Multi-Task Reinforcement Learning: A Hierarchical Bayesian Approach

Aaron Wilson WILSONAA@EECS.OREGONSTATE.EDU
Alan Fern AFERN@EECS.OREGONSTATE.EDU
Soumya Ray SRAY@EECS.OREGONSTATE.EDU
Prasad Tadepalli TADEPALL@EECS.OREGONSTATE.EDU
School of Electrical Engineering and Computer Sciences, Oregon State University, USA

Abstract

We consider the problem of multi-task reinforcement learning, where the agent needs to solve a sequence of Markov Decision Processes (MDPs) chosen randomly from a fixed but unknown distribution. We model the distribution over MDPs using a hierarchical Bayesian infinite mixture model. For each novel MDP, we use the previously learned distribution as an informed prior for model-based Bayesian reinforcement learning. The hierarchical Bayesian framework provides a strong prior that allows us to rapidly infer the characteristics of new environments based on previous environments, while the use of a nonparametric model allows us to quickly adapt to environments we have not encountered before. In addition, the use of infinite mixtures allows for the model to automatically learn the number of underlying MDP components. We evaluate our approach and show that it leads to significant speedups in convergence to an optimal policy after observing only a small number of tasks.

1. Introduction

Reinforcement Learning (RL) based on the framework of Markov Decision Processes (MDPs) is an attractive paradigm for learning by interacting with a stochastic environment and receiving rewards and penalties. In this paper, we consider Multi-Task Reinforcement Learning (MTRL), where an agent is confronted with a sequence of MDPs chosen independently from a fixed distribution. The agent's goal is to quickly find an optimal policy for each MDP. It is, of course, possible to solve each MDP from scratch. Indeed, there is no better way if the MDPs do not share any structure. In our work, we are interested in cases where different MDPs share aspects of the model, so that learning about one MDP can help to solve the other MDPs.

As a motivating example, consider a prospecting agent who is interested in finding gold. Depending on the type of the environment, the indicators for gold might differ. Perhaps certain kinds of rocks might be good indicators of the presence of gold in some environments. In some other environments, gold may be more abundant along the river beds. Moreover, the agent does not know a priori what the indicators of gold are, and what type of environment it currently is in. We would like the agent to learn the distribution of different types of environments and the indicators of gold in each of them after exploring a few such environments. After this, given a small amount of experience in a new environment, the agent could quickly make inferences about the environment type, and use that information to explore more efficiently. For example, if an agent finds that there is no gold near the typical indicator rocks, it might conclude that the river beds might be better places to explore next.

A naïve approach to multi-task reinforcement learning is for the agent to treat all of its observations as coming from a single MDP. In this case, the agent would estimate the most likely set of MDP parameters that generate the given observations. If in fact the MDPs generating the observations are not similar to each other, this might hurt exploration in a new MDP, because the learned model will generalize poorly. It may even result in slower learning than if the agent assumes nothing about the model. Our goal is to take advantage of the similar structure between MDPs when it exists, while at the same time avoid-

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007.

ing or limiting knowledge transfer between dissimilar MDPs. In our work, we do this by using a hierarchical infinite mixture model with a potentially unknown and growing set of mixture components. Each component captures uncertainty in both the MDP structure and parameter values and stands for a class of "similar MDPs." After exploring a set of MDPs, the agent learns the distribution of different classes and their model characteristics. We evaluate our approach on two gridworld domains and demonstrate that (i) our approach can use the learned hierarchical model to explore more efficiently in a new environment than an agent with no prior knowledge, (ii) it can successfully learn the number of underlying MDP classes, and (iii) it can quickly adapt to the case when the new MDP does not belong to a class it has seen before.

2. Multi-Task Reinforcement Learning

We formulate Multi-Task Reinforcement Learning in the framework of Markov Decision Processes (MDPs).

An MDP M is defined by a 5-tuple (S, A, C, T, I, F), where S is a set of states and A is a set of actions. $C(s, a)$ is the immediate cost of executing action a in state s, and the transition function $T(s, a, s')$ is the probability of reaching state s' given that action a is taken in state s. I is a distribution of initial states and F is a set of final or terminating states. A policy π for M is a (possibly stochastic) mapping from S to A. The expected cost of a policy π starting from a state s is the sum of the costs of the actions starting from state s until the policy reaches a final or terminating state. We are interested in finding an optimal policy that minimizes the total expected cost from all states. The minimum expected cost from state s is the unique solution to the following Bellman equations:

$$V^*(s) = \begin{cases} 0 \text{ if } s \text{ is a terminal state} \\ \min_a C(s,a) + \sum_{s'}(T(s,a,s')V^*(s')), \text{ else} \end{cases}$$

The actions that minimize the right hand side of the above equation constitute the optimal policy. In RL, the goal is to compute such a policy by acting in the environment and observing the state transitions and rewards. In model-based reinforcement learning, we first learn an approximate model of the MDP and use algorithms such as value iteration (Sutton & Barto, 1998) to compute an optimal value function V^* with respect to the learned model.

