Φ is given*. However, finding such a good encoding automatically may be very hard (this is akin to the feature construction problem). In the context of learning predictive models, an early attempt at this task is the work on utile distinction memory (McCallum, 2005), which learns an action-value function, based on a variable-length history representation. More recently, Wolfe and Barto (2006) used decision trees to learn similar predictions, but in a fully observable environment.

To make this problem computationally tractable, we restrict our attention to mappings in which $\Phi = \mathbb{R}$ (i.e., each history is mapped to a real number). Furthermore, we would like the value of this function to be updated incrementally as new action-observation pairs are received.

Definition 3.3.1. *A* **history probe** *is a function* $g : H \to \mathbb{R}$ *defined recursively as:*

$$\begin{aligned} g(ao) &= \theta_{ao} \\ g(hao) &= \varphi g(h) + \gamma\theta_{ao}, \end{aligned}$$

where $\theta_{ao}, \forall a \in \mathcal{A}, o \in \mathcal{O}$, γ *and* φ *are parameters.*

We will sometimes use the notation θ_i to mean $\theta_{ao_i} \forall a \in \mathcal{A}$ (i.e. when the weight of an observation does not depend on the action). In this case, we will refer exclusively to the weight of the observation, rather than the action-observation pair.

The idea is that some of the possible observations received from the environment are more interesting, or useful in predicting the tests of interest. These observations (henceforth referred to as *predictive features* of histories) would have a higher θ_i value. Summing up the weights of all the observations in the manner described above yields a representation of a history that measures in a way its predictive value. Thus, the history probe can be seen as constructing a special kind of *history representation* for any history.

This particular form of history representation is inspired by eligibility traces. Eligibility traces (Sutton and Barto, 1998) are a basic mechanism of reinforcement learning for temporal credit assignment. They represent a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. When an event (such as an error) occurs, only the eligible states or actions are assigned credit or blame for the error. Previous work (Loch and Singh (1998), Bellemare and Precup (2007)) observed that these functions produce good representations for partially observable systems, because they provide

implicitly a form of *memory* to remember past events. Note that in an MDP, g would be defined for states, and the usual eligibility traces can be represented by setting $\theta = 1$, φ to the product of the discount factor and the eligibility parameter (usually denoted λ) and $\gamma = 1$. However, for a general partially observable environment, we would like all these parameters to be learned from data.

The intuition behind using eligibility traces as history representations is that of discovering history features. If the observations of interest are specified (e.g., observations producing rewards), then the weights θ_i denote how "important" these observations are in determining the predictive value of the history. In the environment in Figure 3.3, we can see that the the orange wall is always preceded by a strip of green. If the set of tests of interest consists of tests in which the first observation is orange, (i.e., we are answering the question "Will I see the orange wall in the next step?"), then the green observation can be considered a predictive feature . The more recently this observation occurred, the more likely the orange wall will be seen in the next step. By using a history representation based on eligibility traces, we can easily assign temporal credit to the occurrence of this observation, and thus histories in which it has occurred in the last time step will have a higher value than those in which it occurred k steps in the past.

3.4 Conclusions

In this chapter, we have introduced a general way to specify the tests of interest. These are a subset of the possible futures, such as tests related to a specific goal, that the agent wants to predict. We have introduced prediction-based similarity relations

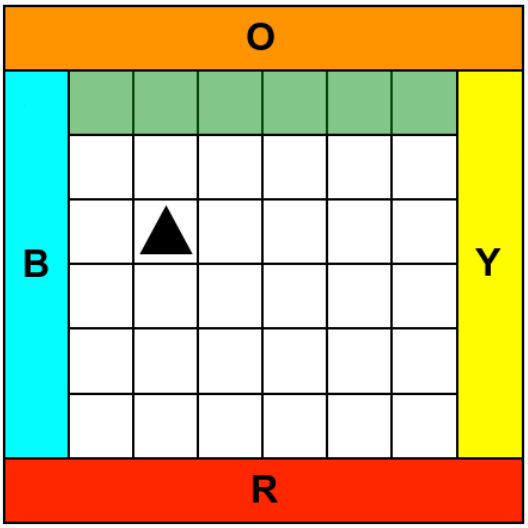

Figure 3.3: An example of history features. Since `green` always precedes `orange` in this domain, we can consider `green` to be the predictive feature in building the history representation

on histories and tests that allow us to collapse the system-dynamics matrix, and bounded the error in prediction caused by this operation.

Furthermore, because not all histories are relevant to predicting the tests of interest, we have introduced the notion of history features. These can be constructed from a history using a history probe, which is function that essentially computes the predictive value of a history (i.e., its usefulness for making predictions), given a set of predictive features. Using this history representation we can now make predictions conditional on "interesting" behaviour.

In the next two chapters, we create two different models based on this framework. The first assumes that the set of observations of interest is given, and learns the assignment of weights that minimizes the prediction error over all possible history-test pairs. The second model, presented in Chapter 5, extracts this set of observations from the agent's short term memory.

Chapter 4
Approximate Agent State Representations

We begin by introducing the first of two state representations based on the framework described in Chapter 3. In this chapter we present the Approximate Agent State Representation (AASR) framework, in which the agent is given the predictive features. This formulation requires the agent to have prior knowledge of both these features, as well as their weight in constructing the history representation. In Chapter 5 we generalize this approach using the idea of temporal coherence, and allow the agent to learn these parameters from its short-term experience. We provide an algorithm that learns this representation of a partially observable model from data, and illustrate it on a set of experiments. The ideas presented apply in non-Markovian environments, as well as systems with continuous states.

4.1 State Representation

Given a history probe g, we can create clusters of histories, where histories h_1 and h_2 are mapped together if their history representations are similar:

$$|g(h_1) - g(h_2)| \leq \epsilon_g,$$

where ϵ_g is a small real value. The history probe value of a cluster is:

$$g(H'_i) = \frac{1}{|H'_i|} \sum_{h \in H'_i} g(h).$$

The idea behind this clustering step is that if under a history probe g, two histories are assigned the same value, they must be very similar in terms of their predictive information as well. However, using solely the history representations (i.e., the history probe values), there is absolutely no guarantee that histories that end up in the same cluster actually have similar predictions. In Chapter 5 we improve on this method so that this clustering step takes into consideration the predictions, thus maintaining the error guarantee from Theorem 3.2.1.

The tuple $\langle T', H', f, g, p_f(T'|H') \rangle$ forms an approximate system dynamics matrix, which we will call the *approximate agent state representation* (AASR). This representation makes predictions only for the tests of interest given by f. The probed predictions $p_f(T'|H'_i)$ for a history cluster H'_i constitute the state of the model: they can be used both to make predictions about arbitrary tests of interest, as well as to maintain and update the state. Note that representing state in this way is reminiscent of PSRs, where state is represented by a set of predictions about core tests.

Making Arbitrary Predictions

Whenever a history h is observed, the prediction $p_f(t|h)$ for a test can be made as follows:

1. Compute the history representation $g(h)$, and uniquely determine the cluster to which it belongs:

$$i^* = argmax_i |g(H'_i) - g(h)|$$

2. Find the cluster of tests of interest T'_i closest to t.

3. $p_f(t|h) = p_f(T'_i|H'_{i^*})$

. Note that if t is not a test of interest, i.e., $f(t) = 0$ then $p(t|h) = 0$;

Maintaining State

This representation can be updated as the agent collects experience. As described in Algorithm 2, the initial estimates of $p(t|h)$ are computed by counting:

$$p(t|h) = \frac{\#\text{ of times } h \text{ as been followed by } t}{\#\text{ of times } h \text{ has been followed by } \sigma(t)}$$

Assuming that we have already mapped h to H'_{i^*} and t to t'_i respectively, then the updated prediction is

$$\hat{p}_f(t'_i|H'_{i^*}) = \frac{|H'_i|}{|H'_i|+1} \times \frac{(\#\text{ of times } h'_i \text{ has been followed by } t)+1}{(\#\text{ of times } h \text{ has been followed by } \sigma(t))+1}$$

Note that we must also update the denominator for all tests t'_j with the same that have the same skeleton, i.e., for which $\sigma(t'_i) = \sigma(t'_j)$. Finally, if this history has not been encountered before, we must update the history probe value of the cluster:

$$\hat{g}(H'_{i^*}) = \frac{|H'_i| \times g(H'_{i^*}) + g(h)}{|H'_i|+1}$$

4.2 Learning Algorithm

Algorithm 2 presents our approach for learning the AASR model from data, assuming that the f and g probes, along with the thresholds ϵ_f, ϵ_g are given. First, we cluster the set of all tests; then, we cluster the histories.

