Hidden-Mode Markov Decision Processes

Samuel P. M. Choi pmchoi@cs.ust.hk
Dit-Yan Yeung dyyeung@cs.ust.hk
Nevin L. Zhang lzhang@cs.ust.hk
Department of Computer Science, Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

Traditional reinforcement learning (RL) assumes that environment dynamics do not change over time (i.e., stationary). This assumption, however, is not realistic in many real-world applications. In this paper, a formal model for an interesting subclass of nonstationary environments is proposed. The environment model, called *hidden-mode Markov decision process* (HM-MDP), assumes that environmental changes are always confined to a small number of hidden *modes*. A mode basically indexes a Markov decision process (MDP) and evolves with time according to a Markov chain.

HM-MDP is a special case of *partially observable Markov decision processes* (POMDP). Nevertheless, modeling an HM-MDP environment via the more general POMDP model unnecessarily increases the problem complexity. In this paper the conversion from the former to the latter is discussed.

Learning a model of HM-MDP is the first step of two steps for nonstationary model-based RL to take place. This paper shows how model learning can be achieved by using a variant of the Baum-Welch algorithm. Compared with the POMDP approach, empirical results reveal that the HM-MDP approach significantly reduces computational time as well as the required data.

1 Introduction

Markov decision process (MDP) [1] is a mathematical framework originated from stochastic optimal control. The generality and flexibility of MDP make it applicable to a wide range of problems, and thus MDP has received much attention from other disciplines such as artificial intelligence. There are, however, three necessary conditions that limit the usefulness of conventional MDPs, namely: 1) The environment model must be given in advance (completely-known environment). 2) The environment states have to be completely observable (completely-observable states, implying Markovian environment). 3) The environment parameters are assumed to be fixed (stationary environment).

In the past, research efforts have been made towards relaxing the first two conditions, leading to different classes of problems as illustrated in Figure 1.

States of Environment	Model of Environment: Known	Model of Environment: Unknown
Completely Observable	MDP	Traditional RL
Partially Observable	Partially Observable MDP	Hidden-state RL

Figure 1: Categorization into four related problems with different conditions. Note that the degree of difficulty increases from left to right and from upper to lower.

In this paper the first and third conditions are partially addressed. We first identify and formalize an interesting subclass of nonstationary environments, and then develop for it a model-learning algorithm. At the end of the paper, how the second condition could be relaxed will be discussed.

1.1 Nonstationary Environments

Traditional MDP problems typically assume that environment dynamics (i.e., MDP parameters) are always fixed (i.e., *stationary*). This assumption, however, is not realistic in many real-world applications. In elevator control [5], for instance, the passenger arrival and departure rates can vary significantly over one day, and should not be modeled by a fixed MDP.

Previous studies on nonstationary MDPs [13] presume that changes of the MDP parameters are exactly known in every time step. Given this assumption, solving nonstationary MDP problems is trivial, as the problem can be recast into a stationary one (with a much larger state space) by performing state augmentation. Nevertheless, extending the idea to incompletely-known environmental

changes (i.e., to the reinforcement learning framework) is far more difficult.

In fact, reinforcement learning (RL) [7; 17] in nonstationary environments is an impossible task if there exists no regularity in the way environment dynamics change. Hence, some degree of regularity must be assumed. There exists a few heuristic approaches along this line [8; 16; 17], but to the best of our knowledge, only one formal model [6] has been proposed. The model, due to Dayan and Sejnowski, assumes that the environmental changes are governed by a known probability distribution.

1.2 Our Proposed Model

This paper proposes another formal environment model. Our model is inspired by observations from an interesting subclass of nonstationary RL tasks. The properties of such nonstationary environments are:

Property 1: Environmental changes are confined to a small number of *environment modes*. These modes are stationary environments that possess distinct environment dynamics and require different control policies. At any time instant, the environment is assumed to be in exactly one of these modes. This concept of modes seems to be applicable to many, though not all, real-world tasks. In an elevator control problem, for example, the system might operate in a morning-rush-hour mode, an evening-rush-hour mode and a non-rush-hour mode. One can also imagine similar modes for other real-world control tasks, such as traffic control, dynamic channel allocation [15], and network routing [2].

