Published in final edited form as:
Proc Int Conf Mach Learn. 2008 ; 301: 256–263. doi:10.1901/jaba.2008.301-256.

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

Finale Doshi,
Massachusetts Institute of Technology, Boston, USA, FINALE@MIT.EDU

Joelle Pineau, and
McGill University, Montreal, Canada, JPINEAU@CS.MCGILL.CA

Nicholas Roy
Massachusetts Institute of Technology, Boston, USA, NICKROY@MIT.EDU

Abstract

Partially Observable Markov Decision Processes (POMDPs) have succeeded in planning domains that require balancing actions that increase an agent's knowledge and actions that increase an agent's reward. Unfortunately, most POMDPs are defined with a large number of parameters which are difficult to specify only from domain knowledge. In this paper, we present an approximation approach that allows us to treat the POMDP model parameters as additional hidden state in a “model-uncertainty” POMDP. Coupled with model-directed queries, our planner actively learns good policies. 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 desired goal. This ability has made POMDPs attractive in real-world problems such as dialog management (Roy et al., 2000), but such problems often require a large number of parameters that are difficult to specify from domain knowledge alone. Recent advances can solve POMDPs with tens of thousands of states (Shani et al., 2007), but learning in POMDPs remains limited to small problems (Jaulmes et al., 2005).

Traditional reinforcement learning approaches (Watkins, 1989; Strehl et al., 2006; Even-Dar et al., 2005) to learning in MDP or POMDP domains require a reinforcement signal to be provided after each of the agent's actions. If learning must occur through interaction with a human expert, the feedback requirement may be undesirable. The traditional approach also does not guarantee the agent's performance during training. We identify and address three limitations in the traditional approach in this work:

1. Gathering sufficient training data for supervised learning may be prohibitively expensive.
2. Most approaches require the agent to experience a large penalty (i.e., make critical mistakes) to discover the consequences of a poor decision.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

3. Accurate numerical reward feedback is especially hard to obtain from people, and inverse reinforcement learning (identifying the reward model without explicit reinforcement) poses its own challenges (Ng & Russell, 2000).

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

We now discuss how our approach will address each of these three issues. To address the issue of long training periods, we adopt a Bayesian reinforcement learning approach and express model-uncertainty as additional hidden state. Bayesian methods (Dearden et al., 1999; Strens, 2000; Poupart et al., 2006; Jaulmes et al., 2005) have received recent attention in reinforcement learning because they allow experts to incorporate domain knowledge into priors over models. Thus, the system begins the learning process as a robust, functional (if conservative) agent while learning to adapt online to novel situations. The domain knowledge specified as a prior can also provide the agent with a basic understanding of potential pitfalls. Our work builds on previous Bayesian reinforcement learning approaches in that we provide both practical approximation schemes as well as guarantees on correctness and convergence.

To ensure robustness toward catastrophic mistakes, we develop an active learning scheme that determines when additional training is needed (typically active learning involves asking for a few labels from unlabeled data; in this work, the ‘label’ corresponds to asking for the optimal action at a particular point in time). If the agent deems that model uncertainty may cause it to take undue risks, it queries an expert regarding what action it should perform. These queries both limit the amount of training required and allow the agent to infer the potential consequences of an action without executing it. Depending on the domain, we can imagine that different forms of information are most readily available. For example, in a navigation task, it may be straight-forward to query a state oracle (i.e., a GPS system) for a location. Similarly, rewards may be easy to measure based on quantities such as energy usage or time to goal. However, in other domains—particularly when working with human-robot interaction and dialog management systems—policy information may be more accurate; a human user may know what he wishes the agent to do, but may be unable to provide the agent with an accurate state representation (which is often complex, for optimization purposes). In these domains, asking for policy information, instead of a traditional reward signal, also side-steps the issue of getting explicit reward feedback from a human user, which can also be inaccurate (Millet, 1998). In this work, we deal exclusively with policy-based queries.

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. To date, Bayesian reinforcement learning has succeeded in learning observation and transition distributions (Jaulmes et al., 2005; Poupart et al., 2006), where updates have closed forms (such as updating Dirichlet counts); previous inverse reinforcement learning work (Ng & Russell, 2000) does not extend to the partially observable case. To overcome this issue, we use a non-parametric approach to model distributions over POMDPs; we demonstrate our approach on several standard problems.

