Active Learning in Partially Observable Markov Decision Processes

Robin JAULMES, Joelle PINEAU, Doina PRECUP

McGill University, School of Computer Science, 3480 University St., Montreal, QC, Canada, H3A2A7

Abstract. This paper examines the problem of finding an optimal policy for a Partially Observable Markov Decision Process (POMDP) when the model is not known or is only poorly specified. We propose two formulations of the problem. The first formulation relies on a model of the uncertainty that is added directly into the POMDP planning problem. This has some interesting theoretical properties, but is impractical when many of the parameters are uncertain. Our second approach, called MEDUSA, is an instance of active learning, whereby we incrementally improve the POMDP model using selected queries, while still optimizing reward. Results show a good performance of the algorithm even in large problems: the most useful parameters of the model are learned quickly and the agent still accumulates high reward throughout the process.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) are a well-studied framework for sequential decision-making in partially observable domains (Kaelbling et al, 1995) . Many recent algorithms have been proposed for doing efficient planning in POMDPs (Pineau et al. 2003; Poupart & Boutilier 2005; Vlassis et al. 2004). However most of these rely crucially on having a known model of the environment. On the other hand, experience-based approaches have been proposed which rely strictly on experimentation with the system to learn a model which can then be used for planning (McCallum, 1996; Brafman and Shani, 2005; Singh et al. 2004). Yet these typically require very large amounts of data, and are therefore impractical for all but the smallest problems. In practice, we would often prefer a more flexible trade-off between these two extremes.

In particular, in many applications it is relatively easy to provide a rough model, though much harder to provide an exact one, and so we would like to use (small amounts of) experimentation to improve our initial model. The overall goal of this work is to investigate POMDP approaches which can combine a partial model of the environment with direct experimentation, in order to produce solutions that are robust to model uncertainty, while scaling to large domains. The idea behind our work is the following: we assume that uncertainty is part of our model and design our agent to take it into account when making decisions. More importantly, the model should be updated based on new experience.

We based our work on the idea of *active learning* (Cohn et al. 1996). It is a well-known technique in machine learning for classification tasks with sparsely labelled data.

The goal is to select which examples should be labelled by considering the expected information gain. As detailed by Anderson and Moore (2005), these ideas extend nicely to dynamical systems such as HMMs. Therefore, we will assume the availability of an oracle that can provide the agent with exact information about the current state, upon request. It is a reasonable assumption in a number of real-world POMDP domains, especially since our work is motivated by applications in robotics and dialogue management. In these domains, it is routine for a human to be involved in the initial calibration of the robot. However, we will assume that using the oracle is expensive and reserved for the learning phase, where we will use it as little as possible. Note that we focus only on a model-based approach because in many applications the model of the dynamics and observation are re-usable.

Our first technique is conceptually simple, though not scalable. In essence, given a problem with model uncertainty, we extend the original problem formulation to include one additional state feature for each uncertain model parameter. This extended model is used for planning, thereby allowing us to obtain a better way of choosing actions, which is also robust to the uncertainty in the model. As discussed in Section 3, this is a straightforward extension of the standard POMDP formulation, which performs well when there are few uncertain parameters.

Our second technique, presented in Section 4, uses oracle queries while the agent interacts optimally with the environment. The query result information is used only to improve the model, not in the action selection process. In this framework the uncertainty is represented using a Dirichlet distribution over all possible models, and its parameters are updated whenever new experience is acquired.

2 Partially Observable Markov Decision Processes

We assume the standard POMDP formulation (Kaelbling et al., 1995); namely, a POMDP consists of a discrete and finite set of states S, of actions A and of observations Z. It has transition probabilities $\{P^a_{s,s'}\} = \{p(s_{t+1} = s' | s_t = s, s_t = a)\}, \forall s \in S, \forall a \in A, \forall s' \in S$ and observation probabilities $\{O^a_{s,z}\} = \{p(z_t = z | s_t = s, a_{t-1} = a)\}, \forall z \in Z, \forall s \in S, \forall a \in A$. It also has a discount factor $\gamma \in (0,1]$ and a reward function $R : S \times A \times S \times Z \to \mathbb{R}$, such that $R(s_t, a_t, s_{t+1}, z_{t+1})$ is the immediate reward for the corresponding transition.

