Bayesian Imitation Reinforcement Learning with a Multiple Number of Mentors

Jean-Noël Rivasseau
Department of Computer Science
University of British Columbia
Vancouver, Canada
jnriva@cs.ubc.ca

Abstract

In this project I explore the possibility of Bayesian Imitation Reinforcement Learning, with more than one "mentor". Using recent techniques for exploration and imitation within a reinforcement learning framework, I conduct several experiments. They involve one learning agent, trying to explore its environment and derive an optimal policy given known objectives, and a number of other agents called mentors, who already have full knowledge of their actions results and a fixed optimal policy for their respective goals. The observation of other agents allows the new agent to improve its own Bayesian exploration technique.

Introduction

The problem of interaction between an agent and a given environment is at the heart of artificial intelligence. At the beginning of its lifetime, an agent can have high uncertainty about the result of its actions in a totally unknown and unexplored environment, in terms of transitions from one state to another; or even about its own goals (he may not know with certainty which states will result in positive rewards). How should he then start acting? The reinforcement learning paradigm addresses this issue, and provides a framework in order to allow the agent to learn, through various ways, what are its best actions in a given state. Once he has accumulated sufficient beliefs about its surrounding world, the agent can start to follow a policy, which will eventually allow him to earn rewards (reach its goals). Two central issues with reinforcement learning are the optimality of the derived policy, and also the rate of convergence to that policy, which can be seen as the speed with which he explores and exploits the new available information.

The generality and power of the reinforcement learning model has given birth to a considerable amount of work. If the dynamics of the underlying model are known, the problem becomes simpler and a number of techniques, based on dynamic programming, have been developed to derive an optimal policy for a given *certain* model. The most used is probably value iteration (Bellman 1957), which computes what is commonly called the *q-value* of every state in the model. This value represents the (discounted) rewards gained by the agent if he starts in that state and follow the theoretically optimal policy from now on. Once these values are known, the agent's choice is trivial: he should pick up the action leading to the highest q-valued state. An alternative to value iteration is policy iteration, which updates the policy and does not deal with the q-values. Improvements and theoretical results regarding both of these methods are available in the literature; see for example (Puterman & Shin 1978).

However, solving a known model is, arguably, only the easiest part of the more general reinforcement learning problem. If the agent has uncertainty about the underlying model, which is the main case in reinforcement learning, he must interact with its environment in order to get precious information about the dynamics of its actions, which will in turn reduce its uncertainty and allow him (by mean of the previously described methods) to compute an optimal policy. This brings us to the central issue of *"exploitation versus exploration"*. Obviously, exploration is costly (time-consuming) and may even be dangerous as it may lead to negative rewards. Thus, learning quickly (with fewer exploration steps) and efficiently is a problem that has been deeply studied. Several popular techniques are currently available, either model-free methods where the agent doesn't explicitly learns a model, such as Q-learning, or model-based ones such as prioritized sweeping (Moore & Atkeson 1993) . In the later ones we try to learn the dynamics of our model before using it to derive a policy.

A recent work (Dearden, Friedman, & Andre 1999) presents an approach to model based reinforcement learning using a Bayesian framework. The agent essentially maintains a distribution of probabilities of the various possible models, updating that distribution as he acquires more information about its environment. Sampling methods are used to compute appropriate q-values from our probability distribution, and we use those q-values to select actions as usual. This Bayesian approach to reinforcement learning exploration compares well to other previous model based methods, and has certain specific advantages: there is no required parameter tuning, for example.

Building on this work, Price and Boutilier (Price & Boutilier 2003) have added Bayesian imitation to this framework in a multi-agent setting. In such a setting, agents are assumed to be somehow similar, not necessarily in their objectives, but in the results of their actions (state transitions probabilities are assumed to be the same for everyone). An agent cannot observe directly the actions chosen by others, but get to observe their states. These assumptions are very

realistic, and can even be used to model a competitive environment when agents would not freely disclose their actions to others, since all that is observed is the outward consequences of these actions. In that setting, agents using information made available by their mentor's behavior learn quicker. It should be noted that this "imitation" is not necessarily about explicitly duplicating the policy of other agents (since they may have different and even opposite goals). It is more about exploiting the additional available information in order to make better posterior updates for the agent's distribution over possible models.