We describe two practical contributions. First, we propose an approximation based on minimizing the immediate Bayes risk for choosing actions when transition, observation, and reward models are uncertain. The Bayes risk criterion avoids the computational intractability of solving large, continuous-valued POMDPs; we show it performs well in a variety of problems. Second, to gather information about the model without assuming state observability, we introduce the notion of *meta-queries*. These meta-queries accelerate learning and help the agent to infer the consequences of a potential pitfall without experiencing its effects. They are a powerful way of gaining information, but they make the strong assumption that they will be answered. Fortunately, a number of decision-making problems exist where this assumption is reasonable, particularly in collaborative human-machine tasks (e.g. automated dialogue systems and shared robot control scenarios).

2. The POMDP Model

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 the agent may transition to after taking action a from state s. The observation function $\Omega(o|s, a)$ is a distribution over observations o that may occur in state s after taking action a. The reward function $R(s, a)$ specifies the immediate reward for each state-action pair. The factor $\gamma \in [0, 1)$ weighs 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 from the previous belief, b_t, the last action a, and observation o, by applying Bayes rule:

$$b_{t+1}^{a,o}(s)=\Omega\left(o|s,a\right)\sum_{s'\in S}T\left(s|s',a\right)b_t\left(s'\right)/Pr\left(o|b,a\right), \tag{1}$$

where $Pr(o|b, a)= \Sigma_{s' \in S}\,\Omega(o|s', a)\,\Sigma_{s \in S}\,T(s'|s, a)b_t(s)$. If the goal is to maximize the expected discounted reward, then the optimal policy is given by:

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

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

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 only tractable for tiny problems, so we use a point-based approximation (Pineau et al., 2003).

3. Modeling POMDP Uncertainty

We assume that the sets S, A, and O are known. The POMDP learning problem is to determine the parameters of 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.

If the state, action, and observation sets are discrete, T and Ω are collections of multinomial distributions. As conjugate priors, 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 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. Although S' is continuous and high dimensional, 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, and by trying various actions, it will be able to reduce uncertainty both in its state and in the parameters. However, the model information provided by the standard actions may be weak, and we would like the agent to be able to explicitly reduce model uncertainty in a safe manner. 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 consult an oracle (e.g., a domain expert) for the optimal action at a particular time step. We assume that the expert has access to the history of actions and observations (as does the agent), as well as the true POMDP model, and thus can advise the agent on the optimal action at any particular time. The agent begins by confirming the action it thinks is best:

> "I think a_i is the best action. Should I do a_i?"

If the oracle answers to the negative, the agent follows with what it thinks is next best:

> "Then I think a_j is best. Is that correct?"

until it receives an affirmative response. The ordered list of actions helps give the expert a sense of the agent's uncertainty; if the agent is uncertain, the expert might advise it to gather information rather than risk an incorrect decision.[1]

Meta-queries may be applied in situations where an expert is available to guide the agent. Unlike the oracle of Jaulmes et al. (2005), the meta-queries ask for policy information, not state information, which can be important if optimization procedures make the state-space unintuitive to the user (e.g., Williams and Young (2005)). In human-robot interaction, it may also simply be more natural to ask "I think you want me to go to the coffee machine. Should I go there?" which may be more natural than "Please enter your most recent statement" or "Please enter our position coordinates."

We can think of these meta-queries simply as additional actions and simply attempt to solve the model-uncertainty POMDP with this augmented action space. However, such an approach quickly becomes intractable. Therefore, we will treat the meta-query as a special action to be taken if the other actions are too risky. We take the cost ξ of querying the user to be a fixed parameter of the problem.