At each time step, the agent is in an unknown state $s_t \in S$. It executes an action $a_t \in A$, arriving in an unknown state $s_{t+1} \in S$ and getting an observation $z_{t+1} \in Z$. Agents using POMDP planning algorithms typically keep track of the belief state $b \in \mathbb{R}^{|S|}$, which is a probability distribution over all states given the history experienced so far. A policy is a function that associates an action to each possible belief state. Solving a POMDP means finding the policy that maximizes the expected discounted return $E(\sum_{t=1}^{T} \gamma^t R(s_t, a_t, s_{t+1}, z_{t+1}))$. While finding an exact solution to a POMDP is computationally intractable, many methods exist for finding approximate solutions. In this paper, we use a point-based algorithm (Pineau et al. 2003), in order to compute POMDP solutions. However, the algorithms that we propose can be used with other approximation methods.

We assume the reward function is known, since it is directly linked to the task that we want the agent to execute, and we focus on learning $\{P^a_{s,s'}\}$ and $\{O^a_{s,z}\}$. These prob-

ability distributions are typically harder to specify correctly by hand, especially in real applications. For instance in robotics, the sensor noise and motion error are often unknown. To learn the transition and observation models, we assume that our agent has the ability to ask a query that will correctly identify the current state (we discuss later how the correctness assumption can be relaxed). This is a strong assumption, but is not entirely unrealistic. In fact, in many tasks it is possible (but very costly) to have access to the full state information; it usually requires asking a human to label the state. As a result, clearly we want the agent to make as few queries as possible.

3 Decision-theoretic model learning in POMDPs

The first algorithm we propose assumes that (1) the parameters of the POMDP model are not known exactly (2) the agent can perform query actions, and (3) these queries are expensive, so they should not be used too much. Based on these three assumptions, we can modify the original POMDP model in order to reflect model uncertainty explicitly. First, we increase the number of states in the POMDP: for each uncertain model parameter, we add a new state feature. This feature is typically discretized into n levels. For instance, suppose that for some pair of states $s, s' \in S$ and action $a \in A$ we know that $P^a_{ss'} \in [0.5, 1.0]$. We will discretize this interval in a certain number of bins, e.g. n, and then the state space will receive a new feature, which can take n possible values. We thereby obtain n groups of states; the transitions are such that transitions always occur between states in the same group. Second, we need to add a "query" action to our set of actions. Finally, we have to set the reward function such that it penalizes the query action adequately.

We analyze the performance of this algorithm on the standard Tiger problem (Kaelbling et al., 1995). In order to use the decision-theoretic approach to model the uncertainty in the tiger problem, we assume that we do not know the probability of the sensor providing the correct state information and consider three possible levels of this probability: 0.7, 0.8 and 0.9. Even with such a simple setting, no exact POMDP solution can be found, but the approximate planning algorithm with a finite horizon finds solutions.

Figure 1 depicts the policies found and the expected reward, as a function of the query penalty. The policies found either alternate between query and the optimal action, or never do any query at all, if the query penalty is very high. Even when no query is done, the agent still manages to learn the observation probabilities. However, we think this is an artifact of having a `Listen` action, which is in effect a noisy version of a `Query` action. The fact that some policies use the `Listen` action but not the `Query` action suggests that noisy queries may be sufficient to learn the parameters of the system.