In this work, we focus on the problem of **multi-task reinforcement learning** (MTRL). An MTRL problem is defined by a distribution D over a potentially infinite set of MDPs or tasks (we use "MDP" and "task" synonymously in this work). Given an MTRL problem a multi-task learner L is provided with a potentially infinite random sequence of MDPs $M_1, M_2, \ldots$ drawn i.i.d. from D and is allowed to act in each MDP until the termination state is reached.

Intuitively our objective is to develop a multi-task learning algorithm that is able to leverage experience in previous MDPs $M_1, \ldots M_n$ to more quickly learn an optimal policy in a newly drawn MDP M_{n+1} compared to a algorithm that ignores the previous MDPs. Clearly if the MDPs drawn from D do not share common structure, then on a newly drawn MDP the multi-task learner will have no leverage over a single-task learner that is applied to each MDP in isolation. At the other extreme, if D assigns probability one to a single MDP, then the multi-task learner will be able to perform much better than a single-task learner after experiencing the first MDP. The most interesting MTRL problems are where D lies somewhere in between these two extremes.

In our work, we consider a particular class of MTRL problems where the MDPs share some components of the model. In particular, we assume a generative process for the models of MDPs drawn from D. To define the process we first associate each state s in each MDP with a descriptive feature vector $f(s)$. For example, in the motivating prospector problem, the features of a state might indicate the types of rocks and other geographic features in a window around the current location of the agent. The generative process first draws a class c from some unknown distribution over an unknown number of possible classes. Conditioned on c, functions over $f(s)$ are then drawn that define the reward and/or transition dynamics of an MDP. This generative model allows for there to be different but shared classes of model structures in the generated MDPs. For example, in the prospecting example, one class of MDP may have a high probability of locating rewards next to a river bank, and another class might be more likely to assign rewards next to certain rocks. The class of an MDP is unobservable to the learner and the number of classes is unknown. It is up to the multi-task learner to hypothesize the possible classes of MDPs based on experience and to use this to infer the likely reward and/or transition structure of a new MDP after some experience in it.

3. Related Work

Recently, knowledge transfer in the MTRL setting has attracted a great deal of interest, and several methods have been proposed that enable RL agents to take advantage of solutions and observations from prior tasks. Some of these methods assume that successive tasks share similar dynamics and reward structure, for example methods that use approximations of prior optimal value functions as initial value functions for new tasks (Konidaris & Barto, 2006). Other approaches attempt to automatically construct general features that enable mapping substantially different tasks together, and transferring value functions defined in terms of these features (Banerjee & Stone, 2007). Yet other approaches (Mehta et al., 2005) assume that the reward function of the new task is given, and maintain value functions for several different prior tasks and initialize the value function of a new task with one that yields the most reward. The current work differs from these previous approaches in that we adopt a more principled hierarchical Bayesian framework to formalize the similarities between the different MDPs. By maintaining probabilistic models at multiple levels of the MDP class hierarchy, we can naturally take advantage of the common structure between different RL tasks, while also learning specialized knowledge in a single task.

Hierarchical Bayesian RL is also related to Bayesian Reinforcement Learning (Dearden et al., 1998a; Dearden et al., 1998b; Strens, 2000; Duff, 2003), where the goal is to give a principled solution to the problem of exploration by explicitly modeling the uncertainty in the rewards, state-transition models, and value functions. Here, a distribution is maintained over possible true MDPs rather than a single maximum likelihood estimate. This distribution is used to select actions of high expected utility. However, in standard Bayesian RL, it is difficult to choose an informed prior distribution over MDPs. In the MTRL setting, however, a Bayesian approach facilitates knowledge transfer across MDPs by providing a much more informed starting point. Thus, we believe the multi-task setting and our hierarchical approach better highlights the potential utility of Bayesian RL.