Property 2: Unlike states, modes cannot be directly observed; the current mode can only be estimated according to the past state transitions. It is analogous to the elevator control example in that the passenger arrival rate and pattern can only be partially observed through the occurrence of pick-up and drop-off requests.

Property 3: Mode transitions are stochastic events and are independent of the control system's response. In the elevator control problem, for instance, the events that change the current mode of the environment could be an emergency meeting in the adminstrative office, or a tea break for the staff on the 10th floor. Obviously, the elevator's response has no control over the occurrence of these events.

Property 4: Mode transitions are relatively infrequent. In other words, a mode is more likely to retain for some time before switching to another one. Take the emergency meeting as an example, employees on different floors take time to arrive at the adminstrative office, and thus would generate a similar traffic pattern (drop-off requests on the same floor) for some period of time.

Property 5: The number of states is often substantially larger than the number of modes. This is a common property in many real-world applications. In the elevator control example, the state space comprises of all possible combinations of elevator positions, pick-up and drop-off requests, and certainly would be huge. On the other hand, the mode space could be small. For instance, an elevator control system can simply have the three modes as described above to approximate the reality.

Based on these properties, an environment model is now proposed. The whole idea is to introduce a mode variable to capture environmental changes. Each mode specifies an MDP and hence completely determines the current state transition function and reward function (property 1). A mode, however, is not directly observable (property 2), and evolves with time according to a Markov process (property 3). The model is therefore called *hidden-mode model*.

Note that the hidden-mode model does not impose any constraint to satisfy properties 4 and 5. In other words, the model is flexible enough to work for environments where these two properties do not hold. Nevertheless, as will be shown later, these properties can be utilized to help the learning in practice.

The hidden-mode model, however, also has its limitations. For instance, one may argue that the mode of an environment should preferably be continuous. While this is true, for tractability, we assume the mode is discrete. This implies that our model, as for any other model, is only an abstraction of the real world. Moreover, we assume that the number of modes is known in advance. We will seek to relax these assumptions in our future research.

1.3 Related Work

Our hidden-mode model is closely related to Dayan and Sejnowski's model. Although our model is more restrictive in terms of representational power, it involves much fewer parameters and is thus easier to learn. Besides, other than the number of possible modes that should be known in advance, we do not assume any other knowledge about the way environment dynamics change[1]. Dayan and Sejnowski, on the other hand, assume that one knows precisely how the environment dynamics change.

The hidden-mode model can also be viewed as a special case of the hidden-state model, or *partially observable Markov decision process* (POMDP). As will be shown later, a hidden-mode model can always be represented by a hidden-state model through state augmentation. Nevertheless, modeling a hidden-mode environment via a hidden-state model will unnecessarily increase the problem complexity. In this paper, the conversion from the former to the latter is discussed in Section 2.2.

1.4 Our Focus

In order for RL to take place, one may choose between the model-based and model-free approaches. Model-based RL first acquires an environment model and then,

[1] That is, the transition probabilities of the Markov process governing mode changes, though fixed, are unknown in advance.

from which, an optimal policy is derived. Model-free RL, on the other hand, learns an optimal policy directly through its interaction with the environment. This paper is primarily concerned with the first part of the model-based RL approach, i.e., how a hidden-mode model can be learned from its experience. Our model-learning algorithm is based on the Baum-Welch algorithm commonly used for learning hidden Markov models (HMM) [14]. We will address the policy learning issue in a separate paper.

2 Hidden-Mode Markov Decision Processes

This section presents our hidden-mode model. Basically, a hidden-mode model is defined as a finite set of MDPs that share the same state space and action space, with possibly different transition functions and reward functions. The MDPs correspond to different modes in which a system operates. States are completely observable and their transitions are governed by an MDP. In contrast, modes are not directly observable and their transitions are controlled by a Markov chain. We refer to such a process as a *hidden-mode Markov decision process* (HM-MDP). Figure 2 gives an example of HM-MDP.

