Multi-Task Reinforcement Learning Using Hierarchical Bayesian Models

Justin Bare

1. Goals

For this project, the objective was to build a working implementation of a multi-task reinforcement learning (MTRL) agent using a hierarchical Bayesian model (HBM) framework described in the paper "Multi- task reinforcement learning: A hierarchical Bayesian approach" (Wilson, et al. 2007). This agent was then to play a modified version of the game of Pacman. In this version of the classic arcade game, a series of episodes are played in sequence and the properties of the map in each episode can differ significantly. The end goal was to show that the new agent performs better than a standard Q-learning agent in this version of the game which presents different types of maps which may have different optimal policies.

These goals were partially met, but the more advanced algorithms which allow the HBM agent to be extremely adaptable will require more time to implement correctly. However, the initial results clearly show the benefits of using an HBM agent in the MTRL setting.

2. Problem Specification

The MTRL problem assumes that the agent will be presented with a sequence of Markov Decision Processes (MDP's) drawn from a distribution. Each MDP is defined by parameters (S, A, T, R) which are the states, actions, transition probabilities, and rewards. Here it is assumed that S and A are constant for all MDP's in a sequence, however, T(s, a, s') and R(s, a, s') for all s, a, and s' can change significantly depending on the distribution they are drawn from. Therefore, the agent must infer these transition and

Paper submitted for final project of CSE573: Artificial Intelligence, taught by Dan Weld at the University of Washington. December 13, 2012

reward dynamics from observations as it acts in each MDP in order to reach the optimal policy for the MDP.

For example, in the Pacman setting, the agent could be presented with one map in which there is a large negative reward for each time step and another MDP in which there is no reward for each time step, but selecting an action most likely causes the agent to do the opposite of that action (e.g. selecting “down” will most like cause the agent to go “up”). These two maps probably have very different optimal policies.

3. System Design

To deal with the issues presented by an MTRL problem, an HBM is a useful tool. Using an HBM, an agent can categorize each MDP it encounters with similar MDP’s it has experienced before, which allows for transfer of knowledge about probable dynamics in the current MDP. This concept is illustrated in the figure below:

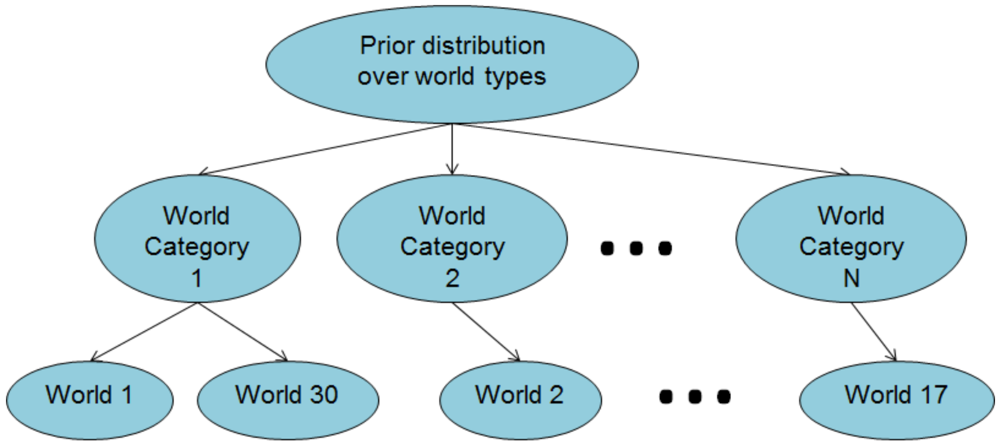

Figure 1: An example HBM. As each MDP (here “world”) is encountered, the HBM categorizes it according to the inferred transition and reward dynamics and updates the prior distribution over MDP categories, if necessary. In this particular example, when the agent encountered World 30, it found that the dynamics were similar to those in Category 1. Therefore, the agent can now try to solve World 30 as if it was drawn from the more specific distribution over MDP’s in Category 1, rather than the possibly huge distribution over all possible MDP’s.