This project will focus on understanding the framework used by (Price & Boutilier 2003), which builds considerably on previous complex work. That framework will be used to build an experiment with a variable number of agents. One will be considered the "newcomer", whereas all the other ones will follow previously defined policies optimized for their respective different goals. (Price & Boutilier 2003) only considered the case of two agents (a learner and a mentor), but it is straightforward to deal with multiple mentors. Since it has already been shown that imitation can lead to a performance increase, the new hypothesis that we will test is the following: "Can the use of more than one mentor can further increase the benefit of Bayesian Imitation, in terms of quicker exploration ?". In other terms, are multiple mentors useful or not?

Background

This section will introduce the framework and notations used in this project. I adopted the same general notations as in (Price & Boutilier 2003) and (Dearden, Friedman, & Andre 1999), and the reader is referred to these two papers for all the details of the described model. For explanations of the basic concepts behind Markov Decision Processes (MDP), see (Kaelbling, Littman, & Moore 1996).

I consider an MDP $\langle \mathcal{S}, \mathcal{A}_0, R_0, D \rangle$, where $\mathcal{S}$ is the state set, $\mathcal{A}_0$ the action set, R_0 the reward function, and D the dynamics. $D^{s,a}$ thus refers to the transition probability distribution for reaching various states, after the agent selected action a in state s. It is assumed that the agent always knows the rewards function $R_0 : \mathcal{S} \rightarrow \mathbb{R}$. Its only uncertainty is about the dynamics D of the MDP, so from now on when I will talk about the probability distribution over possible MDP models I will refer to the distribution over dynamics $P(D)$. The agent tries to learn the true dynamics of this MDP.

The main computational task in our framework and algorithm is to update our beliefs, the probability distribution $P(D)$, with the results obtained from a new data point. Such a data point consists in an action and the observed transition from the previous state, so we will represent it as $\langle s, a, s' \rangle$. For the update, we need a prior for $P(D)$; (Dearden, Friedman, & Andre 1999) choose to place a prior satisfying parameter independence. This means that P can be written as a product over all local densities $P(D^{s,a})$; these local densities specify the transition distribution from state s and action a. We then place Dirichlet priors[1] over these local densities functions; so in the end we only need one parameter for every $P(D^{s,a})$, namely, a Dirichlet vector $\mathbf{n}^{s,a}$ which has a length equal to the number of states. The $n^{s,a,s'}$ represent the actual Dirichlet parameters [2] for each of the possible successor states s'.

It can be shown that after an update, if the prior satisfies parameter independence, the posterior also satisfies parameter independence. What's even better, with Dirichlet priors the posteriors are still given by a Dirichlet distribution. All that is needed in order to perform an update is to update the Dirichlet parameters: if we have the data vector $\mathbf{c}^{s,a}$ where $c^{s,a,s'}$ is the number of times we observed transition from s to s' with action a, with a prior of $\mathbf{n}^{s,a}$ our posterior is just $\mathbf{n}^{s,a} + \mathbf{c}^{s,a}$. This is sufficient to improve the agent's belief about its environment.

Our assumptions about the other agents is that they share the exact same MDP, except for the reward function that can be different (thus allowing different goals for every agent). The states, actions available, and dynamics (probability transitions) are identical between all the agents. In fact, the assumption about the actions can be relaxed a little bit, (Price & Boutilier 2003) only assumes that for every action taken by the mentor, there is one action available to the newcomer that is equivalent (in terms of transition distribution).