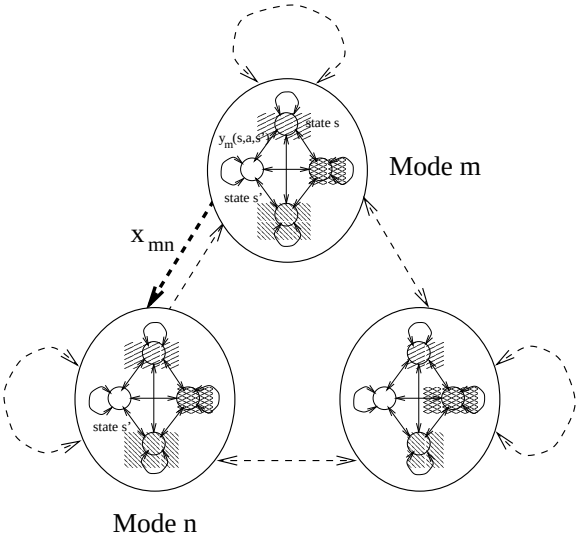


Figure 2: A 3-mode, 4-state, 1-action HM-MDP. The values x_{mn} and $y_m(s, a, s')$ are the mode and state transition probabilities respectively.

2.1 Formulation

More formally, an HM-MDP is defined as an 8-tuple $(Q, S, A, X, Y, R, \Pi, \Psi)$, where Q, S and A represent the sets of modes, states and actions respectively; the mode transition function X maps mode m to n with a fixed probability x_{mn}; the state transition function Y defines transition probability, $y_m(s, a, s')$, from state s to s' given mode m and action a; the stochastic reward function R returns rewards with the mean value $r_m(s, a)$; Π and Ψ denote the prior probabilities of the modes and of the states respectively. The evolution of modes and states is depicted in Figure 3.

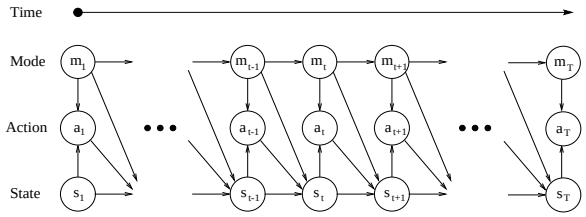


Figure 3: The evolution of an HM-MDP. Each node represents a mode, action or state variable. The arcs indicate dependencies between the variables.

2.2 Reformulating HM-MDP as POMDP

HM-MDP is a subclass of POMDP. In other words, it is always possible to reformulate the former as a special case of the latter. In particular, one may take an ordered pair of any mode and observable state in the HM-MDP as a hidden state in a POMDP, and any observable state of the former as an observation of the latter. Suppose the observable states s and s' are in modes m and n respectively. These two HM-MDP states together with their corresponding modes form two hidden states $< m, s >$ and $< n, s' >$ for its POMDP counterpart. The transition probability from $< m, s >$ to $< n, s' >$ is then simply the mode transition probability x_{mn} multiplied by the state transition probability $y_m(s, a, s')$. For an M-mode, N-state, K-action HM-MDP, the equivalent POMDP thus has N observations and MN hidden states.

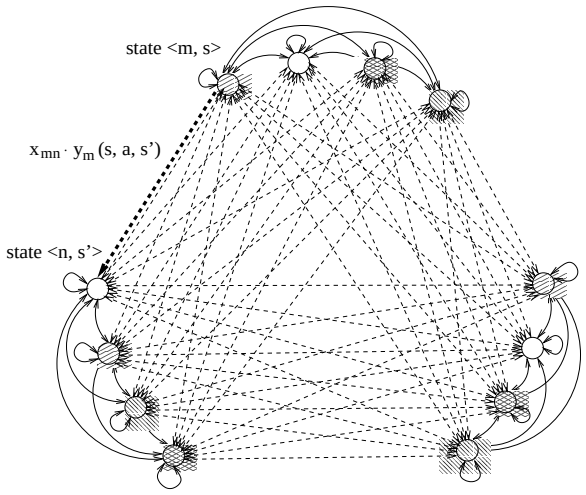


Figure 4: Reformulating a 3-mode, 4-state, 1-action HM-MDP described in Figure 2 as an equivalent POMDP. For brevity, exact transition probability values are not shown. State s of mode m in Figure 2 is relabeled as $< m, s >$. Note that HM-MDP has much fewer model parameters than its POMDP counterpart.

