Reinforcement Learning with Limited Reinforcement: Using Bayes Risk for Active Learning in POMDPs

Finale Doshi and Nicholas Roy and Joelle Pineau

Abstract

Partially Observable Markov Decision Processes (POMDPs) have succeeded in many planning domains because they can optimally trade between actions that increase an agent's knowledge and actions that increase an agent's reward. Unfortunately, most real-world POMDPs are defined with a large number of parameters which are difficult to specify from domain knowledge alone.

In this paper, we treat the POMDP model parameters as additional hidden state in a larger "model-uncertainty" POMDP, and develop an approximate algorithm for planning in the induced 'model-uncertainty" POMDP. This approximation, coupled with model-directed queries, allows the planner to actively learn the true underlying POMDP and the accompanying policy. We demonstrate our approach on several POMDP problems.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) have succeeded in many planning domains because they can reason in the face of uncertainty, optimally trading between actions that gather information and actions that achieve a particular goal. This ability has made POMDPs attractive in real-world problems, such as dialog management (Williams & Young 2005), but such problems typically require a large number of parameters that are difficult to specify *a priori* from domain knowledge.

Traditional reinforcement learning approaches (Watkins 1989; Sutton 1988; Strehl, Li, & Littman 2006; Even-Dar, Kakade, & Mansour 2005) to learning in MDP or POMDP domains require an oracle to provide reinforcement feedback after each of the agent's actions during a training period. The feedback requirement may be immaterial if a learning can be performed in simulation, but if learning must occur through interaction with a human expert, the traditional approach may be undesirable. The traditional approach also does not provide robustness guarantees for the agent's performance during the training period. We identify three undesirable properties of the traditional approach that we will address in this work:

1. The time required to gather sufficient training data to learn these parameters in a supervised manner may be prohibitively expensive.
2. Most domains require that the agent experience large penalties (i.e., make critical mistakes) to learn to avoid a poor decision. While these mistakes result in efficient learning, the mistakes also reduce the perception of good performance and reliability.
3. Accurate numerical reward feedback is especially hard to obtain from human users, and determining the reward model without an explicit reinforcement signal (the inverse reinforcement learning problem) poses its own set of challenges (Ng & Russell 2000).

Our objective is to propose a new framework for simultaneous learning and planning in POMDPs that overcomes the above mentioned limitations, allowing us to build agents that behave effectively in domains with inherent model uncertainty.

We discuss how our approach addresses each of these three issues. To address the issue of long training periods, we adopt a Bayesian reinforcement learning approach. By incorporating expert domain knowledge into priors over models, the system begins the learning process as a robust, functional agent while retaining the ability to adapt online to novel situations. This prior can also provide the agent with a basic understanding of potential pitfalls.

To ensure robustness toward catastrophic mistakes, we develop an active learning scheme that alerts the system when additional information is necessary. If the agent deems that the uncertainty in the model may cause it to take an undue risk, it queries an expert regarding what action it should perform. In addition to limiting the amount of training required, these queries allow the agent to infer the potential consequences of an action without executing it. Asking for policy information, instead of a traditional reward signal, also side-steps the issue of getting explicit reward feedback from the user or expert.

We are still left with the inverse reinforcement learning problem, as the user's response regarding correct actions provides only implicit information about the underlying reward function. Bayesian reinforcement learning traditionally has succeeded best with learning observation and transition distributions (Jaulmes, Pineau, & Precup 2005; Poupart *et al.* 2006), where updates have convenient, analytic forms. However, information from policy space (the information usually provided by inverse reinforcement learning) has proven difficult to integrate into algorithms that learn parametric decision-theoretic models. In our work,

we instead use a non-parametric approach to model distributions over possible POMDPs; coupled with a simple action-selection strategy, we show that our approach works well on several standard problems.

Our method retains the decision-theoretic properties of other (PO)MDP learning approaches (Jaulmes, Pineau, & Precup 2005; Poupart *et al.* 2006) and expresses model-uncertainty as additional hidden state in a larger, continuously-valued POMDP. Within this framework, we describe two important practical contributions. First, we propose an approximation algorithm based on minimizing the immediate Bayes risk for choosing actions in a POMDP with uncertain transition and observation probabilities and uncertain rewards. The Bayes risk objective function avoids the computational intractability of solving large, continuously-valued POMDPs; we show that this approximation performs well in a variety of problems. Second, to efficiently gather information about the model without assuming state observability, we introduce the notion of *meta-queries*. By allowing the agent to request information, the meta-queries enable the agent to behave robustly in the face of model uncertainty. The meta-queries accelerate learning and help the agent to infer the consequences of a potential pitfall without experiencing its negative effects. The meta-queries are a powerful way of gaining information, but also make strong assumptions about the environment (namely that the queries will be answered). Fortuantely, there are a number of decision-making problems where this is a reasonnable assumption, in particular in the area of collaborative human-machine tasks (e.g. automated dialogue systems, shared robot control scenarios, etc.)

The remainder of the paper is structured as follows. Sections 2 and 3 provide an overview of the POMDP and our approach for representing uncertainty in the POMDP parameters as a larger model-uncertainty POMDP. In Section 4, we describe our approximation to the policy in this larger model-uncertainty POMDP. We also present lower bounds on the quality of our approximation, although we note that these bounds are not tight and in practice our algorithm far out-performs the bounds we provide. Section 5 contains the results of our approach on several standard POMDP problems. We conclude with a discussion of our approach in the context of prior work in Section 6.