For the purposes of this project, the HBM framework was divided into two steps to implement:

1. Inference of the dynamics of a novel MDP given a fixed set of categories: In this case, the set of categories is fixed and finite and the parameters of the categories themselves are fixed. The agent is given a prior over categories and a prior over MDP's given each category. The main challenge is using observations (s, a, s', r) of the agent acting in the current MDP to infer which category of the HBM best fits this MDP. Then the agent can sample MDP's from this category's distribution and solve them using the value iteration technique, and then apply the resulting policy in the current episode.
2. Dynamic category management: In this next step of implementation, the categories can change over time. The agent is given a prior over categories and a prior over MDP's given each category, but these priors can be completely removed and replaced by different ones as the sequence of MDP's progresses. This dynamic HBM is managed assuming the categories are drawn from a Dirichlet process, for which we can define a Gibbs sampler to sample possible category assignments for all MDP's that have been seen until a sufficiently high probability assignment is reached.

The specific algorithms used in this implementation are described in detail in the paper by Wilson et al., which is available online (http://machinelearning.org/proceedings/icml2007/papers/463.pdf).

At this point, step 1 is working, but step 2 is not yet able to correctly manage the HBM dynamically. This part of the algorithm proved to be much more difficult to design than the first and completing it will require much more in depth study of the particular Gibbs sampler for Dirichlet processes used in the paper. However, inference with fixed classes works for what it was designed for, as is discussed in the next section.

4. Experiments

To test the new agent's algorithms, the performance was compared to that of a Q-learning agent. Unfortunately, it was difficult to change the Pacman program to make it work in an MDP framework without completely rewriting much of the environmental and gameplay code. However, another program called Gridworld was already designed using an MDP framework, so this game was used for testing instead. Although Pacman might be more enjoyable to watch, this change is not significant in terms of testing the effectiveness of the underlying algorithms of the agent. An example map of the Gridworld game is shown below.

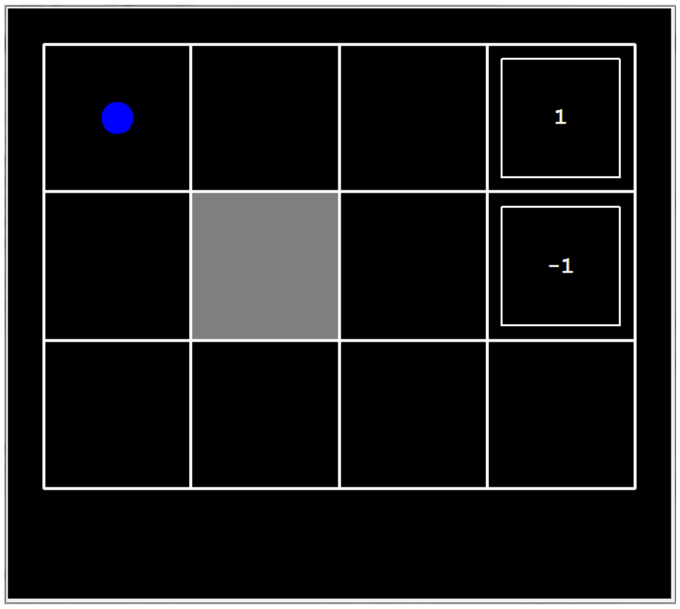

Figure 2: Example Gridworld map. The agent is represented by the blue dot, showing the location of its current state. The states with numbers in them are terminal states which give the reward shown. The agent's available actions at each non-terminal state are up, down, left, or right. The transitions between states and the rewards for transitions are drawn from a prior distribution as described in the MTRL problem specification.

The parameters which were changed in the sequences of MDP's tested were mean noise and mean living reward. The noise parameter was sampled from a distribution over [0, 1] and governs how likely it is for a particular action to result in a transition orthogonal to that action direction. For example, if noise = 0.4 and the agent selects 'up', it will transition 'up' with probability 0.6, 'right' with probability 0.2, and ‘left’ with probability 0.2. The living reward parameter was tested with a distribution over [-10, 0], which determines the penalty for moving once in Gridworld.

For all tests, the distribution from which the MDP’s were drawn was constant. First the agent was tested when the prior was exactly the distribution that the MDP’s were drawn from. Next the agent was tested with priors that were moderately different from the actual distribution. Finally it was tested with priors which were very different from the actual distribution. The three test cases and results are shown in the following figures. Each test used a sequence of 200 MDP’s.