Figure 4 demonstrates the reformulation for a 3-mode, 4-state, 1-action HM-MDP and compares the model complexity with its equivalent POMDP. Note that most state transition probabilities (dashed lines) in the POMDP are collapsed into mode transition probabilities in the HM-MDP through parameter sharing. This saving is significant. In the example, the HM-MDP model has a total of 57 transition probabilities, while its POMDP counterpart has 144. In general, an HM-MDP contains much fewer parameters ($N^2MK + M^2$) than its corresponding POMDP (M^2N^2K).

3 Learning a Hidden-mode Model

There are now two ways to learn a hidden-mode model. One may learn either an HM-MDP, or an equivalent POMDP instead. In this section, we first briefly discuss how the latter can be achieved by a variant of the Baum-Welch algorithm, and then develop a similar algorithm for HM-MDP.

3.1 Learning a POMDP Model

Traditional research in POMDP [12; 9] assumes a known environment model and is concerned with finding an optimal policy. Chrisman [4] was the first to study the learning of POMDP models from experience.

Chrisman's work is based on the Baum-Welch algorithm, which was originally proposed for learning HMMs [14]. Based on the fact that a POMDP can be viewed as a collection of HMMs, Chrisman proposed a variant of the Baum-Welch algorithm for POMDP. This POMDP Baum-Welch algorithm requires $\Theta(M^2N^2T)$ time and $\Theta(M^2N^2K)$ storage for learning an M-mode, N-state, K-action HMMDP, given T data items. However, Chrisman's algorithm does not learn the reward function. One possible extension is to estimate the reward function by averaging the obtained rewards, weighted by the estimated state certainty. The effectiveness of the algorithm will be examined in the next section.

3.2 Learning an HM-MDP Model

We now extend Chrisman's method to the learning of an HM-MDP model. The algorithm is basically identical to the POMDP version, except that auxiliary variables are redefined. Intuitively, the new algorithm estimates model parameters based on the expected counts of the mode transitions, which are computed by a set of auxiliary variables. The resultant algorithm, HM-MDP Baum-Welch, is presented in Figure 5.

HMMDP Baum-Welch requires only $\Theta(M^2T)$ time and $\Theta(M^2 + MN^2K)$ storage, which gives a significant reduction compared with the POMDP approach.

4 Empirical Studies

This section empirically examines the POMDP Baum-Welch[2] and HM-MDP Baum-Welch algorithms in terms of the required data size and time. Experiments on various model sizes and settings were conducted. The results are quite consistent. In the following, some details of a typical run are presented for illustration.

4.1 Experimental Setting

The experimental model is a randomly generated HM-MDP with 3 modes, 10 states and 5 actions. Note that this HM-MDP is equivalent to a fairly large POMDP, with 30 hidden states, 10 observations and 5 actions. In order to simulate the infrequent mode changes, each mode is set to have a minimum probability of 0.9 in looping back to itself. In addition, each state of the HM-MDP has 3 to 8 non-zero transition probabilities, and rewards are uniformly distributed between $r_m(s,a) \pm 0.1$. This reward distribution, however, is not disclosed to the learning agents.

The experiments were run with data of various sizes, using the same initial model. The model was also randomly generated in the form of HM-MDP. To ensure fairness, the equivalent POMDP model was used for POMDP Baum-Welch learning. For each data set, the initial model was first loaded, and the selected algorithm iterated until the maximum change of the model parameters was less than a threshold of 0.0001. After the algorithm terminated, the model learning time was recorded, and the model estimation errors were computed. The experiment was then repeated for 11 times with different random seeds in order to compute the median.

4.2 Performance Measure

The HM-MDP and POMDP Baum-Welch algorithms learn a hidden-mode model in different representations. To facilitate comparison, all models were first converted into POMDP form. Model estimation errors can then be measured in terms of the minimum difference between the learned model and the actual model. As the state indices for the learned model might be different from the actual one, a renumbering of the state indices is needed. In our experiment, an indexing scheme that minimizes the sum of the squares of differences on the state transition probabilities between the learned and the actual models was used (provided the constraints on the observation probabilities are preserved). Figure 6 (a) and (b) report respectively the sum of the squares of differences on the transition function and on the reward function using this indexing scheme.