2 The POMDP Model

Formally, a POMDP consists of the n-tuple $\{S,A,O,T,\Omega,R,\gamma\}$. S, A, and O are sets of states, actions, and observations. The transition function $T(s'|s,a)$ is a distribution over the states to which the agent may transition after taking action a from state s. The observation function $\Omega(o|s,a)$ is a distribution over observations o that may be seen in state s after taking action a. The reward function $R(s,a)$ specifies the agent's immediate reward for each state-action pair. The discount factor $\gamma \in [0,1)$ relates the importance of current and future rewards.

In the POMDP model, the agent must choose actions based on past observations; the true state is hidden. The belief, a probability distribution over states, is a sufficient statistic for a history of actions and observations. The belief at time $t+1$ can be computed recursively from the previous belief, b_t, and most recent action a and observation o, by applying Bayes rule:

$$b_{t+1}^{a,o}(s) = \frac{\Omega(o|s',a)\sum_{s\in S} T(s'|s,a)b_t(s)}{\sum_{\sigma\in S}\Omega(o|\sigma,a)\sum_{s\in S}T(\sigma|s,a)b_t(s)} \quad (1)$$

The solution to a POMDP is a policy that maps beliefs to actions. If the goal is to maximize the expected discounted reward, then the optimal policy is given by:

$$V_t(b) = \max_{a\in A} Q_t(b,a), \quad (2)$$

$$Q_t(b,a) = R(b,a) + \gamma \sum_{o\in O} \Omega(o|b,a)V_t(b^{a,o}), \quad (3)$$

where the value function $V(b)$ is the expected discounted reward that an agent will receive if its current belief is b and $Q(b,a)$ is the value of taking action a in belief b. The exact solution to equation 3 is PSPACE-hard, so we use a point-based approximation (Pineau, Gordon, & Thrun 2003).

3 Modeling POMDP Uncertainty

We assume that the sets S, A, and O are fixed. The POMDP learning problem is to determine the parameters in T, Ω, and R that describe the dynamics and objective of the problem domain. A Bayesian approach is attractive in many real-world settings because we may have strong notions regarding certain parameters, but the value of those parameters may be difficult to specify exactly. We place a prior over the model parameters to express our domain knowledge, and improve upon this prior with experience.

Since the state, action, and observation sets are discrete, T and Ω are collections of multinomial distributions. As conjugates to the multinomial distribution, Dirichlet distributions are a natural choice of prior for T and Ω. We use a uniform prior over expert-specified ranges for the reward function R. Together these priors specify a distribution over possible POMDP models. To build a POMDP that incorporates the model parameters into the hidden state, we consider the joint state space $S' = S \times M$, where M is the space of models as described by all valid values for the model parameters. The new state space is continuous and high dimensional, but the transition model for M is simple (assuming the true model is static).

The formulation above makes the agent aware of the uncertainty in the model parameters but does not give it actions to explicitly reduce model uncertainty. To allow for active learning, we augment the action space A of our original POMDP with a set of meta-queries $\{q_m\}$. The meta-queries attempt to confirm the action $a \in A$ that the system thinks is most appropriate. For example, the agent might ask:

> "I think you {may, probably, definitely} want me to do action a_i. Should I do a_i?"

The adverb gives the user a qualitative sense of the agent's uncertainty. If the user answers to the negative, the agent follows up with further questions:

> "In that case, I think I should take action $\{a_j\}$ instead. Is that correct?"

until it receives an affirmative response (the observation space should be augmented with yes/no keywords if not al-

Table 1: POMDP active learning approach.

ACTIVE LEARNING WITH BAYES RISK

- Sample POMDPs from a prior distribution over POMDPs (Section 4.2).
- Interact with the environment:
  - Use the POMDP samples to compute the action with minimal Bayes risk (Section 4.1).
  - If the risk is larger than a given ξ, perform a meta-query (Section 4.1).
  - Update each POMDP sample's belief based on the observation received (Section 4.2).
- Periodically resample from an updated prior over POMDPs (Section 4.2).

Performance and termination bounds are in Sections 4.3 and 4.4.

ready present).[1] We treat the cost ξ of querying the user to be a fixed parameter of the problem.

Meta-queries may be applied to any situation where an expert is available to guide an uncertain agent. Unlike the oracle of Jaulmes, Pineau, & Precup (2005), the meta-queries ask for policy information, not state information. This aspect is important in applications where optimization procedures make the state-space unintuitive to the user, e. g. (Williams & Young 2005). Policy-related questions may be more amenable to deployment in such applications because humans find it natural to give advice.

4 Solution Techniques

Table 1 describes our overall approach to solve and apply the model-uncertainty POMDP. The approach requires two parts. First, given a history of actions and observations, we must describe how to select the next action. Second, we must describe how to perform a belief update in the joint state-model space S', that is, how to update our distribution over model parameters given additional interactions with the environment. In our continuous-valued POMDP, both steps are computationally intractable via standard POMDP solution techniques. We present approximations and bounds for each of these steps. Section 5 contains an empirical evaluation of our approach.