4. Hierarchical Bayesian MTRL

In this section, we outline our hierarchical Bayesian approach to multi-task reinforcement learning.

4.1. Overview

Our approach to multi-task reinforcement learning can be viewed as extending Bayesian RL to a multi-task setting. In (single-task) Bayesian model-based RL, a posterior probability distribution $P(M|\Theta, O)$ is maintained over possible MDPs, where M denotes a random variable over MDPs, O is the current set of observations from the underlying MDP, and Θ denotes the parameters of the probability model (Figure 1(a)). This distribution is used to select actions and is refined

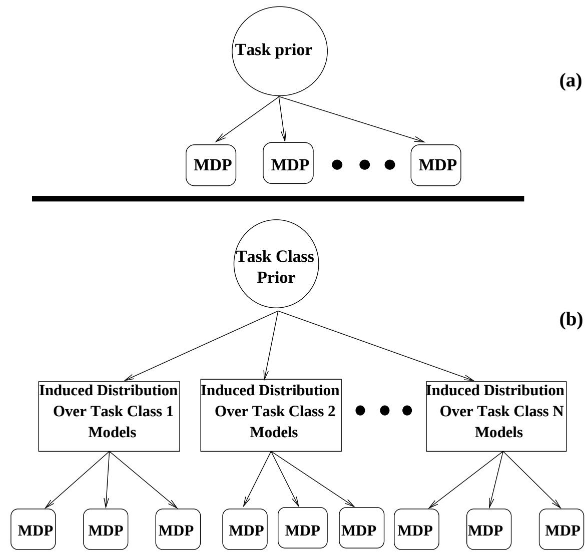

Figure 1. (a) Single task (i.e., MDP) Bayesian RL vs. (b) Hierarchical Bayesian MTRL. The number N and parameters of the induced distributions are learned from data.

as the agent acts and observes the environment.

Our MTRL approach follows this Bayesian framework, adding to it the key idea of using hierarchical Bayes to model "classes" of MDPs. Figure 1(b) depicts the general form of our generative model. We introduce a (hidden) random variable C over "possible MDP classes" where each class $C = c$ in turn induces a distribution over the probability parameters Θ above. Sampling from this induced distribution results in an MDP $M = m$ with probability $\Pr(M = m|\Theta = \theta, C = c)$. This addition of a "class layer" makes the probability model hierarchical. Intuitively, the hierarchical Bayesian approach allows us to explicitly reason about the shared structure of certain MDPs and to transfer knowledge to a new MDP from those in the same class.

To achieve the above behavior it is critical that our system be able to automatically learn the class-level model and to allow for class inference of newly observed MDPs. Therefore, we utilize a nonparametric infinite mixture model at the class layer. This allows us to (a) model a potentially unbounded number of classes, (b) adapt to and make inferences about classes we have not seen before, and (c) learn the shared structure that constitutes each class on the fly, rather than providing a rigid specification for each class (of course, the shared structure needs to conform to the assumptions we make about the probabilistic model).

The key steps in our approach are outlined in Algorithm 1. Our algorithm uses a hierarchical Bayesian model to estimate the MTRL distribution over MDPs, which is used as a strong prior for Bayesian RL in

Algorithm 1 Hierarchical Bayesian MTRL Algorithm

1: Initialize the hierarchical model parameters Ψ
2: **for** each MDP M_i from $i = 1, 2, \ldots$ **do**
3: $\quad O_i = \emptyset$ //O_i is the set of observations for M_i
4: $\quad$ **while** policy π has not converged **do**
5: $\quad\quad \hat{M}_i \leftarrow \texttt{SampleMAP}(\Pr(M \mid O_i, \Psi))$ // Section 4.4
6: $\quad\quad \pi = \texttt{Solve}(\hat{M}_i)$ //e.g. by value iteration
7: $\quad\quad$ `Run` π in M_i for k steps
8: $\quad\quad O_i = O_i \cup \{$observations from k steps$\}$
9: $\quad$ **end while**
10: $\quad \Psi \leftarrow \texttt{SampleMAP}(\Psi | \hat{M}_1, \ldots, \hat{M}_{i-1})$ // Section 4.3
11: **end for**

a new MDP. Initially before any MDPs are experienced, the hierarchical model parameters Ψ are initialized to uninformed values. The procedure `SampleMAP` (line 5) generates a set of MDPs sampled according to $\Pr(M|\Psi)$ and then returns the one, $\hat{M}_i$, that has the highest probability. $\hat{M}_i$ is then solved (for example, using value iteration) for the optimal policy π (line 6) and then π is followed for k steps in the environment (line 7). This is similar to Thompson sampling which has been employed with success in prior work (Thompson, 1933; Strens, 2000; Wang et al., 2005), but differs in that instead of sampling a single MDP and solving it, we draw a set of MDPs and select the one with highest probability. We have observed that this approach tends to give better performance compared to pure Thompson sampling.