Regarding the computational requirement of the algorithms, the total CPU running time was measured on a SUN Ultra I workstation. Table 1 reports the model learning time in seconds.

4.3 Empirical Results

Conclusions can now be drawn. Generally speaking, both algorithms can learn a more accurate environment model as the data size increases (Figure 6). This result is not surprising since both algorithms are statistically-based, and hence their performances rely largely on the amount of data provided. When the training data size is

[2] Chrisman's algorithm also attempts to learn a minimal possible number of states. This paper concerns only with the learning of the model parameters.

Approach	Data Set Size				
	1000	2000	3000	4000	5000
HM-MDP	4.60	18.72	15.14	9.48	10.07
POMDP	189.40	946.78	2164.20	3233.56	4317.19

Table 1: CPU time in seconds

too small (less than 1000 in this case), both algorithms perform about equally poorly. However, as the data size increases, HM-MDP Baum-Welch improves substantially faster than POMDP Baum-Welch.

Our experiment reveals that HM-MDP Baum-Welch was able to learn a fairly accurate environment model with a data size of 2500. POMDP Baum-Welch, on the contrary, needs a data size of 20000 (not shown) in order to achieve a comparable accuracy. In fact, in all the experiments we conducted, HM-MDP Baum-Welch always required a much smaller data set than the POMDP Baum-Welch. We believe that this result holds in general because in most cases, an HM-MDP consists of fewer free parameters than its POMDP counterpart.

In terms of computational requirement, HM-MDP Baum-Welch is much faster than POMDP Baum-Welch (Table 1). We believe this is also true in general for the same reason described above. In addition, computational time is not necessarily monotonically increasing with the data size. It is because the total computations depend not only on the data size, but also on the number of iterations being executed. From our experiments, we noticed that the number of iterations tends to decrease as the data size increases.

5 Discussions

The usefulness of a model depends on the validity of the assumptions made. In this section, we revisit the assumptions of HM-MDP, discuss the issues involved, and shed some light on its applicability to real-world nonstationary tasks. Some possible extensions are also discussed.

Modeling a nonstationary environment as a number of distinct MDPs.

MDP is a flexible framework that has been widely adopted in various applications. Among these there exist many tasks that are nonstationary in nature and are more suitable to be characterized by several, rather than a single, MDPs. The introduction of distinct MDPs for modeling different modes of the environment is a natural extension to those tasks.

One advantage of having distinct MDPs is that the learned model is more comprehensive: each MDP naturally describes a mode of the environment. In addition, this formulation facilitates the incorporation of prior knowledge into the model initialization step.

States are directly observable while modes are not.

While modes are not directly observable, they may be estimated by observing the past state transitions. It is a crucial (fortunately still reasonable) assumption that one needs to make.

Although states are assumed to be observable, it is possible to extend the model to allow partially observable states, i.e., to relax the second condition mentioned in Section 1. In this case, the extended model would be equivalent in representational power to a POMDP. This could be proved by showing the reformulation of the two models in both directions.

Mode changes are independent of the agent's responses.

This property may not always hold for all real-world tasks. In some applications, such as learning in a multi-agent environment or performing tasks in an adversary environment, the agent's actions might affect the state as well as the environment mode. In that case, an MDP instead of a Markov chain should be used to govern the mode transition process. Obviously, the use of a Markov chain involves fewer parameters and is thus preferable whenever possible.

Mode transitions are relatively infrequent.

This is a property that generally holds in many applications. In order to characterize this property, a large transition probability for a mode looping back to itself can be used. Note that this is introduced primarily from a practical point of view, but is not a necessary condition for our model. In fact, we have tried to apply our model-learning algorithms to problems in which this property does not hold. We find that our model still outperforms POMDP, although the required data size is typically increased for both cases.

Using high self-transition probabilities to model rare mode changes may not always be the best option. In some cases mode transitions are also correlated with the time of a day (e.g. busy traffic in lunch hours). In this case, time (or the mode sequence) should be taken into account for identifying the current mode. One simple way to model this property is to strengthen left-to-right transitions between modes, as in the left-to-right HMMs.

Number of states is substantially larger than the number of modes.