4.1 Bayes-Risk Action Selection

To select actions, we follow the active learning framework for classification (Cohn, Ghahramani, & Jordan 1996). Let the loss $L(a, a^*)$ of taking action a in model m be $Q(b, a) - Q(b, a^*)$, where a^* is the optimal action according to model m. Given a belief $p_M(m)$ over models, the expected loss $E_M[L]$ is exactly the Bayes risk:

$$BR(a) = \int_M (Q(b_m, a) - Q(b_m, a^*_m))p_M(m), \quad (4)$$

where M is the space of models, b_m is the current belief according to model m, and a^*_m is the optimal action for the current belief b_m according to model m. Let $a' = \arg\max_{a \in A} BR(a)$ be the action with the least risk. If our agent is a passive learner using Bayes risk action selection, it will simply perform a'.

The pitfall of always performing the least-risky action a' is that the risk $BR(a')$ may still be quite large, and thus even the best action may incur significant losses. We would like our agent to be sensitive to absolute magnitude of the risks that it takes. Unlike a passive learner, our active learner will perform a meta-query if $BR(a')$ is less than $-\xi$, that is, if the least expected loss is still more than a certain threshold. The series of meta-queries will lead us to choose the correct action and thus accrue no risk.

Intuitively, the Bayes risk criterion selects the currently least risky action, hoping that the uncertainty over models will be resolved at the next time step. Indeed, we can rearrange equation 4 to get:

$$BR(a) = \int_M Q(b_m, a)p_M(m) - \int_M Q(b_m, a^*_m)p_M(m). \quad (5)$$

Since the second term is independent of the choice of action; to maximize $BR(a)$, one may simply maximize the first term:

$$V_{BR} = \max \int_M Q(b_m, a)p_M(m). \quad (6)$$

If we consider the distribution p_M to be a belief over models, the Bayes risk criterion is similar to the Q_{MDP} heuristic (Littman, Cassandra, & Kaelbling 1995), which uses the approximation $V(b) = \max \sum_s Q(s, a)b(s)$ to plan in known POMDPs. In our case, the belief over states $b(s)$ is replaced by a belief over models $p_M(m)$ and the action-value function over states $Q(s, a)$ is replaced by an action-value function over beliefs $Q(b_m, a)$. Recall that the Q_{MDP} approximation is derived by assuming that the uncertainty over states will be resolved after the next time step. Our Bayes-risk criterion may be viewed as similarly assuming that the next action will resolve the agent's uncertainty over models.

Although similar, the Bayes risk action selection criterion does differ from Q_{MDP} in two important ways. First, our actions come from POMDP solutions and thus do fully consider the uncertainty in the POMDP state. Unlike Q_{MDP}, we do not act on the assumption that our state uncertainty will be resolved after taking the next action; our approximation supposes that only the model uncertainty will be resolved. In many practical applications, the model stochasticity is an important factor, and our approach will take actions to reduce state uncertainty. This observation is true regardless of whether the agent is passive (does not ask meta-queries) or active.

In the active learning setting, the second difference is the meta-query. Without the meta-query, while the agent may take actions to resolve state uncertainty, it will never take actions to reduce model uncertainty (since it believes that the model uncertainty will soon disappear). However, the meta-query ensures that the agent rarely (with probability δ) takes a less than ξ-optimal action in expectation. These actions both make the learning process robust from the start and provide the agent with information to resolve uncertainty in the model.

Approximation and bounds: Since the integral in equation 4 is computationally intractable, we approximate it with

[1] In our tests, we used an abbreviated form of the meta-queries for simulation speed.

a sum over a sample of POMDPs from the space of models:

$$BR(a) \approx \sum_i (Q(b_i, a) - Q(b_i, a_i^*)) p_M(m_i) \qquad (7)$$

There are two main sources of approximation that can lead to error in our computation of the Bayes risk; fortunately we can bound the error induced by each.

- Error due to the Monte Carlo approximation of the integral in equation 4: Note that the maximum value of the $Q(b_i, a) - Q(b_i, a_i^*)$ is trivially upper bounded by $\frac{R_{\max} - \min(R_{\min}, \xi)}{1-\gamma}$ and lower bounded by zero. Thus, a standard application of the Hoeffding bound states that a sampling error ϵ_s with confidence δ will require

$$n_m = \frac{(R_{\max} - \min(R_{\min}, \xi))^2}{2(1-\gamma)^2 \epsilon_s^2} \log \frac{1}{\delta} \qquad (8)$$

  samples.[2]

- Error due to the point-based approximation of $Q(b_i, a)$: The difference $Q(b_i, a) - Q(b_i, a_i^*)$ may have an error of up to $\epsilon_{PB} = \frac{2(R_{\max} - R_{\min})\delta_B}{(1-\gamma)^2}$, where δ_B is the sampling density of the belief points. This result is directly from the error bound in (Pineau, Gordon, & Thrun 2003).

Combining these bounds, to obtain confidence δ when calculating if the Bayes risk is greater than $-\xi$, we can set $\epsilon_s = \xi - \epsilon_{PB}$, and compute the appropriate number of samples n from equation 8. We note however that the Hoeffding bounds used to derive this approximation are quite loose; in practice we found that we could often achieve good performance with a set of 15 samples, whereas equation 8 suggests over 800 samples were necessary to achieve that same level of performance.