Imitation with Multiple Mentors

So where does the observation of other agents (mentors) play a role? The advantage brought by Bayesian Imitation is a better update of the Dirichlet parameters, resulting in the end in a "quicker exploration" (in fact a quicker reduction of the uncertainty surrounding the MDP dynamics). Instead of updating the Dirichlet parameters with *only the agent's new data point*, we also take into account *the data points generated by the other agents*. If we could observe directly the mentor's actions, we would just observe a new data point exactly as we observe a new data point for the agent itself, in the form $\langle s, a, s' \rangle$. Then we would just treat the mentor just as another chance for us to perform a "free exploration action" (free in terms of time, not of computational requirements), and we would update the Dirichlet parameters with the sum of our counts and the mentor's counts.

However, since we don't observe the actions of the mentors but only their states, things are not that simple. There are mainly two ways to proceed here. We can try to learn the actions the mentors are taking (which is hard), and use this to update our Dirichlet parameters. So we will be in some way trying to learn the Q-values for the agents. Or we can also not care about the mentor's exact actions and only observe their "moves". Thus we obtain the mentors' V values and then use them somehow when we compute our own V values. In this experiment, we chose the first approach (the one also taken by (Price & Boutilier 2003)), which is more

[1] A Dirichlet distribution is a probability distribution, which is a generalization of the Beta distribution to n random variables instead of 2.

[2] As a Beta distribution has two parameters, a Dirichlet distribution has n parameters. It is assumed that the reader is already familiar with Dirichlet distributions; if needed, see for example (Evans, Hastings, & Peacock 2000) for a description of this distribution.

complex but allows everything to stay Bayesian. It also allows us to gain information by observing the mentors even if their policy will never hit a reward (or penalty). If we only used V-values, it is not clear (although it would have to be investigated) we would gain something by observing mentors that never encounter rewards, since their V-values are always 0 for each of their states.

So the newcomer keeps a belief about the mentor's policy (which action it is likely to select in every state), and also updates it as it learns more about the mentor's transitions. How to update this belief is crucial; intuitively, it is clear the newcomer can only guess which action the mentor selected in a given state based on its own information (in other terms, we can try to guess which actions the mentors took only once we have been ourselves in the same state and "explored" the different actions available at this state). The assumption of the same action space for the newcomer and mentors is also obviously needed here. The details of a possible correct update rule are given in (Price & Boutilier 2003).

Once we have a belief over the mentor's policy, we can update, for each observation of our own moves and of the mentor's moves, our Dirichlet parameters. It is important to understand that these Dirichlet parameters govern everything in a Bayesian framework. They will be used to sample MDPs, which will in turn be used to compute the Q-values for the newcomer as will be explained in the next section.

For the case of two agents (one learning agent and one mentor), if we denote $Pr(\pi_m^s)$ our belief distribution about the mentor's action in state s, and H_a, H_m the history (sequence of observed transitions) of respectively the agent and the mentor, we obtain the following complete posterior update rule :

$$\begin{aligned} & P(D^{s,a}|H_a, H_m) \\ = \; & Pr(\pi_m^s = a|H_a, H_m)P(D^{s,a}; \mathbf{n}^{s,a} + \mathbf{c}_a^{s,a} + \mathbf{c}_m^{s,a}) \\ & + Pr(\pi_m^s \neq a|H_a, H_m)P(D^{s,a}; \mathbf{n}^{s,a} + \mathbf{c}_a^{s,a}) \end{aligned}$$

This last equation considers two cases. First, the case where we believe the action taken by the mentor is the same as the one we are currently considering (a); then we can update the Dirichlet with our counts $\mathbf{c}_a^{s,a}$ but also with the mentor's counts $\mathbf{c}_m^{s,a}$. The second case is when the action taken by the mentor is different from the one just considered; then we update the Dirichlet by adding only our counts $\mathbf{c}_a^{s,a}$. The complete update is just the sum of these two cases, weighted by our belief about the mentor's policy. This gives us the correct expectation.