Let θ_g be the weight of the history feature (i.e. the observation that has the highest weight in the history representation), and θ_{non} be the weight of all other observations. For simplicity, we're only assuming there is one history feature, and that all other observations weight the same; of course, a vector of weights can be just as easily used. The history clustering depends on the parameters θ_g and θ_{non} of the history probe. In order to choose the optimal parameters, we fix θ_{non} to a small value, and perform line search over a large range of values for θ_g to find the value that minimizes both the prediction error, as well as the number of history clusters.

At every step in the line search, we can use a subset of the data set $D' \subset D$ to create a temporary AASR, M'. To evaluate its prediction error, we perform cross validation by training a second AASR, M, with the same set of parameters, using the entire data set D. The prediction error for M' is the total average difference (for all tests and histories in D), between the probed predictions of M and M'.

The ϵ_f parameter can be chosen to ensure a good tradeoff between the size of the representation and the prediction accuracy. The granularity of the history clusters is controlled using ϵ_g; a larger ϵ_g will allow histories that do not have the same probed predictions to be collapsed in the same cluster, which will generally lead to a higher prediction error.

4.3 Experimental Results

The goal of these experiments is to learn how to model the environment and predict what will happen in the future, for a set of tests of interest. We consider the "goodnes" of a model to be the total average prediction error, defined as follows:

let $p_f^M(t|h)$ be the prediction given by a model M, and $\hat{p_f}(t|h)$ the true prediction (calculated from the actual Markov Decision Process representing the system), then the total average prediction error is:

$$error = \sum_{\forall h,t \in X} |p_f^M(t|h) - \hat{p_f}(t|h)|,$$

where X is a testing set consisting of history-test pairs.

We compare our approach with the results of models learnt using Expectation Maximization. We do not compare our results directly with PSRs, as they are trying to learn an exact model of the environment, i.e., predict all futures; we would thus expect that PSRs will need much more data to become stable, and will have very high errors for the amounts of data that we use.

4.3.1 Tunnel World

The first environment we will consider is a small probabilistic domain, shown in Figure 4.1. The agent transitions from state i to state $i-1$ with probability p, and to state $i+1$ with probability $1-p$. In our experiments, $p = 0.7$. There are two deterministic observations: dark (D) and light (L).

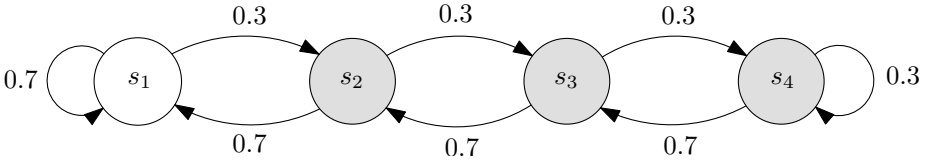

Figure 4.1: Tunnel World

Task

The tests of interest are given by a binary test probe: for any test t whose observation sequence contains light, $f(\omega(t)) = 1$; otherwise, $f(\omega(t)) = 0$. The predictive history feature is defined in the same way because once the agent has reached the end of the tunnel, it is very likely to stay there. This history representation only takes into consideration the observation sequence starting at the last time light was seen, because seeing light once is sufficient to determine exactly the agent's position in the tunnel.

Experiment Details

The data consists of action-observation trajectories of length 16, starting in the rightmost state (s_4 in the figure). From these, we extract the initial $p(t|h)$ predictions through counting, while requiring the histories in the table to be no longer than 10 time steps, and the tests no longer than 6. All results are averaged over 10 independent runs. We use $\gamma = 0.8$, $\phi = 0.2$. Two tests are considered equivalent if their predictions are within $\epsilon_f = 0.04$ away from each other. Similarly, $\epsilon_g = 0.001$. The parameter for the light observation, θ_L, is chosen by a line search, as described in Algorithm 2.

Results

In Figure 4.2 we present the total average prediction error, i.e., the difference, over all history and test pairs, between the true predictions and the predictions given by the learned model. We first observe that all models built with EM converge to a local

optima, and that this optima depends on the initial distributions of the parameters. For example, if we use the true number of states in the MDP (i.e., four), but initialize the transition and observation distributions randomly, the resulting model has a very high variance and prediction error. However, initializing the same model to distributions that are very close to the correct ones, these results improve greatly. We can also see that just overestimating the number of hidden states (to the number of states discovered by the AASR model) does not perform better. Conversely, the AASR model finds a good solution very quickly, and continues to improve as more data is seen, as the probabilities in the approximate system-dynamics matrix become more accurate.

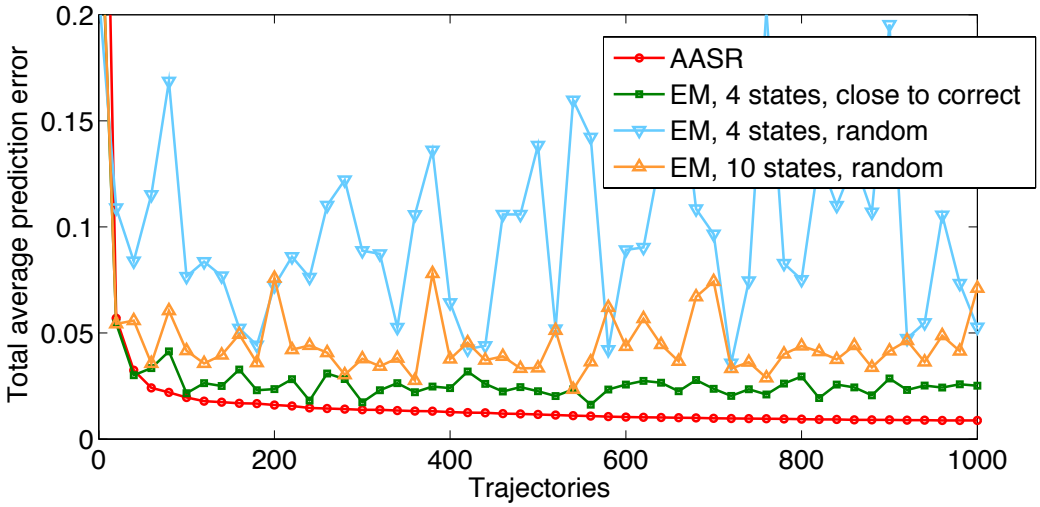

Figure 4.2: Tunnel World - Graph comparing the average prediction error of the AASR model vs. that of 3 different models learnt with Expectation Maximization

We can also look at the greatest prediction error over all history-test pairs, illustrated in Figure 4.3. Overall, the AASR model has the highest maximum error, but this is to be expected - because the history representations do not take predictions

into considerations, there exist some cases in which two histories are incorrectly aggregated together. However, this occurs rarely, as the total average prediction error is still very small. We can also see that this error drastically decreases as more data is seen and the predictions in the system-dynamics matrix become closer to the true values.

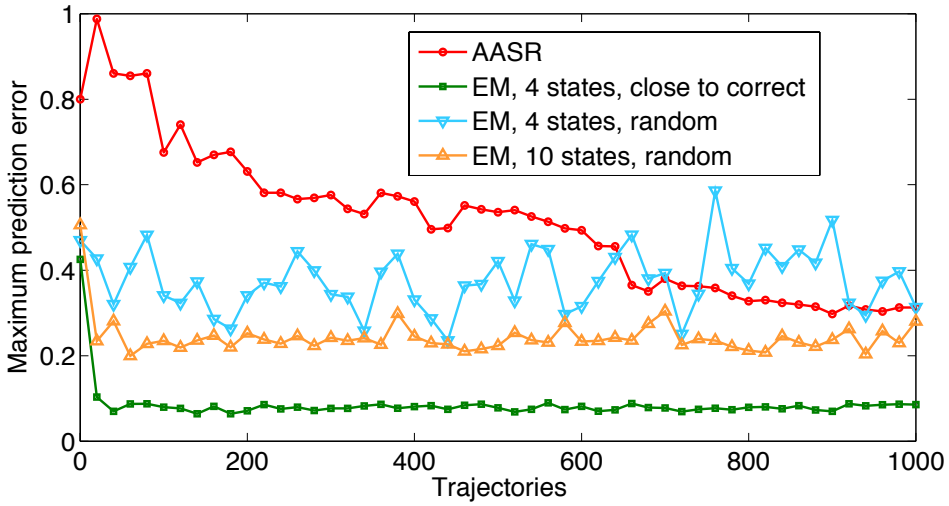

Figure 4.3: Tunnel World - Graph comparing the maximum prediction error of the AASR model vs. that of 3 different models learnt with Expectation Maximization

In Figure 4.4 we can see the number of states (i.e. histories that are not equivalent) that the AASR discovers does not increase as more data is seen, which is a very positive result. From the previous graphs we know that its accuracy increases, thus proving that the original error was caused by the inaccurate $p(t|h)$ estimations, and not from incorrect clustering. The states of the final AASR model correspond to histories which determine the position in the world; for example the histories aD

and aDaDaL indicate whether the agent needs to take at least 2 steps, and respectively one step in order to see Light, and thus each of these histories are identified as different clusters by the algorithm. The other clusters have a similar meaning.