This nice property significantly reduces the number of parameters in HM-MDP compared to that in POMDP, and makes the former more applicable to real-world nonstationary tasks.

The number of states can be determined by the learning agent. States can be distinguished by, for instance, transition probabilities, mean rewards, or utilities. McCallum [11] has detailed discussions on this issue.

Likewise, the number of modes can be defined in various ways. After all, modes are used to discern changes of environment dynamics from noise. In practice, introduction of a few modes is sufficient for boosting the

system performance. More modes might help further, but not necessarily significantly. A trade-off between performance and response time must thus be decided. In fact, determining the optimal number of modes is an important topic that deserves further studies.

6 Conclusion

A formal model for a special case of nonstationary environments is proposed. The environment model, named HM-MDP, is motivated by the observation that the changes of most real-world nonstationary RL tasks are often confined to a small number of modes. In this paper we also show that an HM-MDP is a special case of POMDP. Nevertheless, HM-MDP requires fewer model parameters and is a more natural formulation for many real-world nonstationary tasks. Besides the model formulation, we develop a variant of the Baum-Welch algorithm for HM-MDP model learning. Empirical results confirm that the HM-MDP approach yields a more accurate environment model with less data and time.

7 Future Work

Despite the encouraging results obtained, there are a number of issues that need to be addressed in order to broaden the applicability of HM-MDP. First, the number of modes is currently assumed to be known. In some situations, choosing the right number of modes can be difficult. Hence, we are now investigating the possibility of using Chrisman's or McCallum's hidden-state-splitting techniques [4; 10] to remove this limitation. Next, thus far we are concerned only with the model learning procedures, but have not addressed how an optimal policy can be computed. Although in principle it can be achieved indirectly via any POMDP algorithm (e.g. [3]), a more efficient algorithm based on the model-based approach is possible. We will address this issue in a separate paper. Third, an accurate reward function is shown to be crucial for learning an optimal policy. It is noticed that our intuitive extension of Chrisman's POMDP model-learning algorithm does not always lead to an accurate estimate. In some cases, the estimated mean rewards tend to be averaged out. To further enhance the learning accuracy, the functional form of the reward function is needed. Finally, the exploration-exploitation issue is currently ignored. In our future work, we will address this important issue and apply our model to real-world nonstationary tasks.

References

[1] R. E. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

[2] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: a reinforcement learning approach. In *Advances in Neural Information Processing Systems 6*, 1994.

[3] A. R. Cassandra, M. L. Littman, and N. Zhang. Incremental pruning: A simple, fast, exact algorithm for partially observable Markov decision processes. In *Uncertainty in Artificial Intelligence*, Providence, RI, 1997.

[4] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI-92*, 1992.

[5] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, 1996.

[6] P. Dayan and T. J. Sejnowski. Exploration bonuses and dual control. *Machine Learning*, 25(1):5–22, Oct. 1996.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

[8] M. L. Littman and D. H. Ackley. Adaptation in constant utility non-stationary environments. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[9] W. S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

[10] A. McCallum. Overcoming incomplete perception with utile distinction memory. In *Tenth International Machine Learning Conference*, Amherst, MA, 1993.

[11] A. McCallum. *Reinforcement Learning with Selective Perception and Hidden State.* PhD thesis, University of Rochester, Dec. 1995.

[12] G. E. Monahan. A survey of partially observable Markov decision processes: Theory, models and algorithms. *Management Science*, 28:1–16, 1982.

[13] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, 1994.

[14] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), Feb. 1989.

[15] S. Singh and D. P. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems*, 1996.

[16] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, 1990.

[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, 1998.