4. Solution Techniques

Table 1 summarizes our two-part approach to solving the model-uncertainty POMDP. First, given a history of actions and observations, the agent must select the next action. Second, the agent must update its distribution over the model parameters given additional interactions with the environment. In the most general case, both steps are intractable via standard POMDP solution techniques.[2]

4.1. Bayes-Risk Action Selection

Let the loss $L_m(a, a^*; b)$ of taking action a in model m be $Q_m^*(b, a) - Q_m^*(b, a^*)$, where a^* is the optimal action in belief b according to model m. Given a belief $p_M(m)$ over models, the expected loss $E_M[L]$ is the Bayes risk:

[1] Our simulations used a shortened meta-query for speed.

[2] Analytic updates are possible if the distributions take certain forms (Poupart & Vlassis, 2008), but even here pruning is needed to keep the solutions to a tractable size.

$$BR(a)=\int_{M}(Q_m^*(b_m, a) - Q_m^*(b_m, a_m^*))p_M(m), \tag{4}$$

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. In the passive learning scenario, our agent just performs a'.

If the risk of the least-risky action a' is large, the agent may still incur significant losses. We would like our agent to be sensitive to the absolute magnitude of the risks that it takes. In the active learning scenario, the agent performs a meta-query if $BR(a')$ is less than $-\xi$, that is, if the least expected loss is more than the cost of the meta-query. The series of meta-queries will lead us to choose the correct action and thus incur no risk.

Intuitively, our criterion selects the least risky action now and hopes that the uncertainty over models will be resolved at the next time step. We can rearrange equation 4 to get:

$$BR(a)=\int_{M} Q(b_m, a)p_M(m) - \int_{M} Q(b_m, a_m^*)p_M(m). \tag{5}$$

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

$$V_{BR} = \max \int_{M} Q(b_m, a)p_M(m). \tag{6}$$

The Bayes risk criterion is similar to the Q_{MDP} heuristic (Littman et al., 1995), which uses the approximation $V(b) = \max \Sigma_s Q(s, a)b(s)$ 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 $Q(b_m, a)$. In the Q_{MDP} heuristic, the agent assumes 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 uncertainty over models.

Though similar, the Bayes risk action selection criterion differs 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. Thus, if the model stochasticity is an important factor, 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. However, meta-queries ensure that the agent rarely (with probability δ) takes a less than ξ-optimal action in expectation. Thus the meta-queries make the learning process robust from the start and allow the agent to resolve model uncertainty.

Approximation and bounds—The integral in equation 4 is computationally intractable, so we approximate it with a sum over a sample of POMDPs from the space of models:

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

There are two main sources of approximation that can lead to error in our computation of the Bayes risk:

- 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. Applying the Hoeffding bound with sampling error ε_S and confidence δ, we will require n_m samples:[3]

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

- 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 $\varepsilon_{PB} = \frac{2(R_{\max} - R_{\min})\delta_B}{(1-\gamma)^2}$, where δ_B is the sampling density of the belief points. This result is directly from the error bound due to Pineau et al. (2003).

To obtain a confidence δ when calculating if the Bayes risk is greater than $-\xi$, we combine these bounds, setting $\varepsilon_S = \xi - \varepsilon_{PB}$, and computing the appropriate number of samples n from equation 8. We note however that the Hoeffding bounds used to derive this approximation are quite loose; for example in the shuttle POMDP problem, we used 200 samples, whereas equation 8 suggested over 3000 samples may have been necessary even with a perfect POMDP solver.

4.2. Updating the Model Distribution

As described in Section 3, we initially placed Dirichlet priors over the transition and observation parameters and uniform priors over the reward parameters. As our agent interacts with the environment, it receives two sources of information to update its prior: a history h of actions and observations and a set of meta-queries (and responses) Q. Given h and Q, the posterior $p_{M|h,Q}$ over models is:

$$p_{M|h,Q}(m|h, Q) \propto p(Q|m)p(h|m)p_M(m), \tag{9}$$

where Q and h are conditionally independent given m because they are both computed from the model parameters. The history h is the sequence of actions and observations since p_M was last updated. The set Q is the set of *all* meta-queries asked (and the expert's responses). Each source poses a different challenge when updating the posterior.