This approach will not scale well for large POMDPs, because the number of states is multiplied by n^k where n is the number of possible values for a given parameter and k is the number of uncertain parameters. This greatly increases the complexity of the belief state and the complexity of the policy. Furthemore, the cost of the query can be very difficult to establish. The results above show that if the cost is too low, the query action is used as part of permanent policies instead of being used only in the beginning

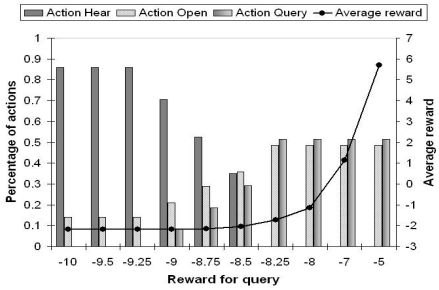

Fig. 1. Experimental results on the Tiger problem using decision-theoretic model learning. The bars indicate the % of time each action is chosen (during both learning and testing). The line indicates performance of the optimal solution obtained with each reward parameter.

to gather information about the model. On the other hand, if the cost is too high, it is likely that the query action will never be picked.

4 Active learning in POMDPs

In this section we describe an approach which draws inspiration from the field of active learning (Dearden, 1999). We will represent the model uncertainty with a Dirichlet distribution over possible models, and update the parameters of the distribution as new experience is acquired.

4.1 Dirichlet Distributions

Consider an N-dimensional multinomial distribution with parameters $(\theta_1, \ldots \theta_N)$. A Dirichlet distribution is a probabilistic distribution over these parameters. The Dirichlet itself is parameterized by hyper-parameters $(\alpha_1, \ldots \alpha_N)$. The likelihood of the multinomial parameters is defined by:

$$p(\theta_1 \ldots \theta_N | D) = \frac{\prod_{i=1}^{N} \theta_i^{\alpha_i - 1}}{Z(D)}, \text{ where } Z(D) = \frac{\prod_{i=1}^{N} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i)}$$

The maximum likelihood multinomial parameters $\theta_1^* \ldots \theta_N^*$ can be computed, based on this formula, as:

$$\theta_i^* = \frac{\alpha_i}{\sum_{k=1}^{N} \alpha_k}, \forall i = 1, \ldots N$$

The Dirichlet distribution is convenient because its hyper-parameters can be updated directly from data. For example, if instance $X = i$ is encountered, α_i should be increased by 1. Also, we can sample from a Dirichlet distribution conveniently using Gamma distributions.

In the context of POMDPs, model parameters are typically specified according to multinomial distributions. Therefore, we propose to use a Dirichlet distribution to represent the uncertainty over these model parameters. We will use Dirichlet distributions

for each state-action pair where either the posterior transition probabilities or the observation probabilities are uncertain.

4.2 The algorithm

The algorithm we propose, called MEDUSA for "Markovian Exploration with Decision based on the Use of Sampled models Algorithm" is an active learning approach that follows a familiar scenario. First, our agent samples a number of POMDP models according to the current Dirichlet distribution. The agent takes an action in the environment, and as a result, obtains an observation. At this point, the agent can decide to query the oracle for the true identity of the hidden state. If it does so, it can update the Dirichlet parameters according to the result of the query. This process is repeated until the distribution over models is sufficiently well-known. Table 1 provides a detailed description of these steps, including the implementation details.

1. Let $|S|$ be the number of states, $|Z|$ be the number of observations, and $\lambda \in (0,1)$ be the learning rate.
2. Initialize the necessary Dirichlet distributions[1].
   For any unknown transition probability, $T^a_{s,.}$, define $Dir \sim \{\alpha_1, \ldots \alpha_{|S|}\}$.
   For any unknown observation $O^a_{s,.}$, define $Dir \sim \{\alpha_1, \ldots \alpha_{|Z|}\}$.
3. Sample n POMDPs $P_1, \ldots P_n$ from these distributions. (We typically use $n = 20$).
4. Compute the (normalized) probability of each model: $\{w_1, \ldots w_n\}$.
5. Solve each model $P_i \rightarrow \pi_i, i = 1, \ldots n$. (We use a finite point-based approximation.)
6. Initialize the history $h = \{\}$
7. Initialize a belief for each model $b_1 = \ldots = b_n = b_0$ (We assume a known initial belief b_0).
8. Repeat:
   (a) Compute the optimal actions for each model: $a_1 = \pi_1(b_1), \ldots a_n = \pi_n(b_n)$.
   (b) Randomly pick and apply an action to execute, according to the model weights: $a_i = \pi_i(b_i)$ is chosen with probability w_i.
   (c) Receive an observation z.
   (d) Update the history $h = \{h, a, z\}$
   (e) Update the belief state for each model: $b'_i = b_i^{a,z}, i = 1..n$.
   (f) If desired, query the current state, which reveals s, s'.
   (g) Update the Dirichlet parameters according to the query outcome:
   $\alpha(s, a, s') \leftarrow \alpha(s, a, s') + \lambda$
   $\alpha(s', a, z) \leftarrow \alpha(s', a, z) + \lambda$ and re-normalize the Dirichlet distributions.
   (h) Recompute the POMDP weights: $\{w'_1, \ldots w'_n\}$.
   (i) At regular intervals, remove the model P_i with the lowest weight and redraw another model P'_iaccording to the current Dirichlet distribution. Solve the new model: $P'_i \rightarrow \pi'_i$ and update its belief $b'_i = b^h_0$, where b^h_0 is the belief resulting when staring in b_0 and seeing history h.