Given a collection of data and an initial model parameter vector $\bar{\theta}$.

repeat

$\theta = \bar{\theta}$

Compute forward variables α_t.

$$\alpha_1(i) = \psi_{s_1} \qquad \forall\, i \in Q$$
$$\alpha_2(i) = \pi_i\, \psi_{s_1}\, y_i(s_1, a_1, s_2) \qquad \forall\, i \in Q$$
$$\alpha_{t+1}(j) = \sum_{i \in Q} \alpha_t(i)\, x_{ij}\, y_j(s_t, a_t, s_{t+1}) \qquad \forall\, i \in Q$$

Compute backward variables β_t.

$$\beta_T(i) = 1 \qquad \forall\, i \in Q$$
$$\beta_t(i) = \sum_{j \in Q} x_{ij}\, y_j(s_t, a_t, s_{t+1})\, \beta_{t+1}(j) \qquad \forall\, i \in Q$$
$$\beta_1(i) = \sum_{j \in Q} \pi_j\, y_j(s_1, a_1, s_2)\, \beta_2(j) \qquad \forall\, i \in Q$$

Compute auxiliary variables ξ_t and γ_t.

$$\xi_t(i,j) = \frac{\alpha_t(i)\, x_{ij}\, y_j(s_t, a_t, s_{t+1})\, \beta_{t+1}(j)}{\sum_{k \in Q} \alpha_T(k)} \qquad \forall\, i, j \in Q$$
$$\gamma_t(i) = \sum_{j \in Q} \xi_{t+1}(i,j) \qquad \forall\, i \in Q$$

Compute the new model parameter $\bar{\theta}$.

$$\bar{x}_{ij} = \frac{\sum_{t=2}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}$$

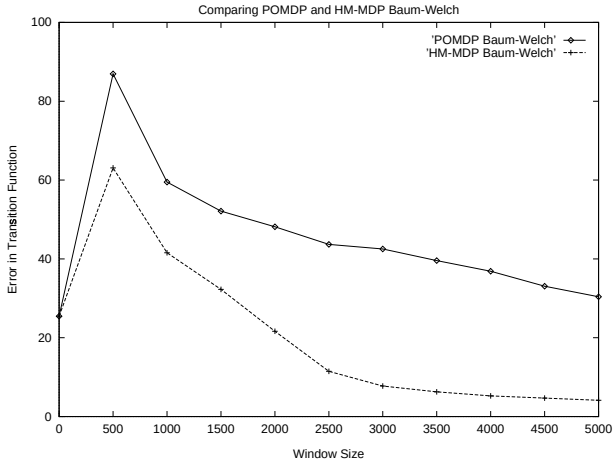
$$\bar{y}_i(j,k,l) = \frac{\sum_{t=1}^{T-1} \gamma_t(i)\, \delta(s_t, j)\, \delta(s_{t+1}, l)\, \delta(a_t, k)}{\sum_{l \in S} \sum_{t=1}^{T-1} \gamma_t(i)\, \delta(s_t, j)\, \delta(s_{t+1}, l)\, \delta(a_t, k)}$$

$$\bar{r}_i(j,k) = \frac{\sum_{t=1}^{T-1} \gamma_t(i)\, \delta(a_t, k)\, \delta(s_t, j)\, r_t}{\sum_{t=1}^{T-1} \gamma_t(i)\, \delta(a_t, k)\, \delta(s_t, j)}$$

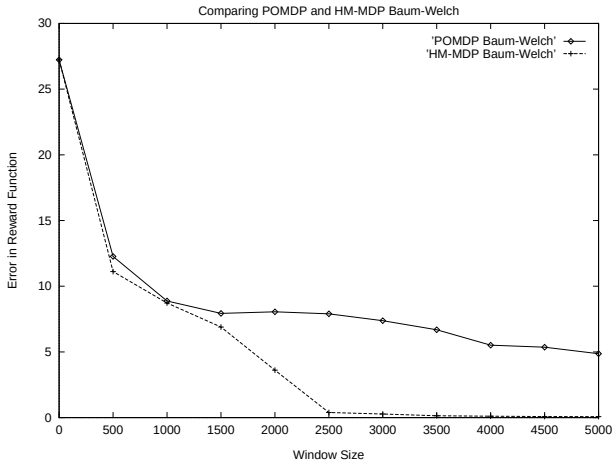
$$\bar{\pi}_i = \gamma_1(i)$$

until $\max_i |\bar{\theta}_i - \theta_i| < \epsilon$

Figure 5: HM-MDP Baum-Welch Algorithm



(a) Error in Transition Function



(b) Error in Reward Function

Figure 6: Model learning errors in terms of the transition probabilities and rewards. All environment models were in their POMDP form for comparison. The errors were measured by summing the squares of differences on the state transition probabilities and on the reward function respectively.