		Actual MDP Distribution	Test 1Priors	Test 2 Priors	Test 3 Priors
Category 1	Mean noise	0.1, 0.01	0.1, 0.01	0.3, 0.01	0.5, 0.01
	Mean living reward	-0.1, 0.1	-0.1, 0.1	-1.0, 0.1	-10.0, 0.1
Category 2	Mean noise	0.9, 0.01	0.9, 0.01	0.7, 0.01	N/A
	Mean living reward	-6.0, 0.1	-6.0, 0.1	-2.0, 0.1	N/A
P(Category 1)		0.5	0.5	0.7	
P(Category 2)		0.5	0.5	0.3	

Table 1: Tests. Each entry for the means is a Gaussian distribution denoted by two numbers: (mean), (variance). In test 3, a second category was not given in the prior, so the agent would assign all MDP’s to the same category no matter what they were drawn from.

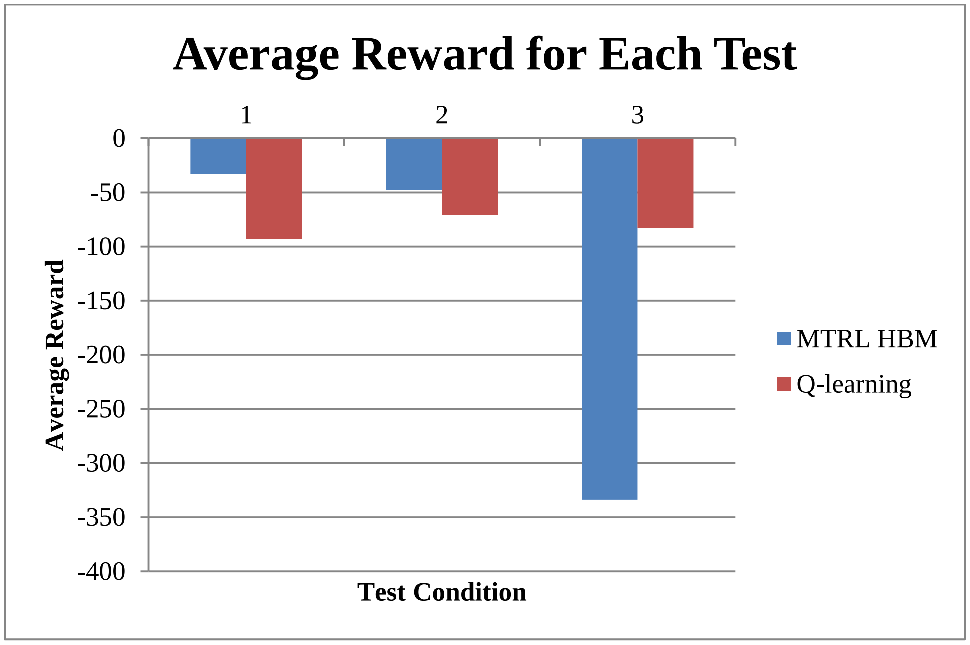

Figure 3: Results for each test case. The HBM agent attains higher average rewards than the Q-learning agent when given perfect and ok prior information. However, when the prior is very wrong, the HBM agent does much worse.

In test 3, the HBM agent does much worse because inference over a fixed set of classes does not allow any adaptation of the model if conditions are completely different than the prior. This is when the second step of the implementation is needed, for dynamic category management.

5. Conclusions

The HBM agent shows promise for MTRL problems. However, its success in real settings is dependent on being able to dynamically manage the HBM effectively, which is a mathematically complex task requiring fine-tuning of several parameters. For the future, this issue will be figured out in order to make the agent useful, and the Pacman program will be adapted for use in an MDP framework. From there, it would be worthwhile to continue this line of research into applying the HBM inside Q-learning, possibly as a hyper-feature for approximate Q-learning, in which features of states are used, rather than the states themselves. This could be a very powerful approach because it would make for a very adaptable, model-free, reinforcement learning agent, which could operate in situations with differing sets of states in the sequence of MDP's, which is something the model-based implementation described in here cannot do.

References

Wilson, A., A. Fern, S. Ray, and P. Tadepalli. "Multi- task reinforcement learning: A hierarchical Bayesian approach." *In Proceedings of ICML 24*, 2007: 1015– 1022.