Constructing States for Reinforcement Learning

M. M. Hassan Mahmud HAMAHMUD42@GMAIL.COM
School of Computer Science, The Australian National University, Canberra, 0200 ACT, Australia

Abstract

POMDPs are the models of choice for reinforcement learning (RL) tasks where the environment cannot be observed directly. In many applications we need to learn the POMDP structure and parameters from experience and this is considered to be a difficult problem. In this paper we address this issue by modeling the hidden environment with a novel class of models that are less expressive, but easier to learn and plan with than POMDPs. We call these models *deterministic Markov models* (DMMs), which are deterministic-probabilistic finite automata from learning theory, extended with actions to the sequential (rather than i.i.d.) setting. Conceptually, we extend the Utile Suffix Memory method of McCallum to handle long term memory. We describe DMMs, give Bayesian algorithms for learning and planning with them and also present experimental results for some standard POMDP tasks and tasks to illustrate its efficacy.

1. Introduction

In this paper we derive a method to estimate the hidden structure of environments in general reinforcement learning problems. In such problems, at each discrete time step the agent takes an action and in turn receives just an observation and a reward. The goal of the agent is to take actions (i.e. plan) to maximize its future time-averaged discounted rewards (Bertsekas & Shreve, 1996). Clearly, we need to impose some structure on the environment to solve this problem.

The most popular approach in machine learning to do this is by assuming that the environment is a partially observable Markov decision process (POMDP), which are (essentially) hidden Markov models with actions. Given the model (structure and parameters) of a POMDP, there exists effective heuristic algorithms for planning (see the survey (Ross et al., 2008)), although exact planning is undecidable in general (Madani et al., 2003). However, in many important problems, the POMDP model is not available a-priori and has to be learned from experience. While there are some promising new approaches (e.g. (Doshi-Velez, 2009) using HDP-POMDPs), this problem is as yet unsolved (and in fact NP-hard even under severe constraints (Sabbadin et al., 2007)).

One way to bypass this difficult learning problem is to consider simpler environment models. In particular, in this paper we assume that each history *deterministically* maps to one of finitely many states and this state is a sufficient statistic of the history (Mccallum, 1995; Shalizi & Klinkner, 2004; Hutter, 2009). Given this history-state map the environment becomes a MDP which can then be used to plan. So the learning problem now is to learn this map. Indeed, the well known USM algorithm (Mccallum, 1995) used Prediction Suffix Trees (Ron et al., 1994) for these maps (each history is mapped to exactly one leaf/state) and was quite successful in benchmark POMDP domains. However, PSTs lack long term memory and had difficulty with noisy environments and so USM was not followed up on for the most part. In our work we consider a Bayesian setup and replace PSTs with finite state machines and endow the agent with long term memory. The resulting model is a proper subclass of POMDPs, but hopefully maintains the computational simplicity and efficiency that comes with considering deterministic history state maps.

We note that belief states of POMDPs are also deterministic functions of the history. But this state space is infinite and so POMDP models learning algorithms try to estimate the hidden states (see for instance (Doshi-Velez, 2009)). As a result, these methods are quite different from algorithms using deterministic history-state maps. Other notable meth-

Appearing in *Proceedings of the 27^{th} International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

ods for learning the environment model include PSRs (Littman et al., 2002) – unfortunately we lack space and do not discuss these further. Superficially, finite state controllers for POMDPs (Kaelbling et al., 1998) seem closely related to our work but these are not quite model learning algorithms. They are (powerful) planning algorithms that assume a hidden but known environment model (at the very least, implicitly, an estimate of the number of hidden states).

We now proceed as follows. We define a general RL environment and then our model, the deterministic Markov model (DMM), and show how to use it to model the environment, infer the state given a history and compute the optimal policy. We then describe our Bayesian inference framework and then derive a Bayesian, heuristic Monte-Carlo style algorithm for model learning and planning. Finally, we describe experiments on standard benchmark POMDP domains and some novel domains to illustrate the advantage of our method over USM. Due to lack of space formal proofs of our results are given in (Mahmud, 2010). Our focus here is on motivating our approach via discussion and experiments.

2. Modeling Environments

To recap, we model a general RL environment by our model, the DMM, and then use the MDP derived from the DMM to plan for the problem. In the following we introduce notation (Sect. 2.1), define a general RL environment (Sect. 2.2), define our model, the DMMs (Sect. 2.3) and show how they can model the environment and construct the requisite MDP to plan with (Sect. 2.4). We then describe the DMM inference criterion and the learning algorithm in Sect. 3 and 4.