4.2 Updating the Model Distribution

As described in Section 4.1, we must sample POMDPs from our belief over models to compute the Bayes risk of a particular action. Initially, we have some prior distribution over the model that we can use to sample POMDPs. However, as the agent gains information through interactions with the environment and meta-queries, this distribution should be updated (and the corresponding sample set should change) to reflect our posterior belief over models. While this update can theoretically be performed at any time, we will see that, for episodic tasks, it will make most sense to resample POMDPs at the end of each trial. The posterior must be updated as a result of two sources of information—interactions and meta-queries. While specific interactions (action-observation sequences) allow us to maintain the posterior in closed form, we will also see that the introduction of meta-queries prevents us from representing the posterior in closed form.

The first source of information is a history h of action-observation pairs since the last resampling. To use this information, we will also require the beliefs of the sampled POMDPs at the time of the last resampling. In episodic tasks, keeping track of the initial belief is especially simple, since all sampled POMDPs begin with some task-specific starting belief at the start of each episode. In non-episodic tasks, we may need to store a longer history of actions and observations in order to reconstruct the belief of each POMDP at the time of the last resampling. We will formulate a closed-form update to the posterior given a history h, so aside from the initial belief question, we only need to store action-observation sequences until each resampling.

The second source of information is a record $Q = \{(q, r, h')\}$ of *all* the meta-queries it has asked. Here, q is the query, r is the response, and h' is the history of actions and observations from the start of the episode containing the query to when the query was asked. Unlike in the case of storing histories, we must keep record of all the meta-queries, not just the most recent, because we do not have a closed-form update to the posterior over models that incorporates query information. As before, we note that if all episodes start in the same belief, then we can use h' to "play forward" from some starting belief to the point at which the query was asked. If the episodes start out in different beliefs, then the record set Q must also contain the starting belief for the episode in which the query was asked so we can "play forward" to the point of the query in a similar manner.

Given h and Q, the posterior $p_{M|h,Q}$ over models is:

$$p_{M|h,Q}(m|h, Q) = \eta p(Q|m) p(h|m) p_M(m), \qquad (9)$$

where Q and h are conditionally independent given m because they are both computed from the model parameters. If p_M is a Dirichlet distribution, then $\eta' p(h|m) p_M(m)$ is also a Dirichlet distribution since the likelihood $p(h|m)$ is a product of multinomials. The second likelihood $p(Q|m)$ truncates the Dirichlet distribution and prevents us from having a closed-form expression for $p_{M|h,Q}$. To sample from $p_{M|h,Q}$, we use the updated Dirichlet distribution—which incorporates information from the most recent history h—as our proposal distribution, and then use rejection sampling to discard samples that are inconsistent with our set of queries and responses Q. In this way, we are able to draw samples from the posterior over models.

Action-Observation Histories: Dirichlet Update. Recall that we have placed Dirichlet priors over the observation and transition parameters. These priors may be interpreted as counts; for example, the Dirichlet parameter for the observation probability $\Omega(o|s, a)$ corresponds to the number of times we have seen observation o after performing action a in state s. Updating the prior simply involves adding counts to the Dirichlet parameters corresponding to the transitions (s', s, a) and observations (o, s, a) the agent has experienced during an episode.

Unfortunately, this simple update requires knowing the underlying state for each step in the episode, and our agent only has access to history of actions and observations. We therefore update our parameters using an online extension of the standard EM algorithm. In the E-step, we estimate the distribution over the underlying state for each time step during an episode. In the M-step, we use our distribution over the underlying states to update counts on our Dirichlet prior. The difference between the online EM algorithm and

[2] An error of ϵ with confidence δ means that the probability that the difference between the estimated and true value is greater than ϵ is less than δ. Small values of δ imply that our bound on the error is more likely to hold.

the batch EM algorithm is that we receive additional data—a new history—between iterations. Just as with the standard EM algorithm, the online version will cause the parameters to converge to a local optimum (Sato 1999).

For the E-step, we first must estimate the true state history in order to update our Dirichlet parameters. When computing the distribution over states for some time step, we have two sources of uncertainty: model stochasticity and unknown model parameters. To compute the expectation with respect to model stochasticity, we use the conventional HMM forward-backward algorithm (Rabiner 1989) to obtain a distribution over states at each time-step for each POMDP sample. Next, we combine the distributions for each sample based on the sample's weight. For example, suppose there are n POMDP samples with weights w_i, and at some time-step t, each sample assigns a probability $p_i(s)$ to being in state s. Then the expected probability $\hat{p}(s)$ of being in state s is

$$\hat{p}(s) = \sum_{i}^{n} w_i * p_i(s). \tag{10}$$

Recall that our set of samples represents a continuous distribution over POMDP models, so the summation above is an approximation to an expectation over all models.

Next, we update our Dirichlet counts based on both the probability that a POMDP assigns to a particular state and the probability of that POMDP. Given an action a and observation o corresponding to time t, we would update our Dirichlet count for $\alpha_{o,s,a}$ in the following manner with

$$\alpha_{o,s,a} = \alpha_{o,s,a} + \hat{p}(s) \tag{11}$$

for each state s. Note that this update combines prior knowledge about the parameters—the original value of $\alpha_{o,s,a}$—with new information from the current episode, $\hat{p}(s)$.