We can also see that the θ_L parameter, the weight of the Light observation also converges within a small number of trajectories, confirming that the model converges quickly to a parameter configuration, and then continues to improve its predictions.

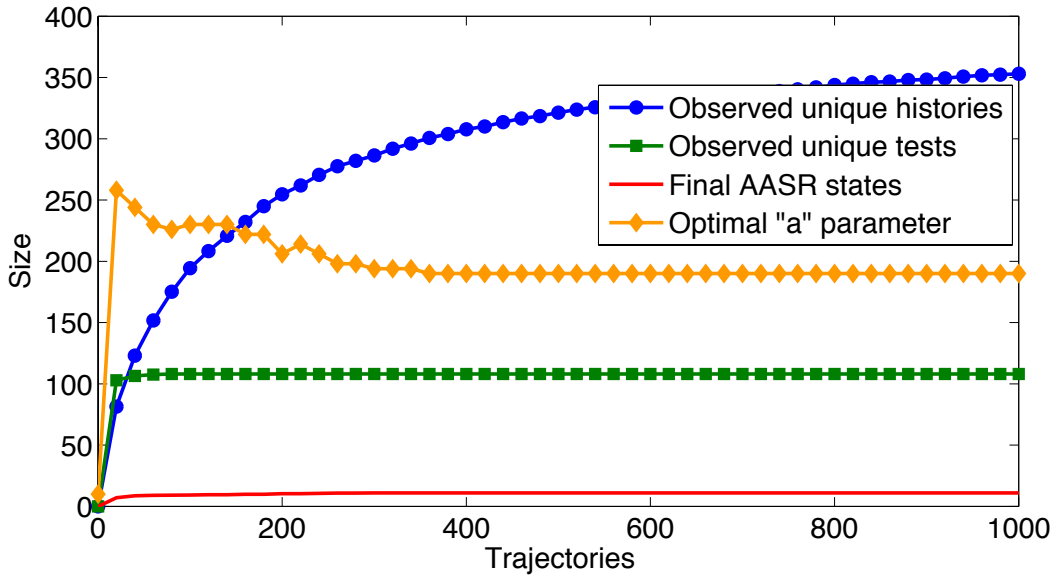

Figure 4.4: Tunnel World - Number of AASR States

Finally, we analyse the effect of ϵ_g on the final AASR model in Figure 4.5. As expected, the number of states in the final representation decreases as ϵ_g increases. This is because for very large values of ϵ_g, all histories are considered essentially equal, and are collapsed in the same cluster. Similarly, the error in prediction increases with an increase in ϵ_g, as histories are aggregated incorrectly. As the number of

clusters goes down, the error decreases again, but not as a result of the quality of the representation. When histories are clustered together, the probed prediction vector is averaged between the members of the cluster. Thus, when there is only one cluster, its prediction vector is an average of all prediction vectors; given that many predictions are 0, this average prediction is essentially noise, which can end up being very close to the average of all correct predictions. However, we can see that the smallest error is for the smallest ϵ_g, in which case only histories that are very close are collapsed, thus minimizing the number of incorrect cluster assignments.

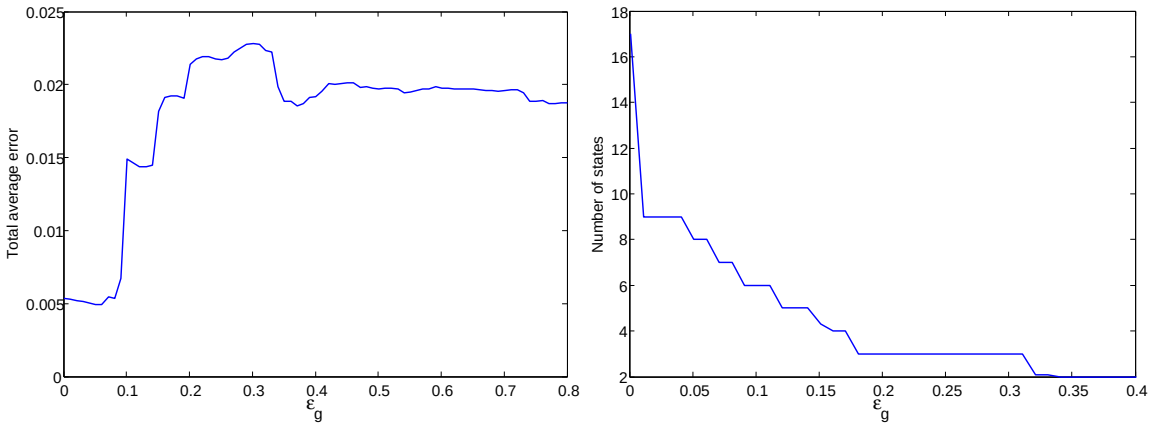

Figure 4.5: Tunnel World: Effect of ϵ_g on the final representation

4.3.2 Non-Markovian Tunnel World

The approximate agent state representation works well in cases where the data does not come from a Markov system as well. We have modified the transitions in the tunnel world to depend on the "direction" of moving. Now, the agent transitions with probability p in the direction of movement, and $1-p$ in the opposite. Initially, the agent is going left, and thus the transitions are the same as in Figure 4.1.

However, when the agent reaches s_0 and loops, the direction changes, as if the agent "bounced" off the wall. Since the agent starts moving right, the probabilities on the transitions in Figure 4.1 are reversed. Similarly, the direction changes from right to left after bouncing in s_3. The observations as well as the start state are the same as before. This domain is non-Markovian because state alone cannot determine the next state; this transition changes depending on the direction of movement.

In Figure 4.6 we present the total average error over all pairs of histories and tests, averaged over 10 trials. Again, we compare the error obtained from our algorithm with that obtained by three models created using Expectation Maximization (EM): one that fits the data to 8 states (i.e. the size of the corresponding Markovian Hidden Markov Model model), and two that fit the data to 12 states (bigger than the size of the learnt AASR). Again, the error from the EM model is significantly higher, and with greater variance.

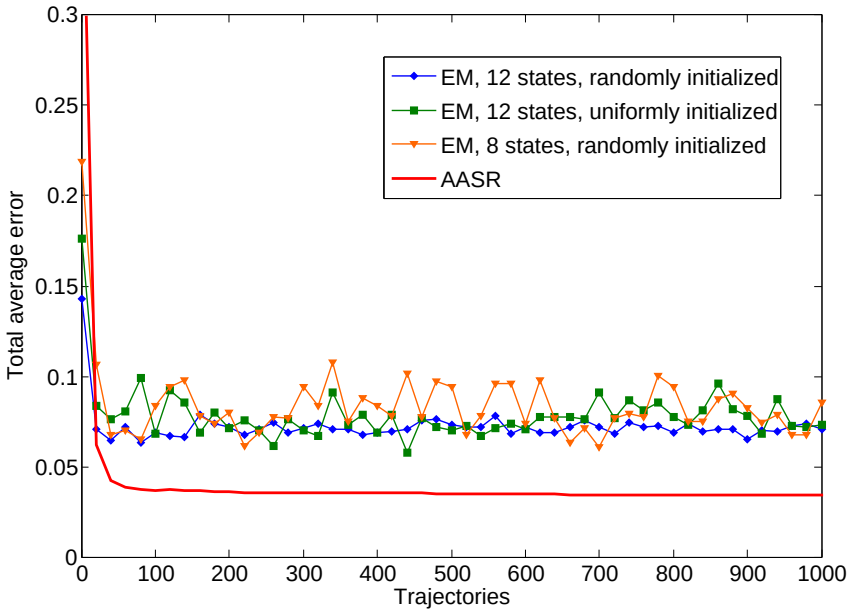

Figure 4.6: Non-Markovian Tunnel World - Total Average Error

4.3.3 Continuous Tunnel World

We can also apply the AASR algorithm to learn the dynamics of a continuous model. In this case, the agent starts randomly in the interval $[0, 1]$. With probability $p = 0.7$, it transitions from state x to state $x' = x - \delta + \epsilon_{noise}$, where $\delta = 0.2$, and ϵ_{noise} is drawn from a Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 0.5$. With probability $1 - p$ it transitions to $x' = x - \delta + \epsilon_{noise}$. States outside of $[0, 1]$ are truncated to 0 and 1 respectively. States that are ≤ 0.2 see light, and all others see dark. The total average error, averaged over 10 trials, is presented in Figure 4.7.