If the agent were to have access to the hidden underlying state, then it would be straightforward to compute $p_{M|h}(m|h) \propto p(h|m)p_M(m)$; we simply need to add counts to the appropriate Dirichlet distributions. However, when the state sequence is unknown, the problem becomes more difficult; the agent must use its belief over the state sequence to update the posterior. Thus, it

[3] An error of ε with confidence δ means $Pr\ [x - \hat{x} > \varepsilon] < \delta$.

is best to perform the update when it is most likely to be accurate. For example, in a robot maze scenario, if the robot is lost, then estimating 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 to recover its path. We focus on episodic tasks in this work and update the belief over models at the completion of a task.

The meta-query information poses a different challenge: the questions provide information about the policy, but our priors are over the model parameters. The meta-queries truncate the original Dirichlet as models inconsistent with meta-query responses have zero likelihood. We approximate the posterior with a particle filter.

4.2.1. During a task: Updating particle weights—Recall that sequential Monte Carlo techniques let us represent a distribution at time t using a set of samples from time $t-1$ using the following procedure (Moral et al., 2002):

$$m_t \sim K(m_{t-1}, m_t), \tag{10}$$

$$w_t = w_{t-1}\frac{p_{M,t}(m_t)}{p_{M,t-1}(m_{t-1})K(m_{t-1}, m_t)}, \tag{11}$$

where $K(m,m')$ is an arbitrary transition kernel and p_M is the probability of the model m under the true posterior.

Sampling a new model requires solving a POMDP, which is computationally expensive and thus may be undesirable while an agent is in the process of completing a task. Thus, we do not change our set of samples during a task (that is, $K(m, m') = \delta_m(m')$ where $\delta()$ is the Dirac delta function). We begin at time $t-1$ with a set of models m_i and weights w_i that represent our current belief over models. If a meta-query occurs at time t, then $p_{M,t}(m) \propto p(Q_t|m)$ $p_{M,t-1}(m)$, and the weight update reduces to

$$w_t = w_{t-1}p(Q_t|m). \tag{12}$$

In theory, the $p(Q|m)$ should be a delta function: either the model m produces a policy that is consistent with the meta-query ($p(Q|m) = 1$), or it does not ($p(Q|m) = 0$). In practice, approximation techniques used to compute the model's policy are imperfect (and expert advice can be incorrect) so we do not want to penalize a model that occasionally acts incorrectly. We model the probability of seeing k incorrect responses in n trials as a binomial variable with parameter p_e, where p_e is the probability a model fails a meta-query due to the approximate solver. This value is hard to characterize, of course, and is problem-specific; we used $p_e = 0.3$ in our tests.

4.2.2. Between tasks: Resampling Particles—Over time, samples taken from the original prior may no longer represent the posterior well. Moreover, if only a few high weight samples remain, the Bayes risk may appear smaller than it really is because most of the samples are in the wrong part of the space. We also need to update the models based on the history information, which we have ignored so far. We do both these steps at the end of a task.

Action-Observation Histories: Dirichlet Update: We first discuss how to update the posterior $p(m|h)$ in closed form. Recall that updating the Dirichlet counts given actions and observations requires knowing the underlying state history, 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 (Sato, 1999). In the E-step, we estimate a distribution over state sequences in the episode. In the M-step, we use this distribution to update counts on our Dirichlet priors. Online EM guarantees convergence to a local optimum.

For the E-step, we first estimate the true state history. Two sources of uncertainty are present: model stochasticity and unknown model parameters. To compute the expectation with respect to model stochasticity, we use the standard forward-backward algorithm to obtain a distribution over states for each sample. Next, we combine the distributions for each sample based on the sample's weight. For example, suppose a sampled model assigns a probability $p_i(s)$ to being in state s. Then the expected probability $\hat{p}(s)$ of being in state s is $\widehat{p}(s)=\sum_i^n w_i p_i(s)$. The samples represent our distribution over models, so this sum approximates 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:

$$\alpha_{o,s,a}=\alpha_{o,s,a}+\widehat{p}(s) \tag{13}$$

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

Resampling Models: As is standard in sequential Monte Carlo techniques, we begin by resampling models according to their weights w_i. Thus, a model with high weight may get selected many times for inclusion in the resampled set of models, while a model with low weight may disappear from the sample set since it is no longer representative of the posterior. Once resampled, each model has equal weight. Before we begin the next task, we perturb the models with the following transition kernel:

- Draw a sample m' from $p_{M|h}$.
- With probability p_k, replace m with m'
- With probability $1 - p_k$, take the convex combination of the model parameters of m and m' so that $m' = p \cdot m' + (1 - p) \cdot m$ with the convexity parameter being chosen uniformly at random on $[0, a]$.