2.1. Preliminaries

$E_{P(x)}[f(x)]$ denotes the expectation of the function f with respect to distribution P. We let $\mathcal{A}$ be a finite set of actions, $\mathcal{O}$ a finite set of observations and $\mathcal{R} \subset \mathbb{R}$ a finite set of rewards. We set $\mathcal{H} := (\mathcal{A} \times \mathcal{O})^*$ to be the set of histories and $\gamma \in [0, 1)$ to be a fixed discount rate. We now need some notation for sequences over finite alphabets. $x_{0:n}$ will denote a string of length $n+1$ and $x_{<n} \equiv x_{0:n-1}$ will denote a string of length n. $x_{i:j}$ will denote elements i to j inclusive, while x_i will denote the the i^{th} element of the sequence. The indices will often be time indices (but not always). If there are two strings $x_{0:n}$ and $y_{0:n}$, we will use $xy_{0:n}$ to denote the interleaved sequence $x_0y_0x_1y_n \ldots x_ny_n$; we will use xy_i and $xy_{i:j}$ to denote x_iy_i and $x_iy_i \ldots x_jy_j$ respectively. Finally, λ will denote the empty string. As an example, each element of $\mathcal{H}$ is of the form $ao_{0:n}$ and λ denotes the empty history.

2.2. General RL Environments

A general RL environment, denoted by grle, is defined by a tuple $(\mathcal{A}, \mathcal{R}, \mathcal{O}, RO, \gamma)$ where all the quantities except RO are as defined above. RO defines the dynamics of the environment: at step t when action a is applied, the next reward×observation is selected according to probability $RO(ro|h, a)$ where $h = ao_{0:t-1}$ is the history before step t. We will write the marginals over $\mathcal{R}$ and $\mathcal{O}$ w.r.t. RO by R and O respectively.

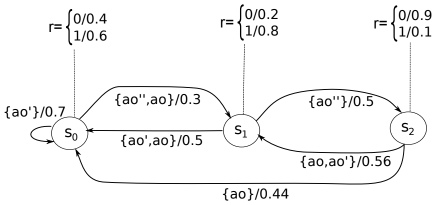

Figure 1. A DMM with $\mathcal{A} = \{a\}$, $\mathcal{R} := \{0, 1\}$ and $\mathcal{O} := \{o, o', o''\}$. The edges are labeled with $z \in \mathcal{A} \times \mathcal{O}$ that cause the transition along with the probability for that transition (parameters $\vec{\phi}_{s,a}$). The reward distributions for each state appear above each state (parameters $\vec{\theta}_{s,a}$).

The actions at each step are chosen using a policy $\pi : \mathcal{H} \to \mathcal{A}$; the *value function* of π is defined as:

$$V^\pi(h) = E_{R(r|h,a)}(r) + \gamma E_{O(o|h,a)}[V^\pi(hao)] \qquad (1)$$

where $a := \pi(h)$. The goal in RL problems is to learn the optimal policy π^* which satisfies $V^{\pi^*}(h) \geq V^\pi(h)$ for each policy π and history h. In particular the value function of this policy is given by:

$$V^{\pi^*}(h) := V^*(h) := \max_a \{E_{R(r|h,a)}(r) + \gamma E_{O(o|h,a)}[V^*(hao)]\} \qquad (2)$$

The existence of these functions follow via standard results (Bertsekas & Shreve, 1996). For the sequel we fix a particular RL environment and denote it by grle $:= (\mathcal{A}, \mathcal{R}, \mathcal{O}, RO, \gamma)$.

2.3. Deterministic Markov Models

Our model is a graphical model and as such defined by two components – a structure and associated parameters/probabilities (see Fig. 1). We use DMM to refer to the structure as during Bayesian learning we will marginalize out the parameters leaving us with the marginal likelihood of the just the structure. We will use the term 'model' to refer to DMM+parameters.

The DMM. A DMM ξ is a standard deterministic finite state automaton (Hopcroft et al., 2006) and is defined by the tuple $(q_\circ, \mathcal{S}, \Sigma, \delta)$. Here, $q_\circ$ is the start state, $\mathcal{S}$ is the set of states, $\Sigma = \mathcal{A} \times \mathcal{O}$ is the edge-label alphabet and $\delta : \mathcal{S} \times \Sigma \to \mathcal{S}$ is the transition function. When ξ is at state s, it transitions to state s' on input $z \in \Sigma$ iff $\delta(s, z) = s'$. The DMM always starts off at state $q_\circ$ and, overloading δ to denote its transitive closure, $\delta(s, h) = s'$ means that ξ transitions to s' from s when h is given as input. We define $\mathcal{H}(s) := \{h | h \in \mathcal{H}, \delta(q_\circ, h) = s\}$ as the set of histories that result in state s.

The Parameters. The parameters of our model are $\{\vec{\theta}_{s,a}\}$ $(:= \vec{\theta})$ and $\{\vec{\phi}_{s,a}\}$ $(:= \vec{\phi})$ indexed over $\mathcal{S}$ and actions $\mathcal{A}$. $Pr(r|\vec{\theta}_{s,a}, \xi)$ and $Pr(s'|\vec{\phi}_{s,a}, \xi)$ are, respectively, the probability that the next reward is r and next state is s when ξ at state s and action a is chosen. Since both $\mathcal{R}$ and $\mathcal{S}$ are finite spaces, both the above distributions are multinomials: $\sum_r \vec{\theta}_{s,a}(r) = 1$ and $Pr(r|\vec{\theta}_{s,a}, \xi) = \vec{\theta}_{s,a}(r)$ and similarly for the $\vec{\phi}_{s,a}$s.