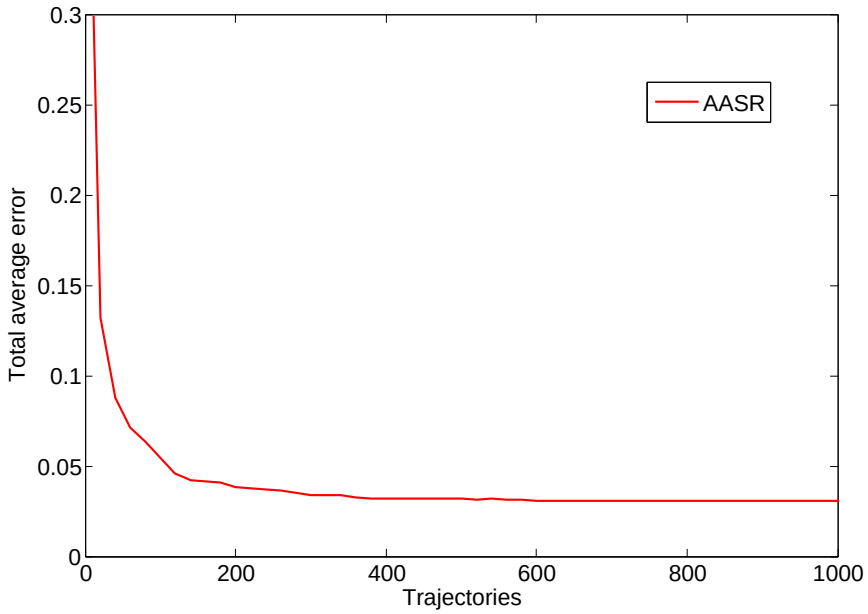

Figure 4.7: Continuous Tunnel World - Total average error

4.3.4 Gridworld

We now consider a larger domain, similar to the one used in Rafols et al. (2005), pictured in the left panel of Figure 4.9. Each grid cell has 4 different orientations,

and thus there are 6x6x4 states. There are 2 actions, forward (F) and turn left (L), which changes the orientation, but keeps the agent in the same grid cell. The system is stochastic, and each action has a 5% probability of failing (in which case the system remains in the same state). The system is also partially observable: if the agent is next to a wall and facing it, it will observe the wall's colour, otherwise it will observe white.

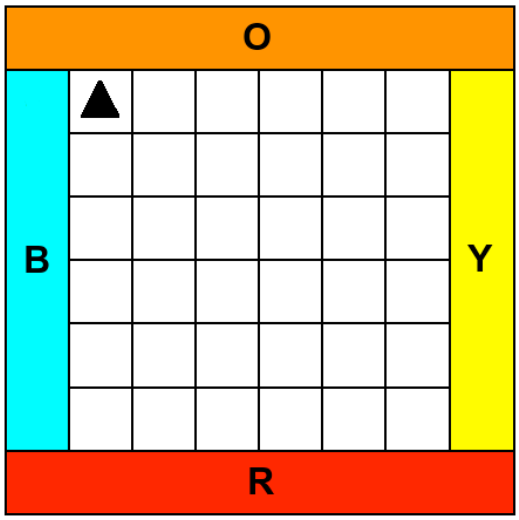

Figure 4.8: Grid World

Task

The goal of the task is to make predictions about whether the agent will see the orange wall again, given its history seen so far, and given a sequence of actions to be taken. The f probe is 1 for all tests that contain the observation orange, and 0 otherwise. The predictive history feature is also orange.

Experimental details

We repeatedly sample new trajectories consisting of 40 transitions (histories of length 30, tests of length 10), and then use them to learn the model. The start state of each trajectory is the top left corner, oriented towards the orange wall. As before, $\gamma = 0.8$, $\varphi = 0.2$, $\epsilon_f = 0.04$, and $\epsilon_g = 0.001$, and all results are averaged over 10 runs.

Results

The final model has no more than 50 states, regardless of the number of observed trajectories, as can be seen in Figure 4.9. As before, the algorithm settles on this model very quickly, and its complexity does not increase as more data is seen. This representation is several orders of magnitude smaller than the number of observed unique histories, illustrating the power of history representations.

Rather than plotting the total average predicting error as before, we instead plot the average prediction error for selected history-test pairs in Figure 4.10. These pairs vary in both the length of the predicted tests, as well as the observed history. As expected, shorter predictions such as $p(fO|[])$ have a smaller prediction error, which converges very quickly. This is because the actual estimate in the approximate system-dynamics matrix is very close to the real value, as the history-test pair occurs very often. This implies that a model predicting the immediate futures can be learned from a very small number of samples. However, longer tests are seen with much lower frequencies, and as a result, their initial $p(t|h)$ predictions are very inaccurate, and decrease slower.

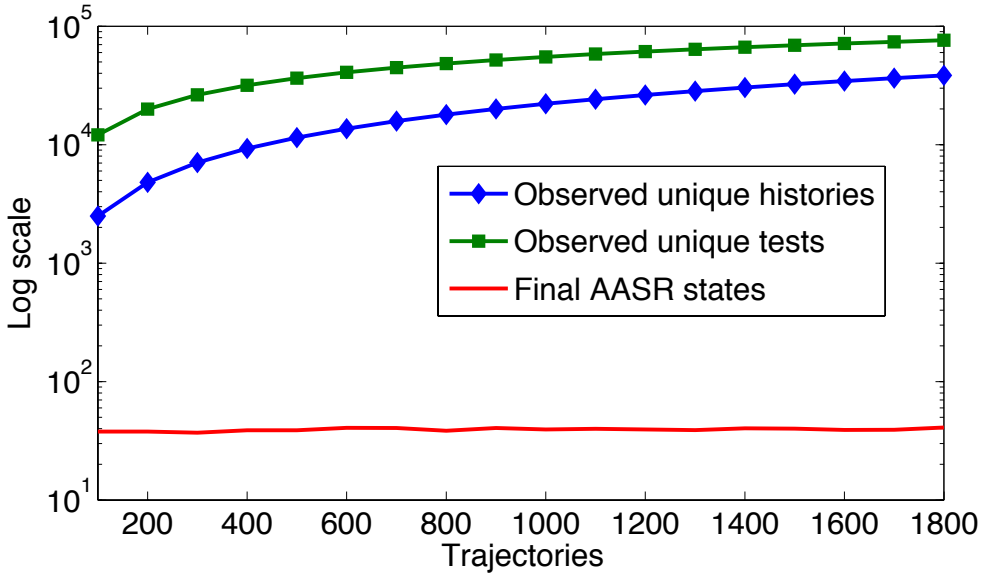

Figure 4.9: Grid World - Number of states in the AASR model

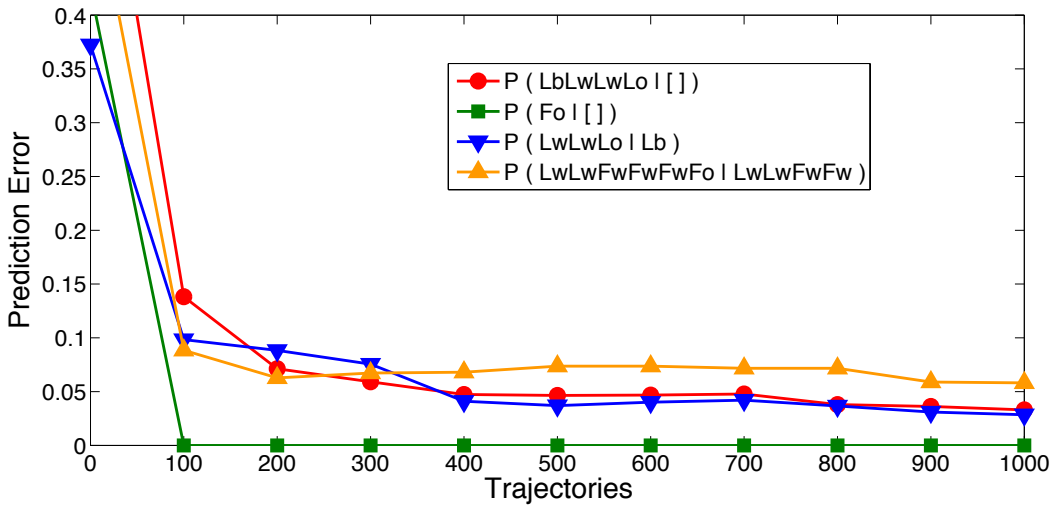

Figure 4.10: Grid World - Prediction error for selected tests

4.4 Conclusions

While the AASR algorithm works well in practice for small examples, it does not scale up. The main flaw is that in order to build an approximate representation,