We reduce the probability p_k from 0.9 to 0.4 as the interactions continue, encouraging large exploration earlier on and fine-tuning in later interactions. We set a to 0.2 in our experiments. Based on this sampling procedure, the weight (keeping in mind that after resampling, all models had equal weight) of the transitioned model m' is given by:

$$w_t \propto \frac{p(Q|m')p_{M|h}(m')}{p(Q|m)p_M(m)K(m,m')}, \tag{14}$$

where, $K(m, m') = p_k \cdot p_{M|h}(m')$ if we keep the newly-sampled model and and $K(m, m') = (1 - p_k) \cdot p_{M|h}(m')/a$ if we perturb m via a convex combination.

4.3. Performance Bounds

Let V^* be the value of the optimal policy under the true model. From our risk criterion, the expected loss at each action is no more than ξ. However, with probability δ, in the worst case, the agent may choose a bad action that takes it to an absorbing state in which it receives R_{min} forever.

To determine the expected discounted reward, we consider a two-state Markov chain. In state 1, the "normal" state, the agent receives a reward of $R - \xi$, where R is the value the agent would have received under the optimal policy. In state 2, the agent receives R_{min}. Equation 15 describes the transitions in this simple chain and the values of the states:

$$\left|\begin{array}{c} V_1 \\ V_2 \end{array}\right| = \left|\begin{array}{c} R - \xi \\ R_{\text{min}} \end{array}\right| + \gamma \left|\begin{array}{cc} 1 - \delta & \delta \\ 0 & 1 \end{array}\right| \left|\begin{array}{c} V_1 \\ V_2 \end{array}\right|. \tag{15}$$

Solving this system and noting that the agent begins in state 1 with probability $1 - \delta$ and state 2 with probability δ, the lower bound V' on the expected value is

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

$$\eta = (1-\delta)(1-\gamma)(1-\gamma(1-\delta))^{-1}. \tag{17}$$

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 (where ξ is the cost of a meta-query). 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 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. However, we can check if k 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.

1. Sampling a Sufficient Number of Beliefs: To test if k interactions lead 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 k interactions, we accept the value of k. We sample from the posterior Dirichlet given k interactions and estimate $\hat{p_q} = n_q/n_b$. We desire $\hat{p_q}$ to be within ε_q of $p_q^{\prime}=p_q-\varepsilon_q$ with probability δ_q. Using the Chernoff bound $\delta_q=e^{-n_b p_q^{\prime} \varepsilon_q^2/3}$, we set ε_q to *2/3*p_q to minimize the samples needed:

$$n_b > -27/4 \cdot (p_q)^{-3} \log \delta_q. \tag{18}$$

2. Computing Bayes Risk from a Conservative Posterior: We next compute the Bayes risk for each belief given a hypothesized set of k 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 (that is, assign the k counts to assign an equal number of counts to each variable). We compute the Bayes risk of each belief from this posterior and accept k 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.[4] Let

$$p_q^{\prime\prime}=p_q(1-2(1-\delta)), \tag{19}$$

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 k interactions lead to a probability of p_q for meta-queries with confidence δ_q, we compute $p_q^{\prime\prime}$ from equation 19, sample n_b beliefs uniformly from equation 18, update the Dirichlet posteriors to be maximum-entropy posteriors, sample the n_m models from 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 k is an upper bound on the number of remaining meta-queries with probability p_q and confidence δ_q.

Reward Convergence—The cost of a meta-query limits the reward resolution. Suppose a 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 $\frac{\varepsilon_r}{1-\gamma}$ (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

[4] 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.

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 $\left(\frac{R_{\max} - R_{\min}}{(1-\gamma)\xi}\right)^d$ POMDPs. (Intuitively, the discretization involves limiting the precision of the sampled rewards.) Since each meta-query invalidates at least one POMDP, we must eventually stop asking meta-queries.