Model Dynamics. The dynamics of our model is deterministic or probabilistic depending on the conditioning values (see also (Vidal et al., 2005)). If ξ is at state s and some action a is taken it transitions to state s' with probability $Pr(s'|\vec{\phi}_{s,a}, \xi)$. However, if the next observation o is also given, the dynamics is deterministic and ξ transitions to state s' iff $\delta(s, ao) = s'$. Therefore, given a history $ao_{0:n}$, there is exactly one state sequence $s_{0:n}$ that ξ could have transitioned through (with $\delta(q_\circ, ao_{0:k}) = s_k$). In other words, DMMs are DPFAs from learning theory (Vidal et al., 2005) but extended with actions to a sequential setting[1]. The states of the DMM can also be interpreted as discretization of belief states in a POMDP. This seems like a rich avenue of future research, but we do not pursue this here. We are now ready to define how DMMs model RL environments.

Modeling Environments. Our central modeling assumption is that for the grle we are trying to learn, there exists a ξ^g (g stands for 'generating') that satisfies for all states s, action a, history $h \in \mathcal{H}(s)$ and reward r,

$$\exists \vec{\theta}^g_{s,a}, Pr(r|\vec{\theta}^g_{s,a}, \xi^g) = R(r|h, a) \tag{3}$$

DMMs do not model the observation distribution O because all we actually care about is the reward distribution (Hutter, 2009). And in fact, we may end up sacrificing reward prediction accuracy or learn a too complex model by trying to model task-irrelevant details of O. For example, in a domain with a million states where O is different at each state but R is the same everywhere, it will be a bad idea to try to learn O. A more realistic example is the fact that when crossing a street we want to model car movements, but not swaying of the trees lining the road.

We are *not* ignoring the observations: they determine the transition function δ and hence our model. So we only use observation in so far as they help predict rewards. But we do need to impose some additional consistency constraint to account for not modeling O – we assume ξ^g satisfies, for all s, s', a and $h \in \mathcal{H}(s)$,

$$\exists \vec{\phi}^g_{s,a}, Pr(s'|\vec{\phi}^g_{s,a}, \xi^g) = \sum_{o|hao \in \mathcal{H}(s')} O(o|h, a) \tag{4}$$

That is, for any s and $h \in \mathcal{H}(s)$, the probability of the state s' on action a must be equal to the cumulative probability of the observations o such that $\delta(s, ao) = s'$. Together, this motivates the following definition

Definition 1. *We say a DMM ξ computes the* grle *reward function R if it satisfies (3) and (4).*

So our central assumption is the existence of ξ^g computing R.

Final Remark. Trivially, from a *formal* perspective, DMMs are maximally general as all the history-state maps computable using finite time and memory must have a FSM representation. Whether this formal property translates to practical efficacy over wide variety of applications will need to be established by future development of the method and learning algorithms.

2.4. DMMs → MDPs and Optimal Policies

Each $\xi = (q_\circ, \mathcal{S}, \Sigma, \delta)$ & parameters $\vec{\theta}$, $\vec{\phi}$ defines an MDP (Bertsekas & Shreve, 1996) $\mathsf{mdp} := (\mathcal{A}, \mathcal{R}, \mathcal{S}, R_M, T, \gamma)$. The first 3 components of mdp are as defined above and R_M and T are the reward and state-transition distributions respectively

$$R_M(r|s, a) := Pr(r|\vec{\theta}_{s,a}, \xi) \tag{5}$$

$$T(s'|s, a) := Pr(s'|\vec{\phi}_{s,a}, \xi) \tag{6}$$

We denote the mdp corresponding to ξ^g by mdp^g.

Given a policy $\bar{\pi} : \mathcal{S} \to \mathcal{A}$ for mdp^g, the value function is defined as follows:

$$V^{\bar{\pi}}(s) := E_{R_M(r|s,a)}(r) + \gamma E_{T(s'|s,a)}[V^{\bar{\pi}}(s')]$$

where $a = \bar{\pi}(s)$. Using $\bar{\pi}$ we act in the grle as follows. If the history at step t is h, we choose $\bar{\pi}(s)$ as our action where $\delta(q_\circ, h) = s$. Note that if $h = h'ao$ and $q = \delta(q_\circ, h')$, it must be that $\delta(q, ao) = s$ and hence we

[1] DMMs are sequential because, unlike DPFAs, they do not have a terminal state and hence never 'stop'. So they define distributions over infinite sequences of rewards instead of reward strings (Shalizi & Klinkner, 2004). This is more appropriate for possibly non-episodic tasks.

can infer states of our model incrementally. It is now easy to show that an optimal policy for mdp^g is also an optimal policy for the grle. This fact follows from (3), (4), (5) and (6) after some algebraic manipulation (Mahmud, 2010). So we now state the following:

Theorem 1. *An optimal policy $\bar\pi^*$ for mdp^g is also an optimal policy for the* grle.

