Artificial Intelligence 187–188 (2012) 115–132

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs

Finale Doshi-Velez [a,*], Joelle Pineau [b], Nicholas Roy [a]

[a] Massachusetts Institute of Technology, Cambridge, USA
[b] McGill University, Montreal, Canada

ARTICLE INFO

Article history:
Received 16 August 2011
Received in revised form 10 February 2012
Accepted 19 April 2012
Available online 25 April 2012

Keywords:
Partially observable Markov decision process
Reinforcement learning
Bayesian methods

ABSTRACT

Acting in domains where an agent must plan several steps ahead to achieve a goal can be a challenging task, especially if the agent's sensors provide only noisy or partial information. In this setting, Partially Observable Markov Decision Processes (POMDPs) provide a planning framework that optimally trades between actions that contribute to the agent's knowledge and actions that increase the agent's immediate reward. However, the task of specifying the POMDP's parameters is often onerous. In particular, setting the immediate rewards to achieve a desired balance between information-gathering and acting is often not intuitive.

In this work, 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 a POMDP with a multi-dimensional continuous state space; we show it performs well in a variety of problems. We use policy queries—in which we ask an expert for the correct action—to infer the consequences of a potential pitfall without experiencing its effects. More important for human–robot interaction settings, policy queries allow the agent to learn the reward model without the reward values ever being specified.

1. Introduction

The Partially Observable Markov Decision Process (POMDP) is a planning framework that allows an agent to reason in the face of uncertainty, optimally trading between actions that gather information and actions that achieve a desired goal. As a result, POMDPs are attractive for real-world applications in which an agent equipped with noisy and limited sensors must plan several steps ahead to achieve its goal. To date, POMDP-based planners have been used for applications as diverse as robot navigation [1,2], ecological monitoring [3], dynamic pricing [4], and several areas of dialog management and healthcare [5–7]. Recent advances in POMDP approximation algorithms allow agents to plan in POMDPs with tens of thousands of states [8,9].

The POMDP model posits that the world is fully described by some state that is hidden from the agent (but may be affected by the agent's actions). The agent observes the world through sensors that may be noisy—that is, imprecise and inaccurate—or that provide only partial information. For example, a voice-activated dialog system must infer what a user wants (the hidden state) based on audio inputs passed through a microphone and a speech-recognition system. The audio input is likely to have been corrupted by background noise, and the speech-recognition system may not be able to convert the (noisy) audio to text with perfect accuracy. The agent's actions may also have unexpected results: in the case of the

* Corresponding author.
E-mail address: finale@alum.mit.edu (F. Doshi-Velez).

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.04.006

dialog system, a question from the system may cause the user to change his or her wants. Finally, the agent also receives a reward after each action, conditioned on the world state. These rewards encode the agent's goals: a dialog manager might receive a small penalty for each question it asks (for taking up the user's time) and a large reward if it completes the task successfully. The relative values of these rewards will bias the agent toward asking more or less questions as it attempts the task.

Three factors are key for POMDP applications that involve human–machine interaction: learning, accurate reinforcement, and robust behavior. The first factor, learning the POMDP model, is crucial because as the scenarios and the agent's sensors become more complex, the POMDP model requires an increasing number of parameters to describe the problem dynamics and the rewards. Often these parameters are difficult to specify from domain knowledge alone, and gathering sufficient training data to estimate all the parameters may also be prohibitively expensive. Learning online ensures that we refine our knowledge about the model in areas that are most relevant to the task at hand. In this work, we present approximation techniques based on Bayesian reinforcement learning methods to learn larger POMDPs. Bayesian methods [10–13] allow experts to incorporate domain knowledge via priors over models, while letting the agent adapt its belief over models as new observations arrive.

Secondly, learning POMDPs from human–robot interaction (HRI) is particularly challenging because traditional learning approaches [14–16] generally require a reinforcement signal to be provided after each of the agent's actions, but such numerical reinforcement signals from human users are often inaccurate [17]. Inverse reinforcement learning approaches [18] identify a reward model without explicit reinforcement but pose computational challenges. In this work, we note that in HRI domains, policy information may be more accurate than asking for numerical reward: a human user may know what he wishes the agent to do, but may be unable to provide the agent with accurate reward signals. We take an active-learning approach in which the agent asks an expert for the correct action to take if asking for help is likely to refine the agent's knowledge of the underlying POMDP model.