5. Results

We first solve a discretized model-uncertainty POMDP solved directly to show the utility of meta-queries. We next couple the meta-queries with our Bayes-risk criterion for learning with with continuous-valued unknown 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 (Doshi & Roy, 2007) where the reward is unknown. We presume the correct reward is one of four (discrete) possible levels and that the meta-query had a fixed associated cost. 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). The difference in median performance is small, but the variance reduction from the meta-queries is substantial.[5]

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 using our Bayes-risk action selection criterion when we no longer discretize the parameter space and instead allow the parameters to take on continuous values within prespecified ranges.

5.2. Learning Continuous Parameters

Table 2 shows our approach applied to several standard POMDP problems (Littman et al., 1995). For each problem, between 50-200 POMDP samples were initially taken from the prior over models. The sampled POMDPs were solved very approximately, using relatively few belief points (500) and only 25 partial backups. The policy oracle used a solution to the true model with many more belief points (1000-5000) and 250 full backups. We took this solution to be the optimal policy. During a trial, which continued until either the task was completed or a maximum number of iterations was reached, the agent had the choice of either taking a normal action or asking a meta-query and then taking the supplied optimal action. POMDPs were resampled at the completion of each trial.

The non-learner (control) always used its initial samples to make decisions, using the Bayes-risk criterion to select an action from the policies of the sampled models. 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

[5] Although the Bayes risk approximation appears higher, the difference in performance in both median and variance is negligible between the optimal policy and the Bayes risk approximation.

systems received explicit reward information, but the active learner used meta-queries to infer information about the reward model. The Hallway problem was too large for the agent to learn (after 50 repetitions, it still queried the oracle at nearly every step); in these results we provided the agent with possible successor states. The smarter prior seemed reasonable as a map and may be easier to obtain for a new environment than to the robot's dynamics. Depending on the problem, tasks required an average of 7 to 32 actions to complete.

Figure 2 shows the performance of the three agents on the shuttle problem (a medium-sized standard POMDP). 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 throughout.

6. Discussion and Conclusion

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. However, their analytic solution does not scale 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. Recently the authors have extended Beetle to partially observable domains (Poupart & Vlassis, 2008), providing similar analytic solutions to the POMDP case. The work outlines efficient approximations but results are not provided.

Prior work in MDP and POMDP learning has also considered sampling to approximate a distribution over uncertain models. Dearden et. al. (1999) discusses several approaches for representing and updating priors over MDPs using sampling and value function updates. Strens (2000) shows that in the MDPs, randomly sampling only one model from a prior over models, and using that model to make decisions, is guaranteed to converge to the optimal policy if one resamples the MDP sufficiently frequently from an updated prior over models. More recently, in the case of POMDPs, Medusa (Jaulmes et al., 2005) avoids the problem of knowing how to update the prior by occasionally requesting the true state based on model-uncertainty heuristics. It converges to the true model but may make several mistakes before convergence. Our risk-based heuristic and policy queries provide correctness and convergence guarantees throughout the learning process.

We developed an approach for active learning in POMDPs that can robustly determine a near-optimal policy. Meta-queries—questions about actions that the agent is thinking of taking—and a risk-averse action selection criterion allowed our agent to behave robustly even with uncertain knowledge of the POMDP model. We analyzed the theoretical properties of our algorithm, but also included several practical approximations that rendered the method tractable. Finally, we demonstrated the approach on several problems from the POMDP literature. In our future work, we hope to develop more efficient POMDP sampling schemes—as well as heuristics for allocating more computation to more promising solutions—to allow our approach to be deployed on larger, real-time applications.

References

Dearden R, Friedman N, Andre D. Model based Bayesian exploration. 1999

Doshi, F.; Roy, N. Technical Report SS-07-07. AAAI Press; 2007. Efficient model learning for dialog management.

Even-Dar E, Kakade SM, Mansour Y. Reinforcement learning in POMDPs without resets. IJCAI. 2005

Jaulmes, R.; Pineau, J.; Precup, D. Learning in non-stationary partially observable Markov decision processes. ECML Workshop on Reinforcement Learning in Non-Stationary Environments; 2005.