The observations gathered during the k steps of π are then stored in the observation database O_i for M_i. The agent then uses the newly updated O_i to update the posterior distribution $\Pr(M|\Psi, O_i)$, and computes a new $\hat{M}_i$. This loop is repeated for MDP $\hat{M}_i$ until the policy has converged. After convergence, we update the hierarchical model parameters Ψ based on the parameter estimates $\hat{M}_i$. This update is also done using the `SampleMAP` procedure, where the sample is generated using the Gibbs sampling procedure described in Section 4.3. The resulting estimate for Ψ specifies the (approximately) most likely class assignment to MDPs along with the parameters associated with each class distribution. Note that the `SampleMAP` procedure will automatically determine the number of distinct classes to utilize. Intuitively, this step will result in a Ψ that captures the inherent class structure of the sequence of MDPs before and including M_i.

In what follows, we give details of our approach. First, we describe our hierarchical probability model. Next, we describe the computation of the model parameters Ψ given a sequence of previously experienced MDPs, followed by the sampling procedure for MDPs given the model parameters Ψ and observations.

4.2. Hierarchical Bayesian Model

We propose a generative model corresponding to the class structure illustrated in Figure 1(b). Each class $C = c$ is associated with a vector of parameters θ_c. These parameter vectors define distributions from which individual MDPs will be drawn. It is important to note that our approach is not tied to a specific form for these distributions, or specific form of a (shared) prior distribution G_0 over all θ_c's. Rather, our approach is generic in that we can utilize any form of MDP distribution as long as we are able to sample from the posteriors of these distributions. As an example, assume that $\theta_c = (\mu_c, \sigma_c)$, defining a Gaussian distribution. Then the generative process draws a class assignment using the Dirichlet Process described below, uses the assignment as an index to the corresponding Gaussian distribution, and samples an MDP from that distribution. When doing inference about an MDP, assume that the prior G_0 is uniform over all μ and σ. Then, given a sequence of MDP observations, we use the Dirichlet Process to hypothesize a class assignment, update the corresponding θ_c (incorporating G_0 into the update), and iterate this procedure till convergence.

To understand the role of the Dirichlet Process, first consider the case when there are a finite, known number of MDP classes. Suppose we observe a set of N MDPs from a set of k classes. In this case, it is natural to use a multinomial model over class assignments, with parameter vector β. The probability of the observed data will be proportional to $\prod_{j=1}^{k} \beta_j^{c_j}$ where c_j is the number of MDPs observed to be in class j and $\sum_j c_j = N$. If we wish to infer β, we can use an informative Dirichlet prior. The Dirichlet prior indicates our prior belief about the prevalence of each class in the data—if we expect that most MDPs will be drawn from class c, the Dirichlet parameter (count) corresponding to c should be large. This will mean that our initial MAP estimates of β_c will be larger than for other classes.

Now consider the case when we do not know how many classes there are. In this case, we cannot explicitly estimate or use the multinomial distribution, since the number of parameters $|\beta|$ is unknown. However, using Dirichlet Process priors, it is possible to estimate the *conditional distribution* of the class of a new observation given the class assignment for previous observations, using the equations:

$$\begin{aligned} \Pr(C_i = c | c_1, \ldots, c_{i-1}) &\propto \frac{n_{-i,c}}{i - 1 + \alpha} \\ \Pr(C_i \neq c_j \forall j < i | c_1, ..., c_{i-1}) &\propto \frac{\alpha}{i - 1 + \alpha} \end{aligned} \quad (1)$$

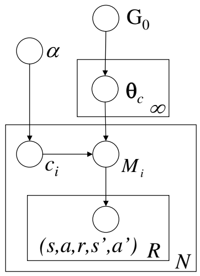

Figure 2. Infinite Mixture Model (Parameter set Ψ). There are N MDPs. For each MDP the agent has made R observations. Parameters θ_c capture the class distributions (level 2 in Figure 1) and the parameter set G_0 is a hyperprior over classes (the root in Figure 1). The class assignments c_i assign each MDP to a class. The parameter α influences the probability with which new classes are considered.