While convergence to a local optimum is guaranteed, the global quality of the update procedure will depend on the quality of the estimates $\hat{p}(s)$. In practice, most problems have natural break points such as "goal-reached states" where backtracking in the forward-backward algorithm to determine the prior state sequence becomes more accurate. For example, consider a navigation scenario in a robot grid-world. If the robot is simply lost in the maze, then trying to estimate its position may be inaccurate. However, once the robot reaches the end of the maze, it knows both its start and end position, providing more information for it to recover its position at intermediate time-steps. We update our priors and resample POMDPs at these episode-termination points[3].

Policy-Query Histories: Rejection Sampling. Incorporating information about the action-observation history leads to a closed-form update of our Dirichlet prior, but unfortunately incorporating meta-query information requires a different approach. Each meta-query response provides information about the policy, not the parameters—models are feasible if their policy is consistent with all meta-query responses. Each component of Q is therefore a hard constraint on the set of feasible models, rather than evidence of model likelihood that can be incorporated into the Dirichlet prior. In particular, $p(Q|m)$ is binary: either the model m is consistent with the set of meta-query responses or it is not.

Thus, the true posterior $p_{M|h,Q}$ is a truncated Dirichlet distribution, where $p_{M|h,Q}(m) = 0$ if $p(Q|m) = 0$, otherwise, the model likelihood is given by the Dirichlet distributions over the model parameters. We do not have a closed-form representation for the truncated Dirichlet, but we can evaluate relative likelihoods, which allows us to use sampling strategies such as rejection sampling. In particular, our proposal distribution is the model likelihood given by the Dirichlets computed from the action-observation histories, $p(h|m)$, and our target distribution is the model posterior given in equation 9. The probability of accepting a sampled model is the ratio of the target to the proposal distribution, which (under the assumption of a uniform model prior) is just $p(Q|m)$. When $p(Q|m) = 0$, the model is rejected with probability 1, otherwise the target and proposal distributions are equal and the model is accepted with probability 1.

To sample POMDPs from the true posterior, we therefore first sample POMDPs from the updated Dirichlet priors. Next, we solve for the optimal policy of each model (which can be done much faster than trying to solve the model-uncertainty POMDP, since each sampled POMDP is discrete) and check if each models' policy is consistent with the previous meta-query responses stored in Q. We reject inconsistent samples; the remaining samples are therefore distributed as if they were drawn from the true posterior.

Practical Sampling Considerations. The approach outlined above rejects any POMDP that is inconsistent with any of the previous queries-response pairs. While theoretically sound, we found that it was nearly impossible to sample fully-consistent POMDPs. One reason is that the approximation techniques used to solve the sampled POMDPs introduces significant noise in the solution, especially when dealing with real-time systems. As the number of queries increases, the feasible set of rewards also shrinks and leads to a high rejection rate. We note that we must solve a sampled POMDP to evaluate its consistency with Q, and solving POMDPs is computationally expensive (although still possible in near real-time). The time required to solve a POMDP effectively constrains the total number of POMDPs we can sample before the agent must again be ready to respond to the environment. Thus, high rejection rates can be quite problematic for real applications.

We therefore smooth the rejection sampling probabilities in the following manner to address the problem of noise in the approximate POMDP solutions and use likelihood weights to model this noise in the samples Let k be the number of meta-query responses with which a model m is inconsistent. Instead of rejecting all POMDPs with a non-zero number of inconsistencies, we assign samples a likelihood weight of $w = p(Q|m) = \frac{1}{1+k}u(k' - k)$, where u is the unit step function and k' is a free parameter. This function is essentially an ad-hoc model of the noise in our estimate of the query responses. Samples with a few inconsistent responses receive lower weights but are not rejected completely.[4]

[3] In our simulations we also reset the problem if a maximum number of steps was reached.

[4] We experimented with several violation-tolerant weighting

The question remains of how to set k'. Given a current set of samples, we set the parameter k' with the following heuristic: given our current set of POMDPs, let k^- and k^+ respectively be the minimum and maximum number of violated meta-query constraints in the set. We set $k' = k^+$, thus, all of the current samples in our set have non-zero weight. Then we sample POMDPs until all n POMDPs have $k \leq k^-$ violations (we have a "balanced sample set") or we reach a maximum number of sample attempts. Intuitively, our heuristic attempts to ensure that new samples are at least as good as current samples. Essentially, this approximation assumes that the high-weight samples will dominate in the Bayes risk approximation; we therefore attempt to get a small, representative set of high-weight samples by throwing out POMDPs with low weight.

To further reduce rejection rates when there are a large number of constraints, we focus our sampling away from regions where we have observed greater than k' violations. We do so by occasionally taking a random convex combination of a new sample and a known good POMDP to produce a hopefully better sample. This change means we are trying to draw samples from something closer to the combined prior. Formally, this change would require us to assign weights $w_i/q(m)$ to the samples, where $q(m)$ was the probability of m from this modified proposal distribution. However, since our choice of noise function to assign likelihood weights was already heuristic, we do not make any changes to the weights. While not fully principled, we find that this approach allowed us to apply our algorithm to near real-time applications in practice.

4.3 Performance Bounds

Let V^* be the value of the optimal policy. From our risk criterion, the expected loss at each action is never more than ξ (with confidence δ). However, with probability δ the agent may choose a bad action due to an error in the model estimate, receiving a reward as small as R_{min}. Even worse, this action may put the agent in an absorbing state in which it receives R_{min} forever.