it must first build an estimate of the system-dynamics matrix. Although this is finite, it is still exponential in the length of histories and tests, and thus becomes intractable very quickly. Secondly, the history representation used by the algorithm does not take predictions of tests in consideration. It is easily conceivable that if the wrong set of observations of interest is given (for example, observations that are very far away from those used by the f probe), then the model will make incorrect predictions, that are conditioned on the wrong kind of behaviour (i.e., meaningless history clusters).

In the next chapter we propose an algorithm that automatically learns these history representations from data at the same time it builds the approximate system-dynamics matrix.

Chapter 5
Local agent state representations

Agents often have very limited resources, in terms of both the space allotted to store data as well as the time required to process it. As a result, the AASR approach outlined in Chapter 4 is not advantageous. Although specifying a set of tests of interest does simplify the problem, we are still required to first compute a reasonably accurate subset of the system-dynamics matrix, and only then can we summarize it further. Since the system-dynamics matrix is exponential in the length of tests and histories, increasing the length of the experience quickly makes this algorithm intractable.

A more desirable approach would be to create an approximate model directly, by summarizing the agent's experience as it occurs, while at the same time constructing a history probe automatically, such that history representations are close for histories with similar predictions, and very different for histories with different predictions. Many environments, such as the Half Moon domain presented in Figure 5.1 are very localized, and summarizing the entire long term experience is not necessarily useful for predictions. We will exploit the temporal characteristics of such environments by creating a model that is based on the agent's short term experience. As the agent

continues to learn, the environment may change, and so will the local model of the world. By considering local models based on short term experience, the agent can avoid certain biases that have happened in the past, and make predictions based on the more relevant present experience.

5.1 Temporal Coherence

Temporal coherence (Koop, 2007) is defined as the tendency of domains, and thus observations, to be consistent over a short term period of time. Imagine a robot that is going down a hallway with no obstacles. If the hallway is partially observable, the agent may receive the same observation for many successive time steps. Storing the entire experience in this case is wasteful, and has no prediction benefits. It is only when the agent observes something of importance, such as a door, or a corner, that its experience can actually be used to determine its position in the world, and thus make predictions. The Half Moon example illustrates this idea. Each of the observations, black and white, are temporally coherent - if black has been observed recently, the agent is very likely to be in the black area and thus observe it again.

The g probe defined in Chapter 4 required the agent to know which history features were important before hand. This is not always a feasible approach, as for complex environments it might not be immediately clear which observations, or combinations of observations are important for predictions. Also, if histories are collapsed based solely on the probe values, there is absolutely no guarantee that histories that end up in the same cluster actually have similar predictions. We would

like to *construct a history probe* that ensures similar history representations make similar predictions.

Thus, we want to learn a parameter configuration $\theta = \{\varphi, \gamma, \theta_{o_i}, \ldots, \theta_{o_n}\}$ that minimizes the distance (in terms of the value of the history probe) between histories in the same cluster, while maximizing the distance among all clusters. Formally, we pick θ with the largest distance between the history representations of clusters:

$$\max_{\theta} \sum_{i,j} [g(H'_i) - g(H'_j)]^2, \forall H'_i, H'_j \text{ clusters} \tag{5.1}$$

such that the distance between the history representations within clusters is small:

$$\forall h_i, h_j \in H'_i, |g(h_i) - g(h_j)| \leq \max_{T_i \in T'} |p_f(T_i|h_i) - p_f(T_i|h_j)| \tag{5.2}$$

Because the distance between histories in the same cluster is bound by a prediction error, which is a probability value, we must normalize the values of the history probes by dividing each $g(h)$ by the theoretical maximal value, computed as follows:

$$\begin{aligned} g(a_0 o_0 \ldots a_n o_n) &= \varphi g(a_0 o_0 \ldots a_{n-1} o_{n-1}) + \gamma \theta_n \\ &= \varphi[\varphi g(a_0 o_0 \ldots a_{n-2} o_{n-2}) + \gamma \theta_{n-1}] + \gamma \theta_n \\ &\ldots \\ &= \varphi^n \theta_0 + \gamma \sum_{i=0}^{n-1} \varphi^i \theta_{n-i} \end{aligned}$$

where $\theta_i = \theta_{a_i o_i}$. In the limit as the length of the history goes to infinity, the first term goes to zero, and using the geometric series, the second term converges to $\gamma \frac{R}{1-\varphi}$, where $R = \max_i \theta_i$ is the highest weight over all observations.

5.2 Learning algorithm

To solve this optimization problem, we begin by constructing a small system-dynamics matrix from the short term experience (i.e. a couple of steps in the past). The size of this matrix can be chosen by the experimenter as an imposed memory limitation on the agent, or experimentally by trying increasingly longer trajectories. Because it only uses short history test pairs, these occur frequently, so the matrix can be easily and accurately computed. Then, we determine the exact clusters of histories and tests of interests. This is similar to the clustering procedure described before, with the exception that histories are now collapsed according to their predictions, and not the values of the history probe. First, the set T' of tests of interest are formed, such that:

$$\forall h |p_f(t_1|h) - p_f(t_2|h)| \leq \epsilon_f.$$

Then, clusters of histories, H', are formed, such that:

$$\forall T_i, |p_f(T_i|h_1) - p_f(T_i|h_2)| \leq \epsilon_g$$

where ϵ_g , ϵ_f are small real values. Then, we perform a search in parameter space to solve the optimization problem above. Note that more efficient solution methods can be used here; we use line search just for simplicity.

For every parameter configuration, we first check whether or not the condition in Equation 5.2 is satisfied. If it is, we calculate the distance between clusters according to Equation 5.1, and if this yields a larger distance, then we update our optimal parameter configuration to the current one.

This approach is outlined in Algorithm 3.

5.3 State Representation

The short-term system dynamics matrix, along with the parametrized g function learned from it are the new predictive model, which we call the *local agent state representation* (LASR). Predictions of tests already in the system-dynamics matrix are made as explained above, i.e. the history h is mapped to the cluster closest to $g(h)$, and read from the cluster's probed prediction vector. To make long term predictions, we assume that predictions are *compositional*, and that

$$p_f(t_1 t_2|h) = p_f(t_2|ht_1)$$

The agent is now guaranteed to have errors in prediction - for example, whether or not ht_1 can occur, it will be mapped to an actual cluster, and a valid, possibly non-zero prediction will be returned. Note that compositionality can be checked from data, and a new representation could be learned if compositionality is violated; however, this goes beyond the scope of this thesis and we leave it for future work.

5.4 Experimental Results

Half Moon World

We illustrate the benefit of using local models on the Half Moon world (Koop, 2007), depicted in Figure 5.1. In this domain, each of the observations, black or white, are temporally coherent - if black has been observed recently, the agent is very likely to observe it again. A model that takes advantage of this should perform better. There

are twenty states, half of which observe black, and the rest white. The agent floats with equal probability between any two adjacent states.