3. Inference of the Correct Model

In this section we define the likelihood function, posterior and predictive distributions for Bayesian learning of DMMs. In the next section we will develop a heuristic algorithm to learn these models and plan with them. We assume a fixed policy π for taking actions (not necessarily a MDP policy).

3.1. The Likelihood function

We assume that we are given a history + reward sequence $\bar a\bar o_{0:n}$ and $\bar r_{0:n}$ generated from the environment using π (that is $\bar a_k = \pi(\bar a\bar o_{<k})$). For convenience, we write the data as $\bar a\bar r\bar o_{0:n}$. We define the likelihood of a DMM+parameters as:

$$\begin{aligned} Pr(\bar r_{0:n}|\bar a\bar o_{0:n},\vec\theta,\xi) &:= \prod_{i=0}^{n} \vec\theta_{\bar s_{i-1},\bar a_i}(\bar r_i) \\ &= \prod_{s,a}\prod_{r} \vec\theta_{s,a}(r)^{m_{s,a}(r)} \qquad (7) \end{aligned}$$

where, $\delta(q_\circ,\bar a\bar o_{0:k}) = \bar s_k$ and $\bar s_{-1} = q_\circ$; the second line is a rewriting using the definitions with $m_{s,a}(r)$ as the number of times reward $\bar r_i = r$ and $\bar s_{i-1} = s$ and $\bar a_i = a$ in $\bar r_{0:n}$ and $\bar a\bar s_{0:n}$. (7) is a distribution over reward sequences that may be observed given the action-state sequence $\bar a\bar s_{0:n}$.

We now check that this likelihood is statistically consistent[2]. In particular, we need to ensure that likelihood function tests whether the $\vec\phi$ for the model are also correct, as they are not used in definition in (7). To establish consistency we need to show two things: (1) a model predicting incorrect rewards is not selected and (2) a model with incorrect state transitions is not selected in the limit of infinite data if we use this likelihood. In the following we will do so informally.

For case (1), if a model given by $\xi^A, \vec\phi^A, \vec\theta^A$ predicts rewards incorrectly, this means $\exists s \in \mathcal{S}^A$ such that $\mathcal{H}(s) \cap \mathcal{H}(\hat s)$ is infinite for some $\hat s \in \mathcal{S}^g$ and $\vec\theta^A_{s,a} \neq \vec\theta^g_{\hat s,a}$ for some a. It is now intuitively clear from line 2 of (7) that the likelihood ratio of the true model and this incorrect model will go to infinity as the length of the data sequence goes to infinity. This means the likelihood function is consistent for case (1).

For case (2) we assume $\xi^A, \vec\phi^A, \vec\theta^A$ predicts the rewards correctly at every step for each history, but the $\vec\phi^A$ are incorrect. That is, $\exists s \in \mathcal{S}^A$ such that $\mathcal{H}(s) \cap \mathcal{H}(\hat s)$ is infinite for some $\hat s \in \mathcal{S}^g$ and $\vec\phi^A_{s,a} \neq \vec\phi^g_{\hat s,a}$ for some a. For simplicity, we further assume that the parameters $\vec\theta^g$ are distinct for each s, a pair.

First, note that by the determinism of the δ functions, for any $ao_{0:n}$, ξ^A and ξ^g deterministically goes through state sequences (say) $s^A_{0:n}$ and $s^g_{0:n}$ respectively. Since ξ^A predicts the rewards correctly at each state in the sequence, and since the reward distributions $\vec\theta^g_{s,a}$ are assumed to be distinct for each pair s, a, each state in $s^A_{0:n}$ must map to exactly one state of $s^g_{0:n}$. That is if $s^A_k = s \in \mathcal{S}^A$, and $s^g_k = \hat s \in \mathcal{S}^g$, then for all other k' such that of $s^A_{k'} = s$, it must be that $s^g_{k'} = \hat s$. So if $s, s' \in \mathcal{S}^A$ corresponds to states $\hat s, \hat s' \in \mathcal{S}^g$ in the above sense, then as $n \to \infty$, the (Bayes or ML) estimate of the state transition distribution $\hat{Pr}(s'|s,a)$ obtained from the sequence $s^A_{0:n}$ must converge to the true value $\vec\phi^g_{\hat s,a}(s')$. Now during learning we will in fact compute a Bayes estimate of the state transition distribution from data (see next section), and hence by the above arguments, the fact that $\vec\phi$ is not in the likelihood function become irrelevant. This establishes consistency of the likelihood function for case (2).

Rigorous proofs of the above, without the simplifying assumption, is long, technical, unilluminating and require measure theoretic arguments (Mahmud, 2010). But the key to all the proofs is the deterministic nature of the transitions when conditioned on actions and observations. In addition, we make the following identifiability assumption about $\xi^g, \vec\theta^g, \vec\phi^g$: for any two states s and s' there must exist a history h such that $\delta(s,h) = q$ and $\delta(s',h) = q'$ and q and q' have different reward distributions for some action. In words, given any two distinct states s, s', there must exist an input history h that, starting from s, s', lead to states q, q' with different reward distributions. It is easy to see that this assumption is satisfied by many, if not all domains we are likely to encounter in practice.