To determine the expected discounted reward over the infinite horizon, consider a Markov chain with two states. The first state is the "normal" state, in which the agent receives a reward of $R - \xi$, where R is the value the agent would have received under the optimal policy. The second state is the "bad" absorbing state, in which the agent receives a reward of $R_{\min}$. The following equation describes the transitions in this simple chain and the values of the states:

$$\begin{vmatrix} V_1 \\ V_2 \end{vmatrix} = \begin{vmatrix} R - \xi \\ R_{\min} \end{vmatrix} + \gamma \begin{vmatrix} 1-\delta & \delta \\ 0 & 1 \end{vmatrix} \begin{vmatrix} V_1 \\ V_2 \end{vmatrix}. \quad (12)$$

Solving the linear set of equations gives us

$$V_1 = \frac{\gamma\delta V_2 + R - \xi}{1 - \gamma(1-\delta)} \quad (13)$$

$$V_2 = \frac{R_{\min}}{1-\gamma}, \quad (14)$$

Finally, the agent's first action puts it in state 1 with probability $1 - \delta$ and state 2 with probability δ. Thus, a **lower bound on the expected value** is:

$$V' = (1-\delta)V_1 + \delta V_2 \quad (15)$$

$$= \eta(V^* - \frac{\xi}{1-\gamma}) + (1-\eta)\frac{R_{min}}{1-\gamma}, \quad (16)$$

where

$$\eta = \frac{(1-\delta)(1-\gamma)}{1-\gamma(1-\delta)}. \quad (17)$$

4.4 Model Convergence

Given the algorithm in Table 1, we would like to know if the learner will eventually stop asking meta-queries. We state that the model is *converged* if $BR(a') > -\xi$ for all histories. Our convergence argument involves two steps. First, let us ignore the reward model and consider only the observation and transition models. As long as standard reinforcement learning conditions—periodic resets to a start state and information about all states (via visits or meta-queries)—hold, the prior will peak around some value (perhaps to a local extremum) in a bounded number of interactions from the properties of the online EM algorithm (Sato 1999). We next argue that once the observation and transition parameters have converged, we can bound the number of meta-queries required for the reward parameters to converge.

Observation and Transition Convergence. To discuss the convergence of the observation and transition distributions, we apply a weaker sufficient condition than the convergence of the EM algorithm. We note that the number of interactions bounds the number of meta-queries, since we ask at most one meta-query for each normal interaction. We also note that the counts on the Dirichlet priors increase monotonically. Once the Dirichlet parameters are sufficiently large, the variance in the sampled models will be small; even if the mean of the Dirichlet distribution shifts with time, no additional meta-queries will be asked.

The specific convergence rate of the active learning will depend heavily upon the problem, which precludes a closed-form expression for the convergence rate. However, we can provide a procedure to determine if r additional interactions are sufficient such that the probability of asking a meta-query is p_q with confidence δ_q. To do so, we will sample random beliefs and test if less than a p_q-proportion have a Bayes risk greater than ξ. For our test to be sufficiently precise, we must consider error due to the belief sampling and our Bayes risk approximation.

1. **Sampling a Sufficient Number of Beliefs.** To test if r interactions leads to a probability p_q of additional meta-queries with confidence δ_q, we compute the Bayes risk for n_b beliefs sampled uniformly. If fewer than $n_q = p_q n_b$ beliefs require meta-queries after r interactions, we accept the value of r. We therefore sample from the posterior Dirichlet given r interactions and estimate $\hat{p}_q = n_q/n_b$. To determine how many beliefs n_b are required to estimate p_q, we apply a Chernoff bound and check if the sampled proportion is within ϵ_q of $p'_q = p_q - \epsilon_q$ with probability δ_q. Using the Chernoff bound $\delta_q = e^{-n_b p'_q \epsilon_q^2/3}$, we set ϵ_q

functions, including e^{-k} and $u(k'-k)$, and found that our function seemed to strike a good balance between not penalizing violations too heavily while still giving sufficiently higher weight to POMDPs with few violations.

to $\frac{2}{3}p_q$ to minimize the samples required to:

$$n_b > \frac{27}{4(p_q)^3} \log \frac{1}{\delta_q}. \quad (18)$$

2. **Computing Bayes Risk from a Conservative Posterior.** We next compute the Bayes risk for each belief given a hypothesized set of r interactions. We do not know *a priori* the response to the interactions, so we use the maximum-entropy Dirichlet posterior to compute the posterior Bayes risk. To compute the maximum-entropy posterior Dirichlet, we note that that each interaction represents a count of some parameter in the model. Given r counts, the max-entropy posterior Dirichlet distribution assigns an equal number of counts to each variable. Thus, we distribute the r counts equally among our Dirichlet parameters. We compute the Bayes risk of each belief from this posterior and accept r if $\hat{p}_q < p_q$.
3. **Correction for Approximate Bayes Risk.** Recall that we approximate the Bayes risk integral with a sum over sampled POMDP models, and the number of models n_m required is given by equation 8. We must correct for the error induced by this approximation. Section 4.1 tells us if a belief b has risk $BR(a) < -\xi$ with confidence δ. Suppose we sample n_b beliefs, and the true fraction of beliefs in which meta-queries are asked is p_q. Due to misclassifications, however, the expected value we will observe is only $(1-\delta)p_q$. We can then apply a second Chernoff bound to determine that with probability δ, no more than $2(1-\delta)n_b$ beliefs will be misclassified.[5] Let

$$p''_q = p_q(1 - 2(1-\delta)), \quad (19)$$

be the minimum fraction of beliefs queries we expect to observe requiring meta-queries if the true fraction is p_q.