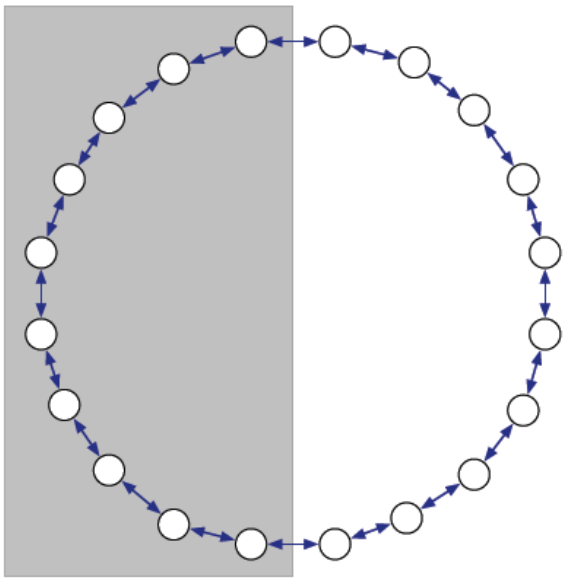

Figure 5.1: Half Moon world

Task

The agent has a very short term memory available: trajectories of length 5, which are split in histories up to length 3 and tests up to length 2. There are several things we are looking for in these experiments. First, we would like to build a model that has a small overall error, meaning that it can predict well despite the approximation. Secondly, we are interested in seeing whether or not the localization assumption is correct, i.e., whether the model does perform better in parts of the system that are temporally coherent.

Experimental Details

The start state is the first white state at the top and the f probe assigns a value of 1 to any test containing Dark, and 0 otherwise, thus answering the question "Will I see Dark in the near future". All results are averaged over 10 runs, with $\gamma = 0.8$, $\epsilon_f = 0.1$ and $\epsilon_g = 0.01$;

Results

We begin by comparing the error in prediction of the learnt LASR with a 20 state model learnt with Expectation Maximization (EM), where the initial parameters are initialized very close to the correct ones. We use two different scenarios, one in which the testing data is composed of histories of length 10 and tests of length 1 (thus effectively answering the question "Will I see Dark" in the next step), and a second in which the testing data has tests of length 4. Because the model was trained on trajectories of length 5, this means that the predictions it is asked to made will be about unexplored parts of the domain.

In Figure 5.2 we can see that in both scenarios the LASR model has a smaller total average prediction error (over all history-test pairs in the training set) when compared to EM. This difference is very obvious inthe harder prediction scenario (with tests of length 4), in which the LASR performs almost as well as the EM model does in the simple prediction case (with tests of length 1).

In Figure 5.3 we present the maximum prediction error of the models.The fact that the error is higher than the theoretical bound of $2(\epsilon_g + \epsilon_f)$ comes from the compositionality assumption. Assuming that there has been enough data for the

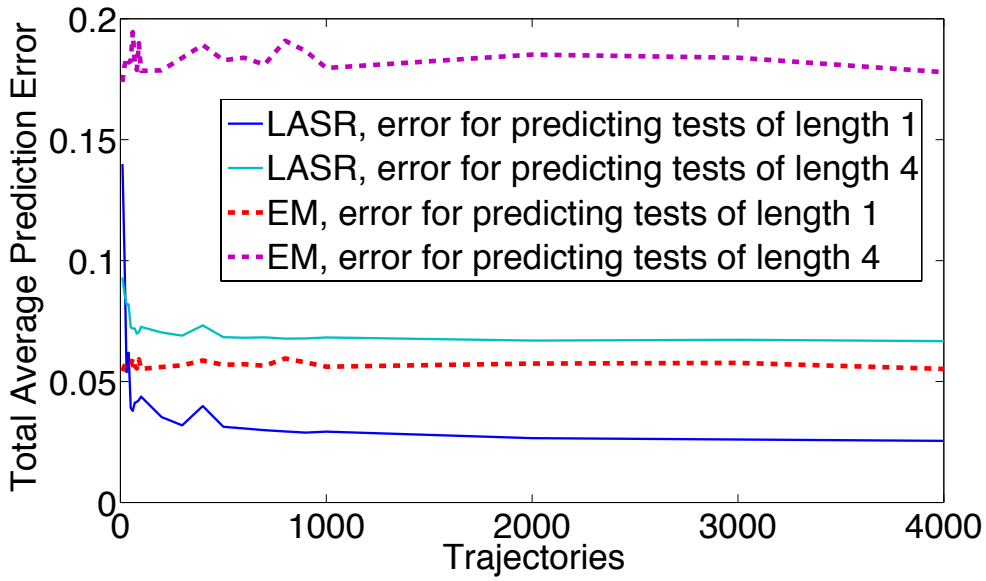

Figure 5.2: Half Moon world - Average Prediction Error

estimates in system-dynamics matrix to be accurate, the bound on prediction error holds for tests that the algorithm has seen, i.e., up to 2 steps in the future. We can see that the maximum prediction error of the LASR model converges to the correct bound in the case where it is asked to predict 1 step in the future. However, when asked to predict 4 steps into the future, the model must compose predictions together. This means that in some cases it will assign non-zero probabilities to trajectories that cannot occur. This is also why the maximum prediction of the LASR model is higher than that of EM.

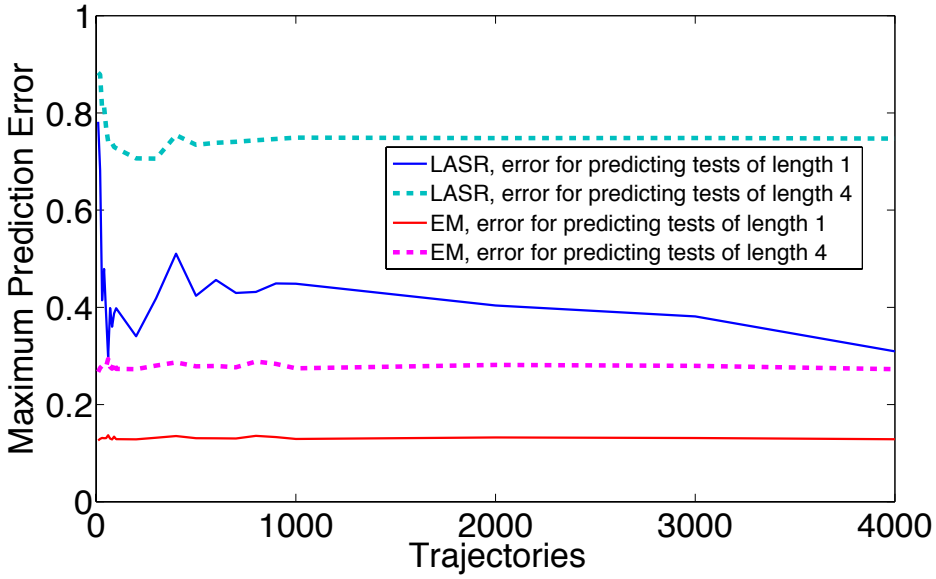

Figure 5.3: Half Moon world - Maximum Prediction Error

In Figure 5.4 we present the change in the two θ parameters over time. It is interesting to note that the LASR agent prefers assigning a higher weight to the White observation, even though it is not the observation of interest in the tests.

Task

We now consider a different task, in which we compare the effect of the temporal coherence of the model on the prediction error. We consider two different trajectories, starting from two different states: the same start as before, as well as a state to its left, in the black zone. We expect that in the second case, black will be predicted accurately, but white will not, as the agent has not experienced the observation in its recent experience.

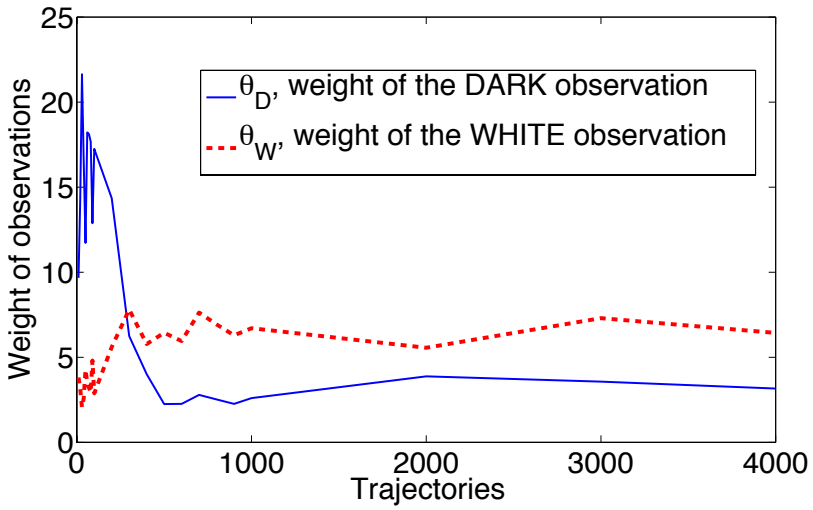

Figure 5.4: Half Moon world - LASR Parameters

Experimental Details

As before, the agent has a very short term memory available: trajectories of length 5, which are split in histories up to length 3 and tests up to length 2. The starting states for the two experiments are given in Figure 5.5. The f probe assigns a value of 1 to any test containing Dark, and 0 otherwise, thus answering the question "Will I see Dark in the near future". All results are averaged over 10 runs, with $\gamma = 0.8$, $\epsilon_f = 0.1$ and $\epsilon_g = 0.01$;