Here, C_i a random variable over possible class assignments to the i^{th} (last) datapoint, $c_1, \ldots, c_{i-1}$ are class assignments to the previous $i-1$ observations, $n_{-i,c}$ is the number of data points, excluding point i, currently assigned to class c, and α is a parameter of the Dirichlet Process, referred to as the "concentration parameter." The parameter α governs the probability with which the Dirichlet Process hypothesizes that the i^{th} observation comes from a new class (sometimes called an "auxiliary class"). Using Equation 1, we can define a Gibbs sampling procedure that repeatedly samples class assignments until convergence. Observe that this procedure implicitly defines a multinomial distribution, but now the parameter vector for this distribution can vary in size as we observe more data. Thus, the Dirichlet Process prior can be used to specify a distribution over an unbounded, potentially infinite set of classes from which our observations are drawn.

In Figure 2, we show our model in plate notation. Rectangles indicate probability distributions that are replicated a certain number of times, shown in the bottom-right corner. The set of learned model parameters Ψ consists of the class parameters θ_c, the number of which is unbounded and the class assignments c_i. The figure shows a distribution over N MDPs in each of which the agent has made R observations.

4.3. Updating the Hierarchical Model

Here we describe how we update the hierarchical model parameters Ψ (line 10 in algorithm 1) given a set of parameter estimates $\hat{M}_1, \ldots, \hat{M}_{i-1}$ and a new parameter estimate for the i^{th} MDP, $\hat{M}_i$. We use a Gibbs sampling routine (Neal, 2000) described in Algorithm 2 to sample the posterior distribution $\Pr(\Psi|\hat{M}_1, \ldots, \hat{M}_{i-1})$ and select the most probable parameters.

Algorithm 2 Auxiliary Class Gibbs Sampler

Let $m =$ Number of auxiliary classes.
Let F be the MDP distribution given a class: $M \sim F(\theta_c)$
Let M_j be the final estimates for the MDP j's parameters, for all j
Initialize the Markov chain state $(\Theta = \Theta_0, C = C_0)$
repeat
  Let $K = |\Theta|$
  for $c = K+1 : K+m$ **do**
    Draw θ_c from G_0
  end for
  Let $\hat{\Psi} = \{C, \Theta, G_0, \alpha\}$
  for j=1:i **do**
    $c_j =$ `SampleAssignment`$(\hat{\Psi}, M_j, F, m)$
  end for
  Remove all classes with zero MDPs.
  $\Theta \leftarrow \text{Sample}(\Pr(\Theta'|c_1, \ldots, c_i))$
until convergence

Algorithm 2 is an auxiliary class Markov Chain Monte Carlo sampling technique for Dirichlet process models where the hyperprior distribution G_0 is not conjugate to the component distribution. In our case the state of the Markov chain includes the set of class assignment variables $C_0 = \{c_1, c_2, \ldots, c_i\}$ and the class parameters $\Theta_0 = \{\theta_1, \theta_2, \ldots, \theta_k\}$ (if we have seen k classes so far). Auxiliary classes are introduced to allow the set of components to grow as the data requires. We found setting the number of auxiliary classes m to a small value gave good performance. Such a Gibbs sampling procedure requires conditional distributions for the discrete class assignment variables, which are given in Equation 1, and a procedure for sampling the posterior distribution for the class parameters θ_c given the class assignments. It takes as input the current parameter estimates $\hat{M}_1, \ldots, \hat{M}_{i-1}$ and model parameters (C_0, Θ_0) and outputs updated model parameters (C, Θ). The update is made by running the sampling routine until the chain has burned in, and then generating a large sample and selecting the most probable parameters (Line 10 in Algorithm 1).

The algorithm works as follows. The Markov chain is initialized with the current parameters. Parameters for the m auxiliary classes are drawn from the hyperprior G_0. Then, the `SampleAssignment` routine (Algorithm 3) is used to propose a new class assignment to the i MDPs. Given an assignment, a new set of class parameters are computed. We do not give details on this step because it depends on the form of the MDP distribution, and we do not assume any specific distribution in this algorithm. As long as there is a procedure to sample from the posterior distribution over Θ, we can define some suitable update for the class parameters. The class assignment and parameter updates are repeated until we have a sufficiently large sample from the posterior. The assignment, $\Psi = (C, \Theta)$, that maximizes the posterior probability is then returned.