In our experiment, we generalize to the case of k mentors in addition to the main learning agent. The update equation we get in this case would then be, with the same notations:

$$\begin{aligned} & P(D^{s,a}|H_a, H_{m_1}..H_{m_k}) \\ = \; & \textstyle\sum_{a_1} .. \sum_{a_k} \prod_{i=1}^{k} Pr(\pi_{m_i}^s = a_i|H_a, H_{m_i}) \\ & \cdot P(D^{s,a}; \mathbf{n}^{s,a} + \mathbf{c}_a^{s,a} + \textstyle\sum_{i=1}^{k} \delta_{a,a_i} \mathbf{c}_{m_i}^{s,a}) \end{aligned}$$

where $\delta_{i,j}$ is the Kronecker symbol[3]. In this equation we sum over all possible actions $a_1..a_k$ for each agent; but actually we have to consider, like in the one-mentor situation, only the two cases when the action is or isn't equal to the currently considered action a. Note that these two cases are embedded in this last equation (with the help of the Kronecker symbol); behind the more complex notation, this second equation is exactly similar to the first one. In fact, to add multiple mentors, we simply need to perform a loop over the update rule for a single tutor.

Intuitively, more agents seems always better, *whatever their goals may be*, since it can lead to better updates of the Dirichlet parameters. However, whatever the number of agents, several problems arise when dealing with imitation, as we will see in the results of the experiments. One of the main obstacle is due to the fact that the main agent needs to have a fairly correct belief of a mentor m_i policy π_{m_i} in order to correctly exploit the mentor's transitions, and as we already pointed out such a belief is hard to obtain in a short amount of exploration time.

[3]The Kronecker symbol, $\delta_{i,j}$, is defined such that $\delta_{i,i} = 1$ and $\delta_{i,j} = 0$ for $i \neq j$

- Initialize our beliefs about the environment and the mentors policies, by initializing all Dirichlet vectors to a constant vector
- Loop for the desired number of steps
  - Sample k MDPs from our current belief distribution. That amounts to sampling the transition probabilities from our current Dirichlet distribution for each state $\langle s, a, s' \rangle$;
  - Use Value-iteration to solve each MDP;
  - Obtain an expected value for the states Q-values, by taking the average of the Q-values for the k sampled MDPs;
  - Use Value of Information techniques to compute what's the best action to take;
  - Move the agent and all mentors according to the environment *true* dynamics;
  - Observe the transitions that have just occurred, and update our belief of the MDP distribution by updating the Dirichlet parameters;
  - Update our belief of the mentors policy.
- Print results obtained

Table 1: Structure of the Algorithm

Algorithm

We will now proceed to a general high level description of the Bayesian learning algorithm. The structure of this algorithm (used for our experiment) is described in Table 1.

In fact, this algorithm uses a lot of different techniques from the Reinforcement-Learning field, and each step will now be briefly explained in more details. First, one needs to know how to sample from a Dirichlet distribution; this problem is hopefully tractable and can be solved by sampling from a Gamma distribution; see (Ripley 1987) for such procedures. Then, Q-Values of each state for every sample must be computed. We use Value-Iteration, a fairly standard algorithm for solving an known MDP (that is, a MDP which dynamics are known without uncertainty). Theoretical results guarantee convergence of this value-iteration algorithm; see (Kaelbling, Littman, & Moore 1996).

It should be noted that all this sampling and Value-Iteration are heavy computational tasks. Thus it is naive to actually sample k MDPs at each step, and solve each of them full, because in fact our distribution over possible MDPs will not change much with only one update (one step). Several more advanced methods are available to reduce computations; Importance Sampling, for example, tries to keep the same MDPs from one step to another, just updating "weights" on this MDPs to reflect the updates in the distribution. We re-sample new MDPs only when the weights go too small and go below a certain threshold. Importance Sampling for Bayesian Exploration is described in (Dearden, Friedman, & Andre 1999).