Table 1. The MEDUSA algorithm

A few aspects of this approach are worth discussing further. First, every time a new model is sampled, the agent finds the corresponding (near-)optimal policy. This policy

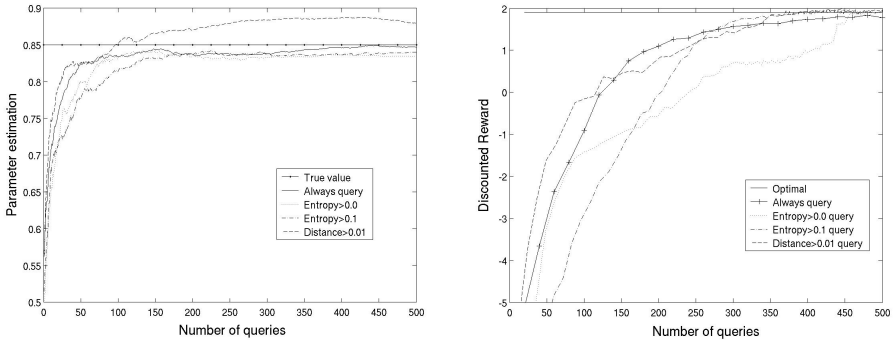

Fig. 2. Experiment 1: Convergence of the estimated parameter $O^{Listen}_{TL,HL}$ (left). Discounted reward as a function of the number of queries (right).

is used to select actions. This allows us to obtain reasonable performance throughout the active learning process: as the sampled models improve, so will the quality of the actions chosen. This also allows the agent to focus the active learning in regions of the state space most often visited by good policies.

We also note that our active learning approach assumes a learning rate, λ. This is used to update the parameters of the Dirichlet distribution over models following each query. In the experiments, we used a fixed learning rate throughout; however this could be varied (e.g., decreasing over time, as is often done in reinforcement learning).

Another important characteristic of our approach is that we need not specify a separate Dirichlet parameter for *each* unknown POMDP parameter. It is often the case that a small number of hyper-parameters suffice to characterize the model uncertainty. For example, noise in the sensors may be highly correlated over all states and therefore we could use a single set of hyper-parameters for all states. In this setup, the corresponding hyper-parameter would be updated whenever action a is taken and observation z is received, regardless of the state.

Finally, while the table below assumes that a query for the state is performed at every time step, this need not be the case. The decision of when to query can be addressed in a decision-theoretic way (as in section 3), but this is intractable when there are many unknown parameters. In the experimental section below, we investigate various heuristics for deciding when to query (or not to query) the state. Another approach (which we have not investigated yet) could be to use queries which do not directly reveal the state, but provide related information, since it is possible to update the parameters of the Dirichlet distribution even without explicit state identification.

4.3 Experimental results on a small domain

To evaluate this approach, we experimented first with the Tiger problem. We considered two cases:

1. The observation probabilities of perceiving {HL, HR} when the Listen action is performed are unknown (randomly initialized).
2. All parameters are unknown (randomly initialized).