4.4. Sampling an MDP

In this section, we describe the procedure for sampling an MDP from the current hierarchical model. To do this, at each step, the agent must have a good idea about which class the current MDP is drawn. A computationally expensive method of maintaining the accuracy of the current hypothesis is to update the parameter estimate, using Algorithm 2 above, after each observation. However, this is unnecessary in practice. We need only sample the full posterior once between environments as this is sufficient to maintain a good estimate. Instead, the output from Algorithm 2, the updated estimate Ψ, including the class assignments C, and class parameters Θ are assumed to be an informed prior and remain fixed during exploration of the subsequent MDP. To take advantage of this information the agent need only determine which current class, if any, M_i belongs. If M_i belongs to a known class, ($c_i \in 1..k$), then the information in θ_{c_i} is useful for exploration. Otherwise, the agent performs no worse than if it used the prior G_0.

To do this properly we need to maintain a sample $\hat{M}_i$ for the MDP model parameters, and a current hypothesis for the class assignment. We initialize $\hat{M}_i$ by sampling from the informed prior, and initialize C_i similarly. These two quantities will constitute the state used for our sampling procedure. After each observation, given Ψ, the algorithm samples a sequence of assignments using Algorithm 3 for $C_i = c$. In this case we also allow a number of auxiliary components, since the current MDP may belong to a novel class. Given the sample the algorithm selects the most probable class assignment c, and then samples an MDP from class c using the posterior distribution $P(M_i|\theta_c, O_i)$. The sampled MDP parameters are used to update $\hat{M}_i$ (Line 5 of Algorithm 1). Note that the specific sampling procedure may depend on the exact form of $F(\theta_c, M_i)$. Given $\hat{M}_i$ the algorithm solves for the optimal policy and acts for a fixed number of steps (Lines 6 and 7 of Algorithm 1). After the final observation the inner loop of Algorithm 1 returns the estimate $\hat{M}_i$.

Algorithm 3 `SampleAssignment`$(\hat{\Psi}, M_j, F, m)$

1: Let i be the total number of MDPs seen so far
2: Let $i_{-j,c}$ be the number of MDPs assigned to class c after class j is removed for all j, c
3: Let $F_{c,j}$ denote $F(\theta_c, M_j)$, the probability of M_j in the class c for all c, j
4: Sample C_j according to:

$$\Pr(C_j = c) \propto \begin{cases} \frac{i_{-j,c}}{i-1+\alpha} F_{c,j}, & 1 \le c \le K \\ \frac{\alpha/m}{i-1+\alpha} F_{c,j}, & K+1 \le c \le K+m \end{cases}$$

5. Empirical Evaluation

In this section, we evaluate our approach on two domains. We hypothesize that, after seeing a few training environments, our hierarchical Bayesian MTRL algorithm will (i) successfully learn the underlying number of MDP classes and (ii) use the learned priors to explore efficiently in a new environment, thus converging more rapidly to the optimal policy than an algorithm which solves the new algorithm from scratch.

To evaluate our hypotheses we consider a colored maze domain. Each square of the domain is randomly colored with one of n colors and the agent may navigate in any of four directions to adjacent squares. In the first instance of this domain the agent is tasked with navigating from the top left corner to the bottom right corner by following the least cost path to that goal. The objective is to maximize the total reward per episode. Episodes end only after the goal is reached. In this domain, we assume that the transition function is the usual deterministic transition function. The unknown MDP parameters are the parameters of the reward function. Thus, each task is characterized by the parameters of the reward function $\mathbf{w}$. We define the reward to be a linear Gaussian function of the colors surrounding the agent. Define a "color vector" $\mathbf{Q} = \{q_{d,j}\}$ where each $q_{d,j}$ is a boolean variable which is true if the d^{th} square relative to the agent's current location (d can be current, up, down, left or right) is colored by j^{th} color. Then, when the agent enters a state, it receives a reward drawn from a Gaussian distribution with $\mu = \mathbf{w} \cdot \mathbf{Q}$ and known diagonal covariance matrix. To define the true class distributions that govern the class parameters θ_c, we use a set of fixed Gaussian distributions. The hyperprior G_0 over θ_c (used in inference) is set to be a Normal Inverse Wishart distribution. The true distribution over class assignments is a fixed multinomial distribution. Thus, to draw individual MDPs we first select a class by sampling from the multinomial distribution over the (true) set of classes, and then sample a task, which is a weight vector $\mathbf{w}$, using the Gaussian distribution associated with that class.