Littman ML, Cassandra AR, Kaelbling LP. Learning policies for partially observable environments: scaling up. ICML. 1995

Millet I. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. Journal of Multi-Criteria Decision Analysis 1998:6.

Moral P, Doucet A, Peters G. Sequential Monte Carlo samplers. 2002

Ng A, Russell S. Algorithms for inverse reinforcement learning. ICML. 2000

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

Poupart P, Vlassis N. Model-based Bayesian reinforcement learning in partially observable domains. ISAIM. 2008

Poupart P, Vlassis N, Hoey J, Regan K. An analytic solution to discrete Bayesian reinforcement learning. ICML. 2006

Roy, N.; Pineau, J.; Thrun, S. ACL. Hong Kong: 2000. Spoken dialogue management using probabilistic reasoning.

Sato M. Fast learning of on-line EM algorithm. TR-H-281, ATR Human Information Processing Lab. 1999

Shani G, Brafman R, Shimony S. Forward search value iteration for POMDPs. IJCAI. 2007

Strehl AL, Li L, Littman ML. Incremental model-based learners with formal learning-time guarantees. UAI. 2006

Strens M. A Bayesian framework for reinforcement learning. ICML. 2000

Watkins, C. Doctoral dissertation. Cambridge University; 1989. Learning from delayed rewards.

Williams J, Young S. Scaling up POMDPs for dialogue management: The “summary POMDP” method. Proceedings of the IEEE ASRU Workshop. 2005

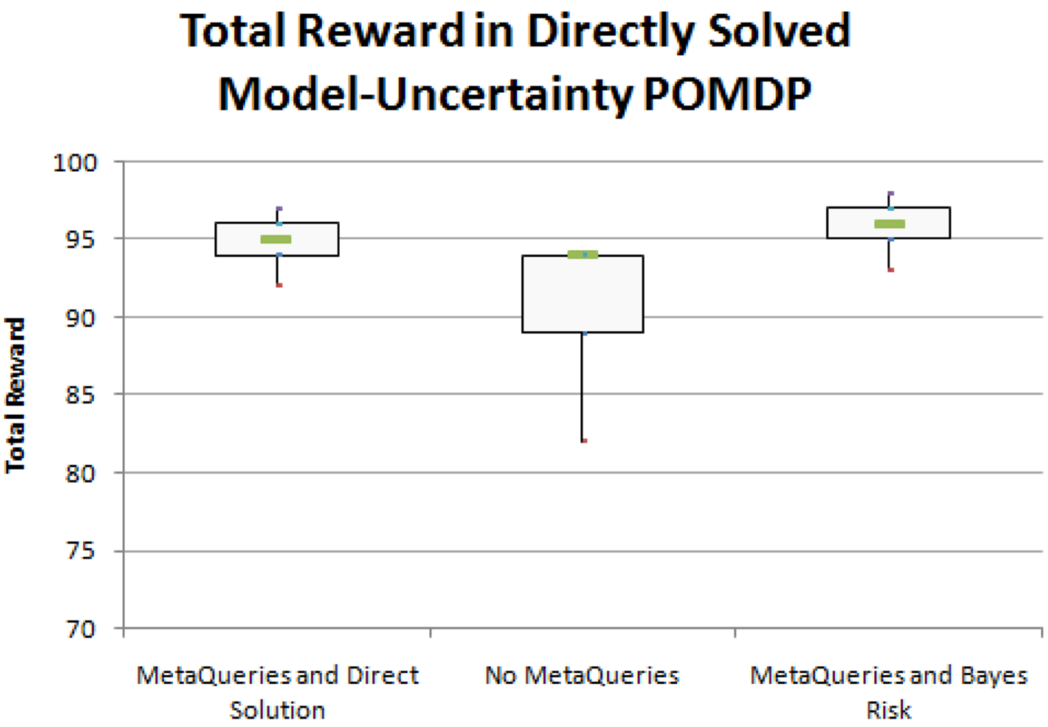

Figure 1.

Boxplot of POMDP learning performance with a discrete set of four possible models. The medians of the policies are comparable, but 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.