Thus, to test if r interactions lead to a probability of p_q additional meta-queries with confidence δ_q, we compute p''_q according to equation 19, sample n_b beliefs uniformly according to equation 18, update the Dirichlet posteriors to be maximum-entropy posteriors, sample the n_m models according to equation 8, and finally compute the posterior Bayes risk for each belief. If less than a p_q-proportion of beliefs require meta-queries, then r is an upper bound on the number of remaining meta-queries with probability p_q and confidence δ_q. If we find that r interactions are not sufficient, we can next test if $r' = 2r$ interactions will be sufficient, et cetera. By testing several values of r, we can determine a bound on the number of meta-queries for the desired values of p_q and δ_q.

Reward Convergence. The cost of a meta-query limits the resolution to which we need to know the rewards. Suppose that we know that a particular POMDP P has an optimal policy π with value V. If we adjusted all the rewards by some small ϵ_r, then the value of the same policy π will differ from V by at most $\frac{\epsilon_r}{1-\gamma}$ (since we will receive at worst ϵ_r less reward at each time step). This value is a lower-bound on the optimal policy in the new POMDP. Thus, a POMDP with all its rewards within $(1-\gamma)\xi$ of P will have a policy of value $V \pm \xi$. In this way, the value ξ imposes a minimal level of discretization over the reward space.

The rewards are bounded between $R_{\min}$ and $R_{\max}$. If our reward space has d dimensions, then our discretization will yield at most $(\frac{R_{\max}-R_{\min}}{(1-\gamma)\xi})^d$ POMDPs. In practice, the discretization simply involves limiting the precision of the sampled rewards. Finally, we note that each meta-query invalidates at least one POMDP sample—otherwise we would not have asked the question. Since there are a finite number of samples, we must eventually stop asking meta-queries.

[5] This bound requires $n_b > \frac{3}{\delta} \log \frac{1}{\delta}$, but we will find that our final bound for n_b is greater than this value.

5 Results

In this section, we first present results in which we solve the model-uncertainty POMDP directly, rather than use the approach outlined in Table 1. This method does not scale, but we can use it to evaluate the utility of meta-queries. We next show results using meta-queries coupled with our Bayes-risk action selection criterion for robust learning of continuous-valued unknown POMDP parameters.

5.1 Learning Discrete Parameters

In domains where model uncertainty is limited to a few, discrete parameters, we may be able to solve for the complete model-uncertainty POMDP using standard POMDP methods We consider a simple POMDP-based dialog management task where the reward in unknown. We presume the correct reward is one of four (discrete) possible levels. Figure 1 compares the performance of the optimal policy *with* meta-queries (left column), an optimal policy *without* meta-queries (middle column), and our Bayes risk policy *with* meta-queries (right column). While the difference in median performance is small, the reduction in variance provided by the meta-queries is substantial. The difference in performance in both median and variance is negligible between the optimal policy and the Bayes risk approximation.

Unfortunately, discretizing the model space does not scale; increasing from 4 to 48 possible reward levels, we could no longer obtain high-quality global solutions using standard techniques. Next, we present results only using our Bayes-risk action selection criterion as an approximation for acting in a continuous-valued model uncertainty POMDP.

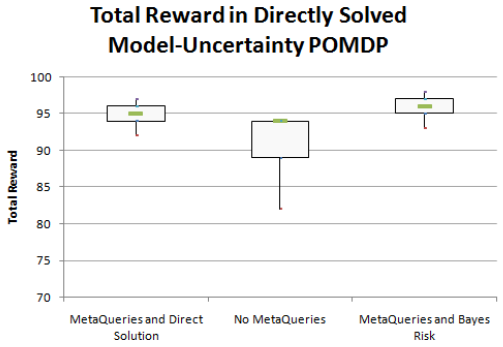

Figure 1: Boxplot of POMDP learning performance with a discrete set of four possible models. Although the medians of the two policies are not so different, the active learner (left) makes fewer mistakes than the passive learner (center). The Bayes risk action selection criterion (right) does not cause the performance to suffer.

Table 2: Difference between optimal and accrued rewards for various problems (smaller = better).

Problem	# States	No Learning	Passive Learning	Active Learning
5x5 Gridworld	26	107.17	111.96	15.46
Tiger-Grid	36	5.83	17.89	0.72
Hallway	57	39.05	39.05	0.85

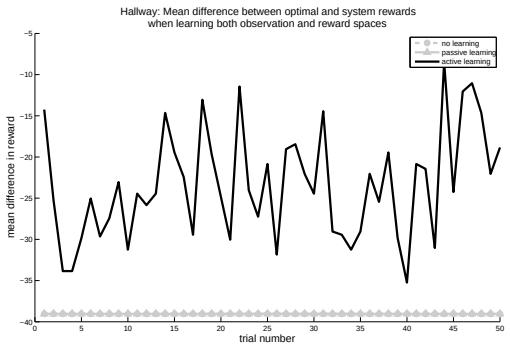

Figure 2: Performance of the non-learner, passive learner, and active learner on the hallway problem. Performance curves for the tiger-grid and gridworld problems were very similar.