3.2. Posterior and Predictive Distributions

We now derive other quantities of interest using routine methods (Chipman et al., 1998). First, we put a conjugate uninformative Dirichlet prior $w()$ on $\vec\theta_{s,a}$ and integrate to obtain $L(\bar r_{0:n}|\bar a\bar o_{0:n}, s, a, \xi)$, the marginal likelihood of each state-action pair s, a:

[2] If DMMs modeled observation distributions, then the likelihood function would be the generative probability of $\bar r\bar o_{0:n}$ given $\bar a_{0:n}$ and there would be nothing to prove.

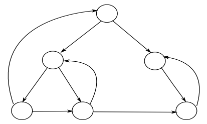

Figure 2. Example of the looped DMM. The edge labels have been removed for clarity. The root node/state is at the top and is the start state. The leaf nodes at the bottom and the parent of each node is the node above it.

$$\int \prod_r \vec{\theta}_{s,a}(r)^{m_{s,a}(r)} w(\vec{\theta}_{s,a}) d\vec{\theta}_{s,a} = \Gamma(1) \frac{\prod_{r'} \Gamma(m_{s,a}(r') + \frac{1}{|\mathcal{R}|})}{\Gamma(\sum_{r'} m_{s,a}(r') + 1)} \quad (8)$$

where Γ is the Gamma function. L is a distribution over rewards at state s for action a. The marginal likelihood of ξ is a distribution over reward sequences (with $\bar{s}_{0:k}$ as the corresponding state sequence) and is given by:

$$Pr(\bar{r}_{0:n}|\bar{a}\bar{o}_{0:n}, \xi) = \prod_{s,a} L(\bar{r}_{0:n}|\bar{a}\bar{o}_{0:n}, s, a, \xi) \quad (9)$$

Finally, we denote the space of DMMs by $\mathbf{H}$ and set it to the set of all DMMs with $< M$ states (say $M = 2^{500}$ (!)) and we put a uniform prior W over this space (so the posterior ratio is equal to the marginal likelihood ratio for DMMs). Given the model space and the prior, the posterior probability of a ξ given $\bar{a}\bar{r}\bar{o}_{0:n}$ is:

$$Pr(\xi|\bar{a}\bar{r}\bar{o}_{0:n}) := \frac{Pr(\bar{r}_{0:n}|\bar{a}\bar{o}_{0:n}, \xi)W(\xi)}{\sum_{\xi' \in \mathbf{H}} Pr(\bar{r}_{0:n}|\bar{a}\bar{o}_{0:n}, \xi')W(\xi')} \quad (10)$$

In the next section during learning the model we will use the posterior as the selection criterion. To use a learned model for planning, we need the predictive distributions for rewards and states for pairs s, a given the data, which are, respectively:

$$Pr(r|s, a, \bar{a}\bar{r}\bar{o}_{0:n}, \xi) \quad := \quad \frac{m_{s,a}(r) + \frac{1}{|\mathcal{R}|}}{1 + \sum_{r'} m_{s,a}(r')} \quad (11)$$

$$Pr(s'|s, a, \bar{a}\bar{r}\bar{o}_{0:n}, \xi) \quad := \quad \frac{\hat{m}_{s,a}(s') + \frac{1}{|\mathcal{S}|}}{1 + \sum_q \hat{m}_{s,a}(q)} \quad (12)$$

where where $\hat{m}_{s,a}(s')$ is the number of times state $\bar{s}_i = s'$ and $\bar{s}_{i-1} = s$ and $\bar{a}_i = a$ in $\bar{a}\bar{s}_{0:n}$. ((12) was obtained by performing the same trick as above on the multinomials $Pr(s'|\vec{\phi}_{s,a}, \xi)$ using the conjugate Dirichlet prior). Now we are ready describe our learning and planning algorithm.

4. Online Learning and Planning

We give a heuristic algorithm for learning the DMMs and planning with them online. The algorithm interleaves phases of acting in the environment and in the process collecting data, and improving its estimate of DMM of the target environment via learning. The algorithm is given in Alg. 1. The initializations are self explanatory and so we focus on the main for loop.

The first step (line 4), learning a better estimate, is the heart of the algorithm and discussed in detail below. The second step (line 5) is performed as follows. First, a MDP corresponding to ξ_i is constructed using the method in Sect. 2.4, but using the estimates (11) and (12) for the MDP reward function (5) and state-transition function (6) respectively. Then the optimal policy is learned offline by using Q-learning on the constructed MDP (we do not use value iteration because Q learning was seen to be quicker without any noticeable difference in performance).