In the first experiment, we assume that all parameters are set to their correct value, except the $O^{Listen}_{:,:}$ parameters, which are unknown. (Note that in this experiment the Dirichlet distribution implicitly enforces $O^{Listen}_{s,HL} = 1.0 - O^{Listen}_{s,HR}, \forall s \in S$.)

We test the following heuristics for deciding when to query the state:

- Always query: perform a query at every step
- Entropy>0.0: query when models disagree on which action to select
- Entropy>0.1: query when models disagree substantially on which action to select; i.e., the entropy of the probability distribution over actions, as suggested by the different models, is larger than 0.1.
- Distance>0.01: query when the distance between the beliefs states corresponding to the different models is too large. This distance is defined by: Distance$= \sum_{k=1}^{n} w_k \sum_{i \in S} (b_k(i) - \hat{b}(i))^2$, where $\forall i, \hat{b}(i) = \sum_{k=1}^{n} w_k b_k(i)$

Otherwise, the algorithm is applied exactly as described in Table 1. We also show the return corresponding to the optimal solution obtained when solving the problem with the known parameters.

As shown in Figure 2, the algorithm allows the agent to learn the correct parameters, and we see that performance quickly improves with additional queries. We do not notice a significant difference between the various heuristics for choosing when to perform queries. This suggests that most queries are useful for learning the model.

In the second experiment, all transition and observation probabilities are learned simultaneously. As shown in Figure 3, all correct parameters are learned very accurately with relatively few queries (200-300) and we see that 2000 queries are needed to reach the optimal reward. In comparison, recent results for learning predictive state representations report using on the order of $10^6 - 10^7$ steps to learn the Tiger problem and other problems of similar sizes (Singh et al. 2004). We note, however, that these algorithms are not allowed to query an oracle for additional information, and therefore face a harder problem.

4.4 Experimental results on larger domain

In this section we test the scaling of the algorithm on a larger domain called *Tiger-Grid* (Littman et al. 1995). It has 36 states, 5 actions, 17 observations, for a total of 9000+ transition and observation parameters. It also features probability distributions that are more characteristic of real robots, including noisy motion and sensor conditions. In a domain of this size, it is unlikely that all parameters will be uncorrelated. More likely, there are a few effects (e.g., sensor noise) that are similar over a number of states. Therefore, the uncertainty in these parameters can be correlated through a single hyper-parameter, rather than learning all parameters independently. In the experiments, we apply the algorithm described in Table 1 and vary the number of α parameters used. In the last case, where we use only 8 parameters, we actually lose the ability to represent

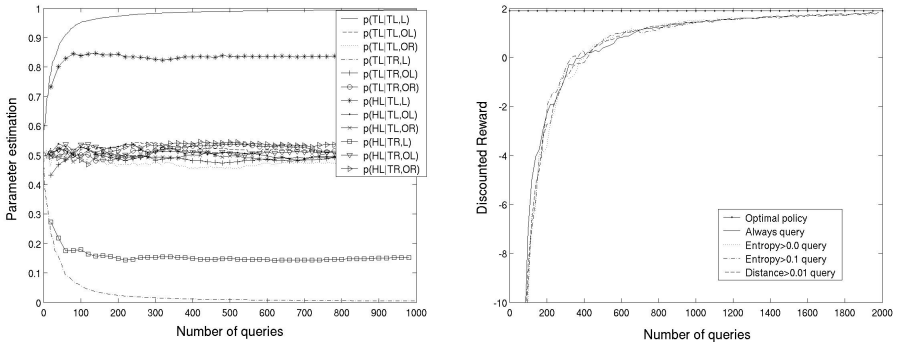

Fig. 3. Experiment 2: Convergence of the parameters for the "always-query" case (left). Discounted reward as a function of the number of queries (right).