In Figure 5.6 we present a sample trajectory that starts in the state marked s_1 in Figure 5.5. The horizontal axis shows the observation at each time step, and the vertical axis is the prediction of seeing dark in the next time step. Because the model is not very temporally coherent (seeing white does not necessarily guarantee you will see white in the next step), it does not perform well when evaluating a long sequence of the same observation. However, because the agent's recent experience

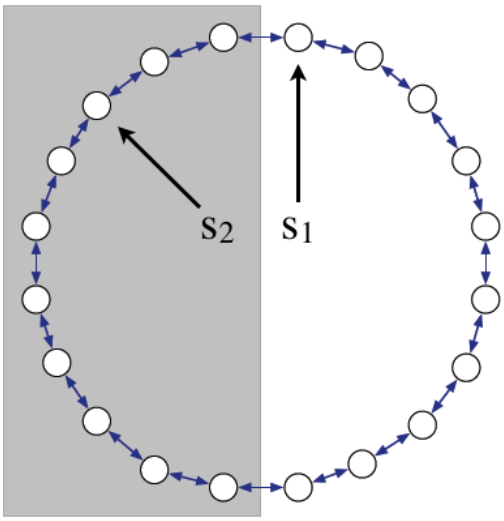

Figure 5.5: Half Moon world

has alternating observations, the model performs much better when evaluating alternating observations. This implies that agent predicts consistently to what it has seen recently. The idea is that as this recent data changes (for example, the agent moving on to a different part of the world), the type of predictions it will make will change as well.

This idea is illustrated in Figure 5.7. We can now assume that the agent has moved to a different part of the world, as trajectories are now starting in the state marked s_2 of Figure 5.5. In this case, black is very temporally coherent, as the agent observes black for at least 3 steps in a row. We can see that the model predicts black much better when in the black part of the trajectory than everywhere else. However, when as soon as it starts exploring the white area, its predictions start incurring errors, as it has not explored this part of the environment before. In the future, the model should have a way to update itself based on this fact.

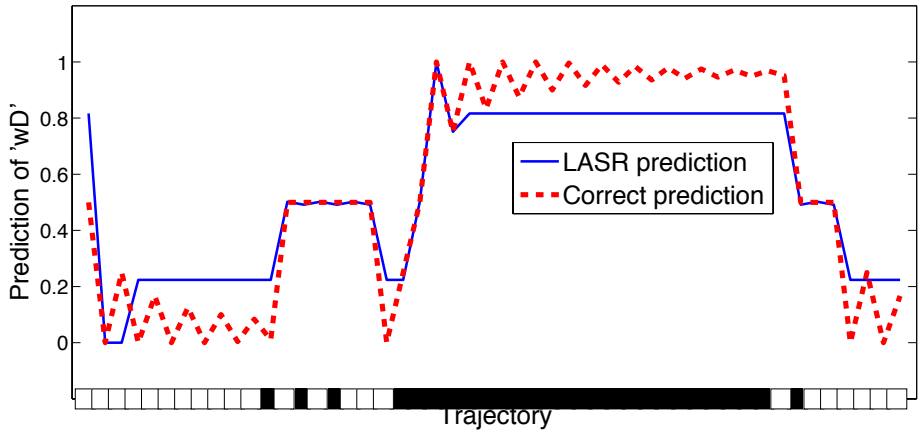

Figure 5.6: Half Moon world - Trajectory starting in a less temporally coherent part of the world

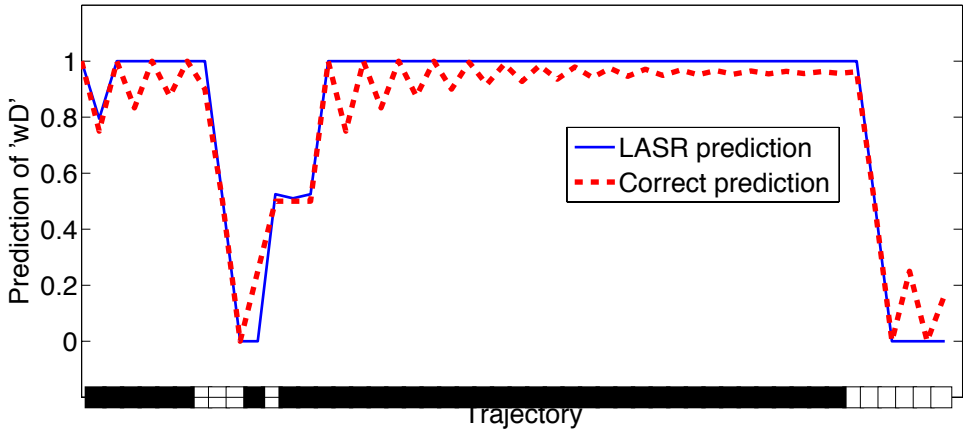

Figure 5.7: Half Moon world - Trajectory starting in a temporally coherent part of the world

5.5 Conclusions

The temporal coherence of the agent's experience tends to affect the short term predictions it needs to make. In this chapter we illustrated this concept by showing that

a small model, built from an agent's short term experience can be used make accurate predictions. This model has the advantage that it automatically discovers the observations of interest that make a history useful for predicting the set of interest, by extracting at the "interesting" behaviour from its recent experience. Because this algorithm does not try to construct a large system dynamics matrix, it can easily scale up to large examples, with more observations and actions than the previous AASR approach could.

The LASR model still has all the benefits of models with predictive state. Furthermore, it can be easily used for planning in control tasks. If the test probe f is chosen to either select tests containing the goal observation, or as a sum of future discounted rewards, then the predictions for each of the test of interests can be used as feature representation. These features can be used as input to standard Reinforcement Learning algorithms (Kaelbling et al., 1996) to learn a control policy.

Chapter 6
Conclusions and Future Work

In this thesis we are primarily concerned with ways of representing the agent's state that allows it to predict the conditional probability of future events, given sequences of the agent's past experience. The agent's experience comes from interacting with a complex, partially observable environment. Due to computational limitations an agent will rarely be able to store its entire experience, and thus will not be able to build enough of the system-dynamics matrix to construct a correct and accurate linear PSR. We investigate methods of reducing the size of the system-dynamics matrix by specifying the agent's interest in future events. Building a model that makes predictions about a restricted set of tests is simpler, as it requires less data. These models, accurate with respect to a set of tests of interest, can be combined to answer a more comprehensive set of questions about the world.

We formally specify the tests of interest to the agent through the notion of a test probe. Using this probe, we showed how one can collapse the system-dynamics matrix to yield a smaller matrix, that makes accurate predictions with respect to the set of tests of interest. In a similar manner, we defined the idea of history probes, which allow us to represent the predictive information of a history according to a

set of features of interest. This function is very similar to eligibility traces, as it implicitly provides a form of memory to remember past events. These events, or observations, represent the interesting behaviour in the agent's experience, that can be used to make the predictions of interest.

Assuming that this set of history features were given, we can construct the Approximate Agent State Representation model. This model has predictive state, and can directly make arbitrary predictions. However, since the the system-dynamics matrix is exponential in the length of tests and histories, increasing the length of the experience quickly makes this algorithm intractable. In addition, specifying which observations in the agent's experience can be considered useful for predictions is not always obvious. Instead, we provided an algorithm that automatically constructs a history probe, such that histories with similar representations make similar predictions, and different history representations make very different predictions. This history probe is learned from the agent's short term experience. The resulting model, the Local Agent State Representation has the advantage that it automatically discovers the observations of interest that make a history useful for predicting the set of interest, by extracting at the "interesting" behaviour from its recent experience.

6.1 Future Work

We presented a blueprint for creating approximate representations of partial environments from data, and a simple algorithm which makes these ideas concrete. The

results are very promising, but more experimentation, in larger domains, is necessary. For example, it needs to be further examined how the LASR model will scale up to domains which are not temporally coherent.