Algorithm 1 Algorithm for learning and planning.

```
Require: $T$ the length of each phase of acting in the world, $N$ the number of phases.
1: Generate a random sample $\bar{a}\bar{r}\bar{o}_{0:T}$ from the environment by choosing actions at random.
2: Let $\xi_0$ be the initial estimate of the target DMM with a single state (i.e. all transitions are to $q_\circ$).
3: for $i = 1$ to $N$ do
4:    Learn $\xi_i$ from $\xi_{i-1}$ using history so far $\bar{a}\bar{o}_{0:T \cdot i}$ by setting $\xi_i$ to search $(\xi_{i-1}, \bar{a}\bar{r}\bar{o}_{0:T \cdot i})$.
5:    Estimate optimal policy $\hat{\pi}^*$ for MDP corresponding to $\xi_i$ using Q-learning.
6:    Generate a sample $\bar{a}\bar{r}\bar{o}_{T \cdot i:T \cdot (i+1)}$ from the environment by choosing actions according to $\hat{\pi}^*$.
7: end for
```

In the third step (line 6) we follow the policy as described at the end of Sect. 2.4. Additionally, whenever the MDP/DMM transitions from state s to s' on action a and observes reward r, we perform Q-learning updates on the Q-value of s at a (the Q-values learned in line 5 are used as initial values for this step). This additional Q-learning is a heuristic to ensure that we overcome incorrect parameter estimates due to mismatch between δ^i, δ^g and $\mathcal{S}^i, \mathcal{S}^g$ of ξ_i and ξ^g and still properly explore the domain. This is our approach to dealing with the infamous 'exacerbated exploration problem' that plagues general RL (Chrisman, 1992). During this Q-learning, we also take a random action with probability 0.1 to further aid in exploration.

4.1. Estimating ξ_i

We now discuss how to estimate ξ_i in the first step of the for loop (line 4). DMMs are sequential versions of DPFAs (Vidal et al., 2005) and learning the latter is NP-hard when states might have equal distributions.

Algorithm 2 Main Stochastic Search

```
function: search($\xi, \bar{a}\bar{r}\bar{o}_{0:n}$)
Require: $K$, the number of iterations.
1: Set $\xi_{cur}$ to $\xi$.
2: for $i = 1$ to $K$ do
3:     With probability 0.6 and 0.4 respectively, set $\xi_{prop}$ to extend($\xi_{cur}, \bar{a}\bar{r}\bar{o}_{0:n}$) or loop($\xi_{cur}, \bar{a}\bar{r}\bar{o}_{0:n}$).
4:     Choose $rnd \in [0, 1]$ uniformly at random.
5:     If $\frac{Pr(\xi_{prop}|\bar{a}\bar{r}\bar{o}_{0:n})}{\alpha Pr(\xi_{cur}|\bar{a}\bar{r}\bar{o}_{0:n})} > rnd$ set $\xi_{cur}$ to $\xi_{prop}$.
6: end for
7: Return $\xi_{cur}$.
```

Since this is in fact often the case in RL problems, we need to put significant bias in our search algorithm.

Algorithm 3 Function for Extending Edges

```
function: extend($\xi, \bar{a}\bar{r}\bar{o}_{0:n}$)
1: Sample a leaf node/state $s$ according to frequency in $\bar{s}_{0:n}$ ($\bar{s}_k = \delta(q_\circ, \bar{a}\bar{o}_{0:k})$).
2: Sample an outgoing edge $ao$ at $s$ according to the empirical (w.r.t. $\bar{a}\bar{r}\bar{o}_{0:n}$) next step reward at $ao$.
3: Create a new state $s_{new}$ with context $x_s ao$ where $x_s$ is the context of $s$ (the context of root is empty).
4: Set $\delta(s, ao)$ to $s_{new}$; set $\delta(s_{new}, a'o') = \arg\max\{length(x_{s'})|x_{s'}$ is a suffix of $x_{s_{new}}\}$.
5: Add $s_{new}$ to the state of $\xi$.
6: Return $\xi$.
```

We introduce bias by restricting our algorithm to a particular class of DMMs. Each DMM in this class is a tree, such that outgoing edges at each node can go only to a child or an ancestor. Only leaf nodes are allowed to transition to other nodes that are not its descendants – see Fig. 2. This is similar to a looping suffix tree (Holmes & Isbell-Jr., 2006), but our model is not a suffix tree structure – each node in the graph is in fact a state. It is straightforward to show that all DMMs have such a loopy representation although with possibly exponentially many states. For the typical POMDP domains this representation seems particularly suitable, as dynamics in such domains have a neighborhood structure, where we are only able to transition back to states that were visited recently. Now our search algorithm (**search**, Alg. 2) is a stochastic search over this space with the posterior (10) as the selection criterion. **search** at each step performs one of two possible operations. It chooses a leaf node, and either it extends it by adding another leaf node (**extend**, Alg. 3); or it chooses an internal or a leaf node and loops one of its un-extended transitions back to an ancestor (**loop**, Alg. 4). The modification is then accepted as the estimate of ξ^g in the next iteration if it satisfies a Metropolis-Hastings (MH) style condition (line 5). The condition contains a parameter α which was set to 2^5 in all our experiments (this simulates the proposal distribution ratio in the MH acceptance probability).