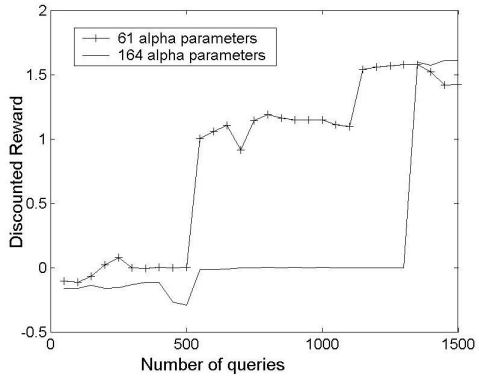

Fig. 4. Experiment 3: Discounted reward as a function of the number of queries for Tiger-grid domain for different numbers of alpha parameters .

the correct model (since we correlate parameters that are not exactly identical). We see on this case that we do not reach a positive reward with such a setting.

The results in Figure 4 confirm that with the appropriate number of parameters, the algorithm can effectively improve the model using queries. As expected, the speed of learning depends on the number of hyper-parameters. Thus, our approach can effectively trade-off learning speed versus model accuracy.

5 Conclusions and future work

In this paper, we studied active learning strategies for POMDPs, a topic that has not yet been addressed in the literature. We presented two alternative algorithms for handling this problem. The first, based on an extended POMDP representation, offers a decision-

theoretic way of handling uncertainty, but it scales poorly when there are many uncertain parameters. The second, called MEDUSA, scales much better but does not offer the same theoretical guarantees. As shown in the empirical results, MEDUSA can handle significant uncertainty with relatively little training data. Furthermore, the correct parameters can be learned even when the priors on the model are uninformative, or in the presence of noise in the oracle. The algorithm offers significant flexibility. We can specify different initial values and confidence levels for the parameters, and when appropriate we can correlate model parameters through the use of the same Dirichlet hyper-parameters. Both of these are effective tools for improving the effectiveness of learning.

Some parts of the algorithm leave something to be desired. At the moment, POMDPs are re-sampled from the Dirichlet distribution at fixed intervals. A better strategy would be to perform as many redraws as possible, and to use all the computational power that is not used by applying the policy in the redrawing subroutine. The question of when to do queries, and whether to consider a more varied set of queries is also of interest. Clearly the decision-theoretic approach of Section 3 can contribute to this decision. However, using this approach calls for POMDP algorithms that can handle very large (possibly continuous) state spaces, and these are currently lacking. Finally, a theoretical analysis of the convergence properties of MEDUSA remains to be done. Our next goal is to apply the ideas proposed in this paper to the control of a mobile interactive robot.

References

Anderson, B. and Moore, A. “Active Learning in HMMs”. In submission. 2005.

Brafman, R. I. and Shani, G. “Resolving perceptual asliasing with noisy sensors”. Neural Information Processing Systems. 2005.

Cohn, D. A., Ghahramani, Z. and Jordan, M. I. “Active Learning with Statistical Models”. Advances in Neural Information Processing Systems.

Dearden, R.,Friedman, N.,Andre, N., ”Model Based Bayesian Exploration”, Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence (UAI), 1999.

Kaelbling, L., Littman, M. and Cassandra, A. ”Planning and Acting in Partially Observable Stochastic Domains” Artificial Intelligence. vol.101. 1998.

Littman, M., Cassandra, A., and Kaelbling,L. ”Learning policies for partially observable environments: Scaling up”, Brown University, 1995.

McCallum, A. K. Reinforcement Learning with Selective Perception and Hidden State. Ph.D. Thesis. University of Rochester. 1996.

Pineau, J., Gordon, G. and Thrun, S. “Point-based value iteration: An anytime algorithm for POMDPs”. IJCAI. 2003.

Poupart, P. and Boutilier, C. “VDCBPI: an Approximate Scalable Algorithm for Large Scale POMDPs”. NIPS 2005.

Singh, S., Littman, M., Jong, N. K., Pardoe, D., and Stone, P. “Learning Predictive State Representations”. Machine Learning: Proceedings of the 2003 International Conference (ICML). 2003.

Spaan, M. T. J. Spaan, and Vlassis, N. “Perseus: randomized point-based value iteration for POMDPs”. Journal of Artificial Intelligence Research. 2005. To appear.