Once each sample is "solved", it is easy to compute an expected Q-value over all these solved samples. Since we are dealing with an exploration problem, we need to take a decision for our action based not only on our expected reward but also on what new information (which may potentially increase our reward) we get by exploring an unknown state. For this we adopt the Value Of Information exploration technique describe by Dearden et al. The general idea is to define an incentive for exploring states based on the gain in rewards if we knew the true value (perfect information) of that state. It seems to me that when dealing with Bayesian Imitation, another concept could be introduced and superposed to this one: the value of information should also reflect the gain in rewards based in a better knowledge of the mentor's policy, which would allow less uncertain use of the mentor's moves. Thus the agent should be encouraged to explore states which will result in a diminution of the uncertainty surrounding the mentor's policy.

Finally, the updates of our belief of the MDP distribution, as well as our belief of the mentors policies, were detailed in the previous sections, and mainly consist in Dirichlet updates.

Experimental Results

Setting

All of this experiments were ran using the same background setting. Agents move along a "maze" consisting of 10 by 10 squares. The possible states for the agents are just equivalent to their position in the maze, which can be represented as coordinates (x,y). In this maze, agents can take four actions at any given position: North, South, West, or East. Each of this action will take them to the corresponding square with a probability of 0.9 (or 0 if the square is indeed a wall); with a probability of 0.1 they will not move (or 1 if it's a wall). The "learning agent" initially thinks that the result of every action can get him to any surrounding square, or the square itself (so he has at the beginning, for each action, a prior consisting in uniform probabilities for the 9 possible outcomes of the action). He has no knowledge of the walls in the maze, except for the "outer walls" that bound the maze. These walls (at the edges of the maze) are known by the agent, and they are accordingly incorporated into its prior beliefs (for example, an agent at position (0,5) knows that the results of any of its actions can't get him to the "forbidden" positions (-1,5) or (-1,4).

The agent starts at the position (0;0) and wants to get to the position (9;9) where he will fetch a reward and be instantaneously teleported back to (0;0). The reward is discounted with a discount factor of 0.9 for each passing step. Note that changing the discount factor doesn't have any fundamental effect on our experiments for the comparisons we aim to make (exploration with or without a given number of mentors). Any mentor will use a fixed policy predefined for a configuration of the maze. This policy doesn't necessarily get him to the reward, as I wanted to see what would happen if the mentors had different goals.

The learning agent has to learn (for *every* state) that, for example, the action labeled "South" will effectively get him South with a high probability. He also needs to learn where are the walls (they correspond to states where some actions don't have the "expected" result, although the agent doesn't know what is the expected result...). Once his beliefs about the results of its actions are fairly good, he will be able to get to the reward and his behavior will converge to an optimal policy.

I ran 3 distinct experiments. Two consider the simplest possible maze, without any walls. In the first of these two, mentors actually follow an optimal policy to fetch the reward. In the second, they just move around the maze but never get to the reward position. The last experiment is done with a big wall separating the maze in two parts, so the agent actually has to find the only possible way across. The mentors are using an optimal policy in this third experiment.

The setup for these 3 different experiments can be found in Figures 1, 2 and 3. The bold arrows represent the policy of the mentors.

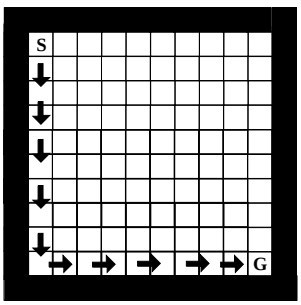

Figure 1: First experiment

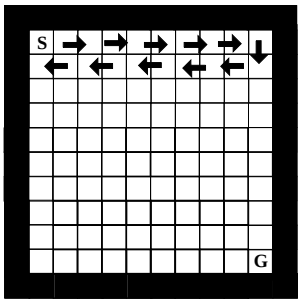

Figure 2: Second experiment. Mentors don't follow an optimal policy

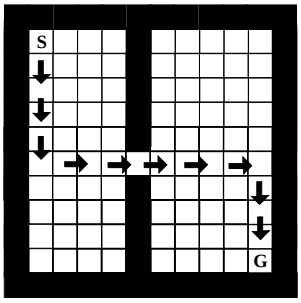

Figure 3: Third experiment. This time a wall is present

Results

In Figure 4, 5 and 6, I present the results of the different experiments as the cumulated rewards obtained in the last 100 steps (sampled at intervals of 50 steps). Other numbers could be given, such as the number of steps needed to fetch the first reward, the time needed to converge to the optimal policy, etc... However these numbers are very sensitive to the inherent randomness of the experiment; for example, one can use a quasi-optimal policy but there is always a slight probability that it will deviate at one step, so it is hard to really judge when the optimal policy is reached. Displaying the cumulative rewards is a very good way to estimate graphically how the agent is learning.