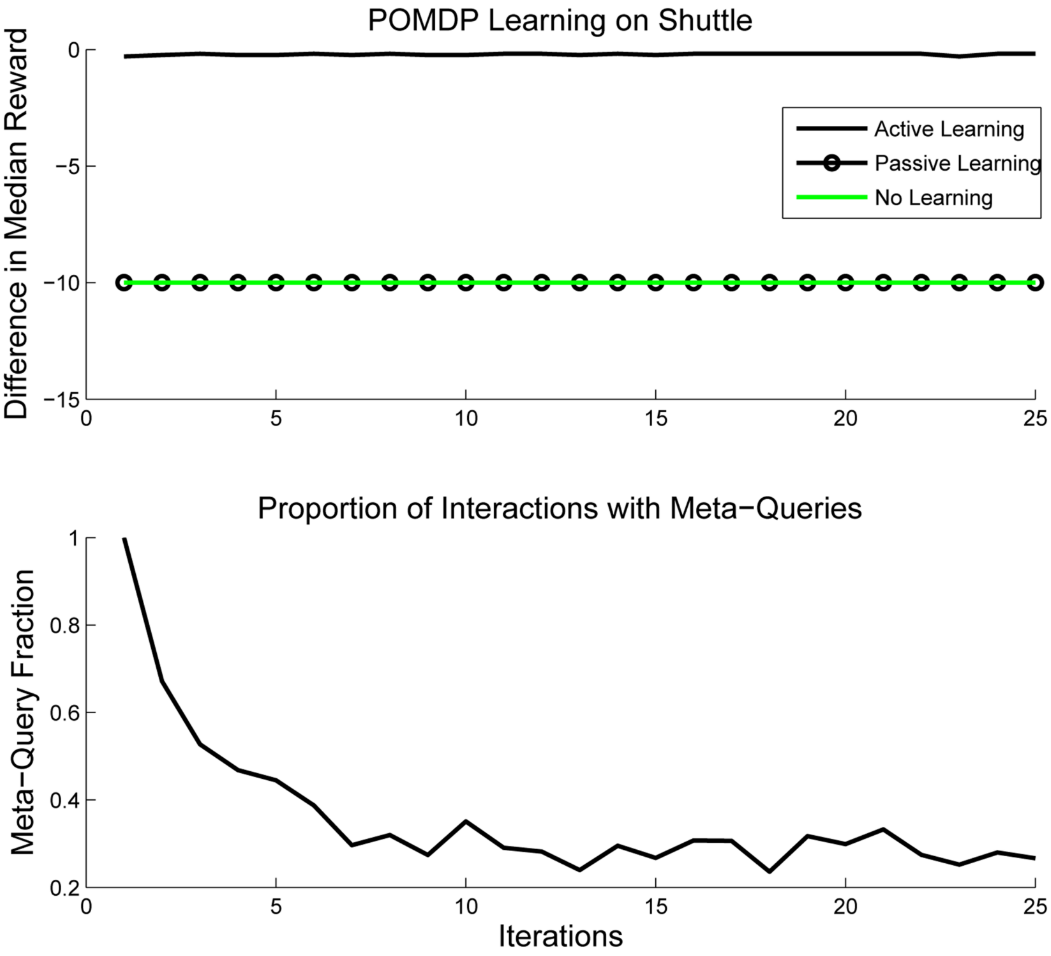

Figure 2.

Performance of the non-learner, passive learner, and active learner on the shuttle problem.

Table 1

POMDP active learning approach.

Active Learning with Bayes Risk

- Sample POMDPs from a prior distribution.
- Complete a task choosing actions based on Bayes risk:
    - 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).
- Once a task is completed, update prior (Section 4.2):
    - Use a kernel incorporating action-observation history to propagate POMDP samples.
    - Weight POMDPs based on meta-query history.

Performance and termination bounds are in 4.3 and 4.4.

Table 2

Mean difference between optimal (under the true model) and accrued rewards (smaller = better).

Problem	States	Control	Passive	Active
Tiger	2	46.5	50.7	33.3
Shuttle	8	10.0	10.0	2.0
Gridworld-5	26	33.1	102	21.4
Hallway	57	1.0	1.0	0.08