5.2 Learning Continuous Parameters

Table 2 shows our approach applied to several POMDP problems[6]. In each case, we used 15 POMDP samples and resampled at the completion of each trial. The non-learner used the 15 samples from the initial prior to make decisions using the Bayes-risk action selection criterion. Its prior did not change based on the action-observation histories that it experienced, nor did it ask any meta-queries to gain additional information. The passive learner resampled its POMDP set after updating its prior over transitions and observations using the forward-backward algorithm. The active learner used both the action-observation histories and meta-queries for learning. None of the systems received explicit reward information, but the active learner used meta-queries to infer information about the reward model.

Figure 2 shows the performance of the non-learner, passive learner, and active learner on the hallway problem (all problems had similar results). In each case, the agent began with observation and transition priors with high variance but peaked toward the correct value (that is, slightly better than uniform). We created these priors by applying a diffusion filter to the ground-truth transition and observation distributions and using the result as our initial Dirichlet parameters. All reward priors were uniform between the minimum and maximum reward values of the ground-truth model. The active learner started (and remained) with good performance because it used meta-queries when initially confused about the model. Thus, its performance was robust from the start.

[6]Tiger-grid and hallway are directly from (Littman, Cassandra, & Kaelbling 1995); the 5x5 gridworld is an extension of the 4x3 gridworld of (Littman, Cassandra, & Kaelbling 1995).

6 Discussion and Conclusion

Prior work in MDP and POMDP learning has also considered sampling approaches to model a distribution over uncertain models. Dearden et. al. (Dearden, Friedman, & Andre 1999) discusses several approaches for representing and updating priors over MDPs using sampling and value function updates. Strens (Strens 2000) shows that in the MDP case, randomly sampling only one model from a prior over models, and using that model to make decisions, is still guaranteed to converge to the optimal policy as long as one resamples the MDP sufficiently frequently from an updated prior over models. However, Strens' approach does not consider risk during the learning process, so the algorithm is not robust to poor initial choices of prior.

One recent approach to MDP model learning, the Beetle algorithm (Poupart *et al.* 2006), converts a discrete MDP into a continuous POMDP with state variables for each MDP parameter. As we saw in section 5.1, however, the computationally-intensive solution techniques required for continuous POMDPs do not scale well enough to handle the entire model as a hidden state in POMDPs. Also, since the MDP is fully observable, Beetle can easily adjust its prior over the MDP parameters as it acquires experience; in our POMDP scenario, we needed to estimate the possible states that the agent had visited.

Another recent approach targeting the problem of Bayesian POMDP learning, is Medusa (Jaulmes, Pineau, & Precup 2005). This approach also captures uncertainty in the model by sampling POMDPs from a prior. Medusa avoids the problem of knowing how to update the prior by occasionally requesting the true state according to various heuristics. Medusa guarantees convergence to the true model, but the learning process may make several mistakes before convergence occurs. Furthermore, a state oracle may be unachievable in many domains; we believe that meta-queries are often a more intuitive form of feedback. Our conservative action-selection policy makes us robust to mistakes.

We developed a new approach for active learning in POMDPs that robustly determines a near-optimal policy. Meta-queries—questions about actions that the agent is thinking of taking—and a risk-averse action selection criterion are proposed, to allow our agent to behave robustly even when its knowledge of the POMDP model is uncertain. We analyze the theoretical properties of our algorithm, but also include several practical approximations that render the method tractable. Finally, we demonstrate the approach on several problems in the POMDP literature. In our future work, we hope to develop more efficient POMDP sampling schemes to allow our approach to be deployed on larger, real-time applications.

References

Cohn, D. A.; Ghahramani, Z.; and Jordan, M. I. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research* 4:129–145.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based bayesian exploration. 150–159.

Even-Dar, E.; Kakade, S. M.; and Mansour, Y. 2005. Reinforcement learning in pomdps without resets. In *IJCAI*, 690–695.

Jaulmes, R.; Pineau, J.; and Precup, D. 2005. Learning in non-stationary partially observable markov decision processes. In *ECML Workshop*.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: scaling up. *ICML*.

Ng, A., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of ICML*.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for pomdps. *IJCAI*.

Poupart, P.; Vlassis, N.; Hoey, J.; and Regan, K. 2006. An analytic solution to discrete bayesian reinforcement learning. In *ICML*, 697–704. New York, NY, USA: ACM Press.

Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Sato, M. 1999. Fast learning of on-line em algorithm. *Technical Report, TR-H-281, ATR Human Information Processing Research Laboratorie*.

Strehl, A. L.; Li, L.; and Littman, M. L. 2006. Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*.

Strens, M. 2000. A bayesian framework for reinforcement learning. In *Proc. of the 17th International Conf. on Machine Learning*.

Sutton, R. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3.

Watkins, C. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, Cambridge University.

Williams, J., and Young, S. 2005. Scaling up pomdps for dialogue management: The "summary pomdp" method. In *Proceedings of the IEEE ASRU Workshop*.