Algorithm 4 Function for Looping Edges.

```
function: search($\xi, \bar{a}\bar{r}\bar{o}_{0:n}$)
1: Sample a leaf node/state $s$ according to frequency in $\bar{s}_{0:n}$ ($\bar{s}_k = \delta(q_\circ, \bar{a}\bar{o}_{0:k})$).
2: Sample an outgoing, unextended edge $ao$ at $s$ according to the empirical (w.r.t. $\bar{a}\bar{r}\bar{o}_{0:n}$) next step reward at $ao$.
3: Choose a ancestor $s'$ of $s$ according $2^{-m}/Z$ where $m$ is the no. of ancestors of $s$ between $s$ and $s'$.
4: Set $\delta(s, ao)$ to $s'$.
5: Return $\xi$.
```

We now discuss the heuristics used in **extend** and **loop** to choose nodes to modify. In line 1 of both procedure, we sample states according to its frequency in the data because we expect the impact to the likelihood to be the greatest if we modify high frequency states. Then in line 2 of both we sample an outgoing edge ao of the state s from line 1 according to the average reward seen at s after taking action a and observing o. The intuition is that if the reward at that edge is high, then it might be a good idea to create a new state there (in **extend**) or move it to a different state (in **loop**) to get a better model. A more sophisticated version of this heuristic would be to use the future discounted reward at the edge. In lines 3 and 4 **extend** constructs the new state for the chosen edge. In line 3 **loop** chooses an ancestor to loop back to, choosing a closer ancestor with higher probability. This is because in typical POMDP domains we expect the state to transition back to states visited very recently. **loop** then loops the edge in line 4.

5. Experiments

5.1. Setup

We ran experiments to illustrate that we do in fact extend McCallum's Utile Suffix Memory in desirable ways. We first ran three experiments on three POMDP maze problems to evaluate our model on standard problems. Then we ran experiments on three novel, non-episodic domains to test the long term memory ability of our method. The domains for the first are given in Fig. 3 (numbered 1,2, and 3 respectively) and they have the same dynamics as in (Mccallum, 1995)

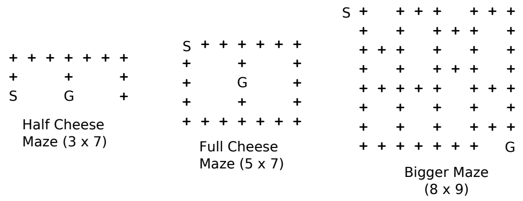

Figure 3. The three POMDP mazes. S is the start state and G is the goal state.

with state transition and observation noise.

The domains for the second type of experiments are a set of 'harvesting' tasks with m crops. A crop i become ready to be harvested after being developed through k_i different phases. It is ready to be harvested for just 1 time step once development is complete and then it spoils immediately, requiring to be grown again. The agent has $m + 1$ different actions. Action a_i develops crop i to its next phase with probability 0.7 and develops some other crop with probability 0.3. The remaining action allows the agent to harvest any crop ready to be harvested at that step. The observation at each step is the crop that has been developed at that step. So essentially this is a counting task where the agent has to remember which phase each crop is at – so a finite history suffix is not sufficient to make a decision. The agent receives -1.0 reward for each action, and 5.0 reward for a successful harvest. There are no episodes and the each problem instance essentially runs forever.

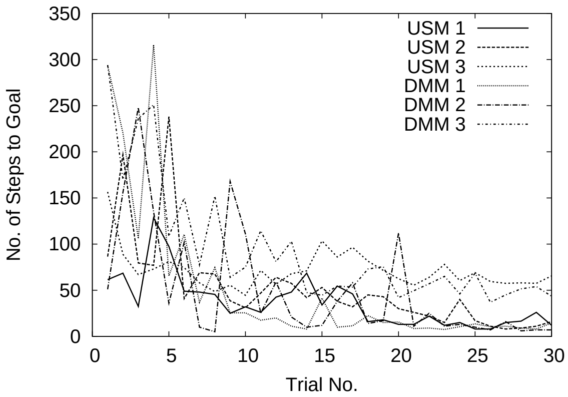

Figure 4. Results for the maze domains; USM i is the number of steps to goal state for maze i at the given trial. Similarly for DMM i etc.

We used several different values of $[m, (k_1, k_2, \cdots, k_m)]$: $[4, (2, 2, 2, 2)]$, $[5, (2, 2, 2, 2, 2)]$ and $[7, (2, 2, 2, 2, 2, 2, 2)]$ (referred to as harvesting problems $1, 2$ and 3 respectively from now on). The POMDP representation of these problems have, respectively, $4 \cdot 2^4 = 64$, $5 \cdot 2^5 = 160$, $7 \cdot 2^7 = 896$ states. In each of these cases, we let each USM and DMM run for 3000 steps and report the total accumulated reward for each method averaged over 10 trials.

5.2. Results

The results for the three maze domains are given in Fig. 4 we report the median time per trial to get to the goal state to be comparable to (Mccallum, 1995) – the performance of our implementation of USM matches that of (Mccallum, 1995). In these experiments in Alg. 1, T is variable, equal to the number of steps the agent reaches the goal, while $N := 30$; and $K := 5000$.