Our experiments are set up as follows. Using the true

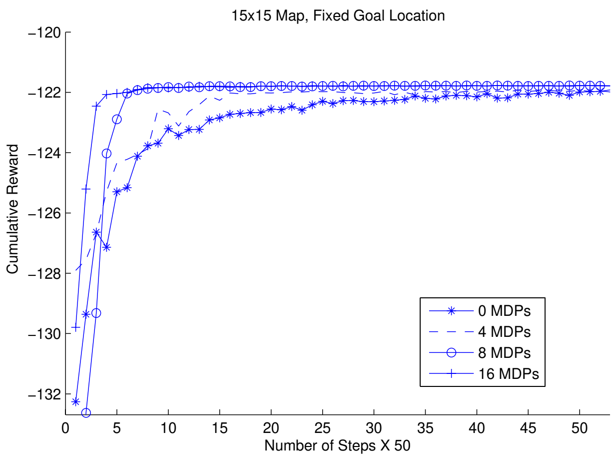

Figure 3. Total reward every 10 steps, averaged on the test set, for gridworld problems with unknown reward parameters.

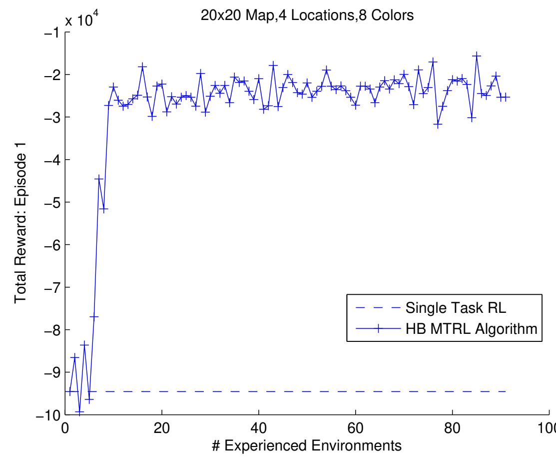

Figure 4. Cumulative reward on the first episode plotted against the number of training MDPs on 20×20 map, for gridworld problems with unknown goal states.

distribution, we draw a set of training MDPs and draw a fixed test set of 50 MDPs. To evaluate the agent's performance, we report the total reward per episode, averaged over the test set, plotted against the number of training environments seen by the agent. Figure 3 shows our results with 8 different colors and 4 underlying MDP classes on maps of size 15-by-15. From these results we observe that in this domain, even the algorithm that learns each new task from scratch converges quite rapidly. This is because in this domain, essentially every step the agent takes gives some information about the parameters of the reward function. Despite this, we observe that our MTRL algorithm derives considerable benefit from the training MDPs. After having experienced 16 training MDPs the agent finds an optimal solution in approximately 100 steps. In contrast, more than 2500 steps are required for the uninformed algorithm to reach the same level of performance. This also indicates that our algorithm's estimate of the posterior closely approximates the true distribution over tasks after a very few examples. Further, we also observed in these experiments that, although the true distributions over classes overlapped, our inference procedure was able to find the true number of underlying classes.

In the second set of experiments, we consider a modification of our domain that makes learning more difficult without a strong prior on model parameters. In particular, we let the location of the goal vary between environments. In this task, the agent is returned to the starting state and the episode ends after finding the goal location which may be any square on the map. The objective is to maximize the total reward per episode. The map construction algorithm and reward function remain unchanged. The true distribution over the goal locations is defined by a Gaussian distribution centered around a different fixed location for each class. Thus the probability that, for any new map, a given location will be chosen as the goal is proportional to the Gaussian probability with the mean being the goal location for each class. The priors over class reward parameters are specified as before. To sample the posterior distribution for goal locations, in line 5 in Algorithm 1, we use a rejection sampling routine. This routine discards sampled goal locations corresponding to previously visited areas.

In Figures 4 and 5, we show the the total reward received during the first episode for our algorithm and the baseline, which learns each environment from scratch (this measure is sometimes called the "jump-start"). On the x-axis is shown the number of previous training environments for our algorithm (the baseline algorithm's performance is constant over the x-axis because it solves every task from scratch). We chose this graph because in this domain, we observed that the cost to find the goal dominates the exploration. By the time the goal is found, the agent usually also has a good estimate of the reward weights and converges rapidly to the optimal policy. Since by our specification an episode ends only after the goal is found, it is reasonable to measure improvement by considering the agent's behavior in just the first episode.