The main area of future work is that of history representations. In more complex tasks, a mapping of histories into the real numbers may not be sufficient, and instead one may need to use vectors of observations, or action-observation pairs.

The models that we learn do not have a set of underlying states, and thus do not try to model the dynamics of the system directly. Because they only maintain predictions about future observable events, these models can be used better in generalized tasks. One interesting example would be to see whether models can be easily transferred across similar domains. For example, one could learn a model (in which the tests of interest are those related to a goal), and use it to make predictions in domains with slightly different dynamics. These models can also be used for control, and they don't depend on the reward signal directly, so it would be interesting to see their performance in learning different policies. Finally, while our approach allows for the test probe to take the form of the discounted sum of rewards, experiments are needed to demonstrate its effectiveness in control tasks.

In terms of solving the the optimization problem that we defined (i.e., that of finding the weights of each observation in calculating the value of a history probe), much better methods than line search can be used. For example, the problem could be formulated using $L1$ regularization, or as a linear program.

Chapter 7
Appendix A: Algorithms

Algorithm 1 Estimating an approximate system-dynamics matrix

Input: A set of data $D = (h, t), \forall h \in H, \forall t \in T$, probes: f, ϵ_f, ϵ_g
Output: history clusters H', tests of interest T', probed predictions $p_f(T'|H')$

1. Use the data to estimate $p(t|h)$, e.g. by counting:

$$p(t|h) = \frac{\#\text{ of times } h \text{ as been followed by } t}{\#\text{ of times } h \text{ has been followed by the action sequence of } t}$$

2. Compute the probed predictions:

$$p_f(t|h) = p(t|h)f(\omega(t)), \forall h \in H, t \in T$$

3. Cluster tests into the set of tests of interest

$$T' = \{T'_i | t_1, t_2 \in T'_i \Rightarrow \forall h \in H, |p_f(t_1|h) - p_f(t_2|h)| \leq \epsilon_f\}$$

4. Compute the probed prediction vectors for all histories $h \in H$

$$p_f(T'|h) = [p_f(t'_1|h), p_f(t_2|h), \ldots, p_f(t'_k|h)],$$

5. Cluster histories

$$H' = \{H'_i | h_1, h_2 \in H'_i \Rightarrow \forall t \in T', |p_f(t|h_1) - p_f(t|h_2)| \leq \epsilon_g\}$$

6. Compute the probed prediction vectors for the history clusters

$$p_f(T'|H'_i) = \frac{1}{|H'_i|} \sum_{h_i \in H'_i} p_f(T'|h_i).$$

Algorithm 2 Learning the Approximate Agent State Representation

Input: A set of data $D = (h, t), \forall h \in H, \forall t \in T$, probes: f, g, ϵ_f, ϵ_g
Output: history clusters H', tests of interest T', probed predictions $p_f(T'|H')$

1. Compute the approximate system-dynamics matrix using Algorithm 1
2. Initialize the history probe parameters: $minError = \infty, \theta_g = 1, \theta_{non} = 1, H' = H$
3. Repeat until $minError \leq \epsilon_f$

   (a) Cluster histories using the g probe given by the current parameters:

   $$H^{current} = \{H_i^{current} | h_1, h_2 \in H_i^{current} \Rightarrow |g(h_1) - g(h_2)| \leq \epsilon_g\}$$

   (b) Compute the prediction error $currentError$ of this representation using cross validation

   (c) Pick the representation with the smallest prediction error and smallest number of states:
   If ($currentError \leq minError$) and ($|H^{current}| \leq |H'|$), then:

   $$minError = currentError, H' = H^{current}$$

   (d) Advance to the next parameter configuration

Algorithm 3 Learning the Local Agent State Representation

Input: A set of data $D = (h, t), \forall h \in H, \forall t \in T$, probes: f, ϵ_f, ϵ_g
Output: g probe, history clusters H', tests of interest T', probed predictions $p_f(T'|H')$

1. Compute the small (i.e. $|h| \leq 5, |t| \leq 2$) approximate system-dynamics matrix using Algorithm 1
2. Initialize the history probe parameters: $\theta^* = \varphi^*, \theta_i^*$
3. $maxClusterDistance = -\infty$
4. Line search: for all parameter values $\theta^k = \varphi^k, \theta_i^k$

   (a) if the distance between histories in a cluster is less than the maximum error

   $$\forall h_i, h_j \in H_i', |g(h_i) - g(h_j)| \leq \max_{T_i \in T'} |p_f(T_i|h_i) - p_f(T_i|h_j)|$$

   (b) Then calculate the distance between clusters for this parameter configuration

   $$currentClusterDistance = \sum_{i,j} [g(H_i') - g(H_j')]^2, \forall H_i', H_j' \text{ clusters}$$

   (c) If $currentClusterDistance > maxClusterDistance$, then:

   $$\theta^* = \theta^k, maxClusterDistance = currentClusterDistance$$

   (d) Advance to the next parameter configuration

Bibliography

Bellemare, M. and Precup, D. (2007). Context-driven predictions. In *Proceedings of IJCAI*, pages 250–255.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Bellman,MA.

Bowling, M., McCracken, P., James, M., Neufeld, J., and Wilkinson, D. (2006). Learning predictive state representations using non-blind policies. In *Proceedings of ICML*, pages 129–136.

Dearden, R., Friedman, N., and Andre, D. (1999). Model based bayesian exploration. In *Proceedings of UAI*, pages 150–159.

Dempster, A. P., Laird, M. N., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39:1–22.

Hundt, C., Panangaden, P., Pineau, J., and Precup, D. (2006). Representing systems with hidden state. In *AAAI*.

Jaeger, H. (1998). Discrete-time, discrete-valued observable operator models: a tutorial. Technical report.

James, M. R. and Singh, S. (2005). Learning predictive state representations in dynamical systems without reset. In *Proceedings of ICML*, pages 980–987.

James, M. R., Wolfe, B., and Singh, S. (2005). Combining memory and landmarks with predictive state representations. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, pages 734–739.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1995). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4(1):237–285.

Koop, A. (2007). *Investigating Experience: Temporal Coherence and Empirical Knowledge Representation*. PhD thesis, University of Alberta.

Littman, M., Sutton, R., and Singh, S. (2002). Predictive representations of state. In *Proceedings of NIPS*, pages 1555–1561.

Loch, J. and Singh, S. (1998). Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *Proceedings of ICML*, pages 323–331.

McCallum, A. (2005). *Reinforcement learning with selective perception and hidden state*. PhD thesis, Rochester University.

McCracken, P. and Bowling, M. (2006). Online discovery and learning of predictive state representations. In *Proceedings of NIPS*, pages 875–882.

Ng, A. Y. and Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14*.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc.

Rafols, E. J., Ring, M. B., Sutton, R. S., and Tanner, B. (2005). Using predictive representations to improve generalization in reinforcement learning. In *Proceedings of IJCAI*, pages 835–840.

Rivest, R. L. and Schapire, R. E. (1994). Diversity-based inference of finite automata. *Journal of the ACM*, 41(3):555–589.

Rosencrantz, M., Gordon, G., and Thrun, S. (2004). Learning low dimensional predictive representations. In *Proceedings of ICML*, pages 695–702.

Shatkay, H. and Kaelbling, L. (1997). Learning topological maps with weak local odometric information. In *Proceedings of IJCAI*, pages 920–929.

Singh, S., James, M., and Rudary, M. R. (2004). Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of UAI*, pages 512–519.

Singh, S., Littman, M., Jong, N., Pardoe, D., and Stone, P. (2003). Learning predictive state representations. In *Proceedings of ICML*, pages 712–719.

Singh, S. P. and Sutton, R. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

Sondik, E. J. (1971). *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford, California.

Still, S. (2009). Information theoretic approach to interactive learning. *European Physics Letters*, 85.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S. and Tanner, B. (2005). Temporal-difference networks. In *Proceedings of NIPS*, pages 1377–1384.

Talvitie, E. and Singh, S. (2008). Simple local models for complex dynamical systems. In *Proceedings of NIPS*.

Talvitie, E. and Singh, S. (2009). Maintaining predictions over time without a model. In *IJCAI*, pages 1249–1254.

Wolfe, A. P. and Barto, A. G. (2006). Decision tree methods for finding reusable mdp homomorphisms. In *Proceedings of AAAI*, pages 530–535.

Wolfe, B., James, M. R., and Singh, S. (2008). Approximate predictive state representations. In *Proceedings of AAMAS*, pages 363–370.