As can be seen, our algorithm more or less matches USM, except at the beginning where the number of steps is much higher. This is because our model is trying to learn a recursive model and hence gets confused when there is little data. Unsurprisingly, the average number of states estimated by our method was 45.3, 67.5 and 90.5 compared to 204.5, 231.5 and 623.5 for USM. Similarly, our stochastic search took significantly longer off-line processing to learn (5 minutes compared to 10s of seconds for USM). So for these types of episodic tasks, it might be better to first learn a model using USM and then compress it to a DMM (Shalizi & Klinkner, 2004). Regardless, the experiments give evidence that DMMs are viable for modeling hidden RL environment structure , and that our inference criterion (7) is correct.

The results for the 3 harvesting/counter domains are in Fig. 5. Here the difference between the two methods become quite significant, with our method dominating USM completely. The differences in computational time was quite similar to that in the previous set of experiments. The average number of states inferred differed by about 400, with the USM continually creating new states.

6. Conclusion

In this paper we introduced a novel model for learning hidden environment structure in RL problems. We showed how to use the models to approximate the environment and compute optimal policies. We further introduced a novel inference criterion, a heuristic algorithm to learn and plan with these models. We then performed experiments to show the potential of this approach. The avenues for future research are many. The most important at this time seems to be devel-

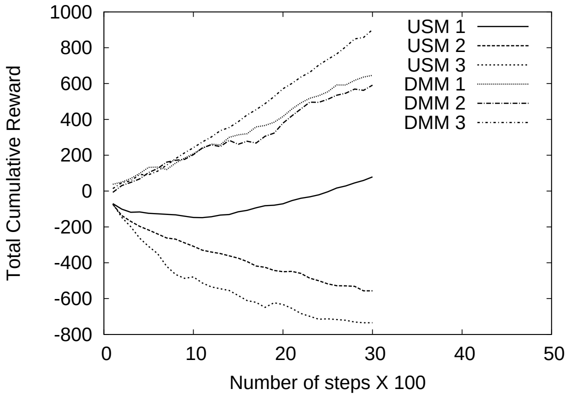

Figure 5. Results for the mining domains. USM i is the total reward accumulated by USM on mining problem i at the given step. Similarly for DMM i etc.

oping a principled algorithm for learning these models that will apply across many different tasks. Another important generalization would be to consider extending DMMs to more structured, possibly relational, state spaces. It would also be interesting to extend the notion of DMMs to consider arbitrary measurable state spaces and see how the inference criterion works in that situation.

Acknowledgments

We would like to thank John Lloyd and Marcus Hutter for many useful comments and fruitful discussions. The research was supported by the Australian Research Council Discovery Project DP0877635 "Foundation and Architectures for Agent Systems".

References

Bertsekas, D. P. and Shreve, S. E. *Stochastic Optimal Control: The Discrete-Time Case.* Athena Scientific, paperback edition, 1996.

Chipman, H. A., George, E. I., and McCulloh, R. E. Bayesian CART model search. *Journal of the American Statistical Association*, 93:935 – 948, 1998.

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

Doshi-Velez, F. The infinite partially observable markov decision process. In *Proc. of the* 20th *Neural Information Processing Systems Conference*, 2009.

Holmes, M. P. and Isbell-Jr., C. L. Looping suffix tree-based inference of partially observable hidden state. In *Proceedings of the* 23^r*d International Conference on Machine Learning*, 2006.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. *Introduction Automata Theory, Languages and Computation.* Addison Wesley, 3rd edition, 2006.

Hutter, M. Feature markov decision process. In *Proceedings of the* 2nd *AGI Conference*, 2009.

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

Littman, M., Sutton, R., and Singh, S. S. Predictive representations of state. In *Proceedings of the* 15^t*h Neural Information Processing Systems Conference*, 2002.

Madani, O., Hanks, S., and Condon, A. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.

Mahmud, M. M. H. A deterministic probabilistic approach to modeling general rl environments. Technical report, Australian National University, 2010.

Mccallum, A. Instance-based utile distinctions for reinforcement learning. In *Proceedings of the* 12th *International Machine Learning Conference*, 1995.

Ron, D., Singer, Y., and Tishby, N. Learning probabilistic automata with variable memory length. In *Proceedings of the* 7th *Conference on Computational Learning Theory*, 1994.

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

Sabbadin, R., Lang, J., and Ravaonjanahary, N. Purely epistemic markov decision process. In *Proceedings of the* 22nd *National Conference on Artificial Intelligence*, 2007.

Shalizi, C. and Klinkner, K. Blind construction of optimal nonlinear recursive predictors for discrete sequences. In *Proc. of the* 20th *Conference on Uncertainty in Artificial Intelligence*, 2004.

Vidal, E., Thollard, F., Higuera, C., Casacuberta, F., and Carrasco, R. C. Probabilistic finite-state machines part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, 2005.