From the figures, we observe that in this domain, learning a strong prior on the goal locations (and the rewards) using the previous MDPs has a large effect on exploration in a new map. We observed that the algorithm quickly discovers the set of classes, and then focuses exploration on areas around the class goals. This intelligent search process is much more efficient

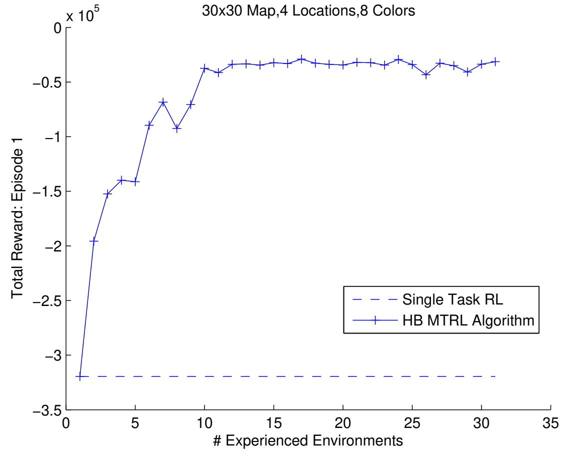

Figure 5. Cumulative reward on the first episode plotted against the number of training MDPs on 30×30 map, for gridworld problems with unknown goal states.

than the nearly exhaustive search performed by the uninformed agent. This is particularly true in larger maps (Figure 5), and demonstrates the value of our approach. Furthermore, The ability to propose new classes prevents the algorithms past experience from overwhelming the information in the current set of observations. Thus, in both domains, the algorithm finds a good estimate of the true underlying prior and successfully uses its estimate to improve exploration in new MDPs.

6. Summary and Future Work

Our work is motivated by the goal of transferring knowledge between different but related RL tasks. Hierarchical Bayesian models provide an attractive framework for representing, learning, and reasoning about shared knowledge. To our knowledge this is the first work that considers applying Bayesian RL in the context of multi-task learning, which is arguably the learning setting where Bayesian methods may have the most impact. Our approach was based on a straightforward, but novel, combination of hierarchical Bayesian models and myopic action selection strategies. Our experiments in a colored gridworld MTRL problem demonstrated the potential for the model to transfer knowledge across MDPs to speed up convergence to the optimal policy in a new problem.

In future work, we plan to make our algorithm more computationally efficient. The primary computational bottleneck is our current action selection strategy, where after each resampling step, we solve an MDP. This works effectively for the domains we have considered and in many cases it may be possible to construct domain specific planners that are efficient. However, a more general solution requires additional consideration of the action selection mechanism in the Bayesian context. Another direction we are investigating is to model shared structure in the optimal value functions and policies. An eventual goal is to develop a single Bayesian model that can coherently combine evidence about all of these quantities and use the information for effective transfer between MDPs.

Acknowledgements

We thank the reviewers for their helpful comments and suggestions. We gratefully acknowledge the support of Defense Advanced Research Projects Agency under DARPA grant FA8750-05-2-0249.

References

Banerjee, B., & Stone, P. (2007). General game learning using knowledge transfer. *Proceedings of the 20th International Joint Conference on Artificial Intelligence.*

Dearden, R., Friedman, N., & Andre, D. (1998a). Model-based Bayesian exploration. *Proceedings of the 15th International Conference on Machine Learning.*

Dearden, R., Friedman, N., & Russell, S. (1998b). Bayesian Q-learning. *Proceedings of the Fifteenth National Conference on Artificial Intelligence.*

Duff, M. (2003). Design for an optimal probe. *Proceedings of the 20th International Conference on Machine Learning.*

Konidaris, G., & Barto, A. (2006). Autonomous shaping: knowledge transfer in reinforcement learning. *Proceedings of the 23rd international conference on Machine Learning* (pp. 489–496).

Mehta, N., Natarajan, S., Tadepalli, P., & Fern, A. (2005). Transfer in variable-reward hierarchical reinforcement learning. *Workshop on Transfer Learning at Neural Information Processing Systems.*

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, *9*, 249–265.

Strens, M. J. A. (2000). A Bayesian framework for reinforcement learning. *Proceeding of the 17th International Conference on Machine Learning.*

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

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*, 285–294.

Wang, T., Lizotte, D., Bowling, M., & Schuurmans, D. (2005). Bayesian sparse sampling for on-line reward optimization. *Proceedings of the 22nd Internationl Conference on Machine Learning.*