Analysis of the Results

The first and second experiments (Figure 4 and 5) show that imitation is not really useful for such a simple setting. All agents performed more or less identically, and imitation didn't allow better exploration. This is without doubt due to the fact that even if we can observe the transitions of the mentors, and we imitate them, we still don't know which action they effectively choose. Thus in the end, we often fail when trying to imitate the mentors and it takes time for us to come back on the "mentor's tracks". No real gain is obtained, compared to the simple Bayesian exploration, by adding imitation.

It is worth noting, however, that with mentors that follow an optimal policy, the agent always converge to this particular policy (this is not shown in the results, but I noticed that).

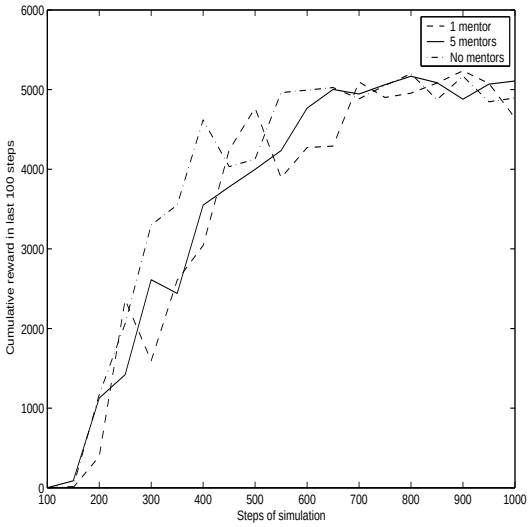

Figure 4: No walls with Optimal Policy

When there are no mentors, the agent obviously converge at random to one of the many available optimal policy. This demonstrates the fact that our algorithm is working, even if we cannot see the difference in performance for the first experiment.

One other important remark is that imitation doesn't hurt if the mentors follow a policy that is not an optimal policy for the agent's own goals. This is why I included the results of the second experiment: this experiment shows that even if the mentors follow a policy not useful for the agent, he is still able to find - without any penalty in exploration time - the correct path to reward, and after learning his behavior does converges to an optimal policy. This illustrates the fact that within a Bayesian framework, the mentors could have different goals that the agent, but that doesn't necessarily penalize him. He will just ignore the mentors if they don't have a useful policy.

The third experiment, more complex, clearly shows the benefit of imitation. Without any imitation, the agent learns very slowly. After the 1000 steps of the experiment, it wasn't even yet at the optimal policy. On the contrary, with imitation, the "learning agent" was able to overcome the wall obstacle, and he reached the optimal policy within the time frame of the experiment. With 5 mentors, the agent learns even faster, and as shown on Figure 6, is able to fetch high cumulated rewards much sooner than the agent observing only a single mentor. Thus it is demonstrated than more mentors do help, and even if the difference in performance is not so high between 1 and 5 mentors than between 0 and 1 (which seems perfectly logical), it is still very real.

Intuitively, these results make sense. Within a simple setting, observing which states (positions) are valuable to the mentors isn't really useful, since anyway we pretty much know that already (since we want to get to the reward state, we value highly the states that are close to it). Following the mentor's optimal policy is not better than following any

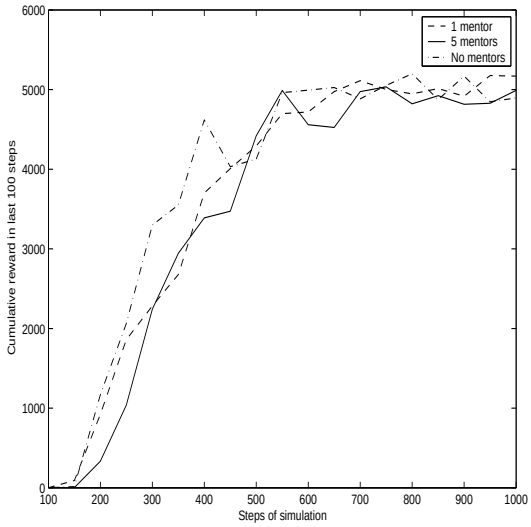

Figure 5: No Walls with Different Policy

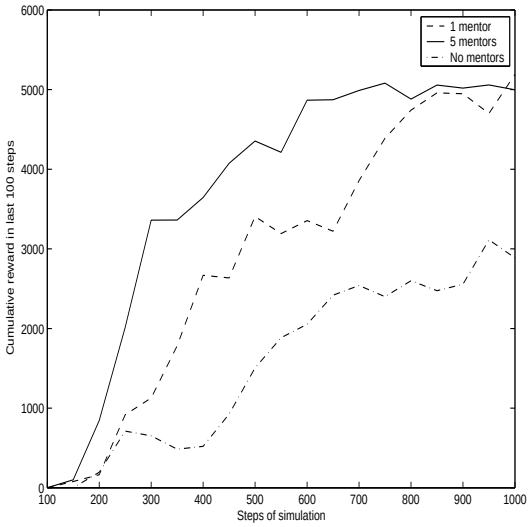

Figure 6: Wall results

other optimal policy, and since the hard task during exploration is, in this case, to learn the correct mapping between actions and state transitions, imitation doesn't help much. However, with walls, the hard task is no more to find a correct mapping, but on the contrary to find the valuable states (since the challenge is to get pass the wall: and there is only one state where we can do that - the "hole" in the wall). So here observing what the mentors do (they effectively get to that desirable state) is very helpful... To summarize the results of my project, I would say that imitation can make a very real difference when the difficulty is to learn the desirability of the states. If the difficulty is to learn the mapping of actions to state transitions, imitation has much less appeal.

Conclusion

This project was an investigation of the Bayesian Imitation and Exploration framework for Reinforcement Learning. The goal was to run several experiments in order to assess the benefits of imitation, and if more mentors would help even more. In the experiments I have ran, it turns out that indeed more mentors is an improvement, but the performance gain really varies from setting to setting. In a simple setting, the benefits of imitation are almost reduced to zero, due to the fact that the agent has weak beliefs about the mentors actions.

Future research on Bayesian Exploration and Imitation could take many forms. Many more interesting experiments can still be done; in particular, a simple prolongation of this project would be to find (for a given setting) the optimal number of mentors: the limit after which further increasing the number of mentors doesn't bring any significant performance increase. The algorithm could also be rewritten in order to directly manipulate the V-values of the mentors as suggested earlier; this wouldn't be completely Bayesian anymore but would probably lead to the same results. Finally, extending Bayesian Imitation to a different action space for the mentor and the agent, suggested by (Price & Boutilier 2003), is something that remains to be done.

References

Bellman, R. 1957. *Dynamic Programming*. Pinceton University Press, Princeton, NJ.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based bayesian exploration. In *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence.*, 150–159. San Francisco: Morgan Kaufmann.

Evans, M.; Hastings, N.; and Peacock, B. 2000. *Statistical Distributions*. New York: John Wiley & Sons.

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

Moore, A. W., and Atkeson, C. G. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning* 13:103–130.

Price, B., and Boutilier, C. 2003. A bayesian approach to imitation in reinforcement learning.

Puterman, M., and Shin, M. 1978. Modified policy iteration algorithms for discounted markov decision processes. *Management Science* 24:1127–1137.

Ripley, B. D. 1987. *Stochastic Simulation*. Wiley, NY.