Lastly, asking an expert for the correct policy also addresses our third factor for HRI-oriented POMDP learning: robustness. Most reinforcement learning approaches require the agent to experience a large penalty (that is, make a critical mistake) to discover the consequences of a poor decision. When interacting with humans, a poor decision may undermine the user's confidence in the system and potentially compromise safety. We apply policy queries to allow the agent to act robustly in the face of uncertain models while learning: 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; actions that are recommended by the expect can be assumed to be better than those that are not recommended. Combined with Bayesian reinforcement learning, policy queries allow the agent to learn online, receive accurate reinforcement, and act robustly.

To date, Bayesian reinforcement learning has focused on learning observation and transition distributions [13,12], where updates have closed forms (such as updating Dirichlet counts), and rewards, if initially unknown, are at least observable during learning. The use of a policy query does reflect a more accurate way of extracting information from human users, but it also poses computational challenges. The user's response regarding the correct action places constraints on possible reward values: for example, if a user states that the agent should provide information about the weather when queried about the forecast, the agent can assume that other actions—such as providing information about the TV schedule—would have resulted in lower rewards. However, there are many values of the rewards that could result in the expert's choice being the correct action.

In this work,[1] 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 a POMDP with a multi-dimensional continuous state space; we show our approach performs well on a range of benchmark problems. To gather information about the model without assuming state observability, we use policy queries, which provide information about actions that the agent should take given the current history. Combined with importance sampling techniques to maintain the posterior over possible models, we show that policy queries accelerate learning and help the agent to infer the consequences of a potential pitfall without experiencing its effects.

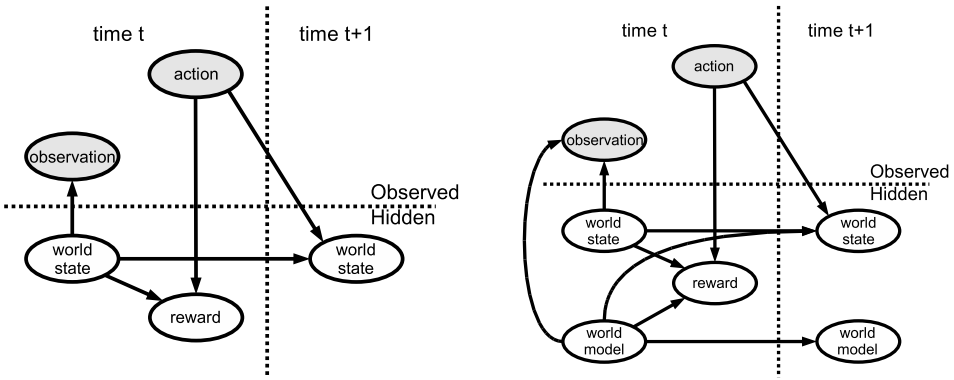
2. The POMDP model

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 [20]. The transition function $T(s, a, s')$ is a distribution over the states to which the agent may transition after taking action a from state s. The observation function $\Omega(s, a, o)$ is a distribution over observations o that may occur in state s after taking action a. The deterministic 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: $\gamma = 0$ implies that only current rewards matter; $\gamma = 1$ implies that current rewards and future rewards are equally valuable. We assume that the sets S, A, and O are discrete and finite.[2]

Assumption 2.1. We assume that R is a discrete and finite set of possible rewards.

[1] This work expands upon work previously published by the authors [19].

[2] There has been recent progress in developing solvers for POMDPs with continuous S, A, and O [21,22]. However, these solvers are not yet as effective as the solvers for discrete models.



(a) Standard POMDP model (b) Model-Uncertainty POMDP

Fig. 1. (a) The standard POMDP model. (b) The extended POMDP model. In both cases, the arrows show which parts of the model are affected by each other from time t to $t+1$.

This assumption makes inference more tractable and is reasonable in many problems where learning the relative scales of the rewards is most of the challenge.

In the POMDP model, the agent must choose actions based on past observations; the true state is hidden. In general, all of the agent's past actions and observations may be needed to determine the optimal decision at the current time. Keeping track of the entire history of actions and observations can become cumbersome, but fortunately the probability distribution over the current true state, known as a belief, is a sufficient statistic for the history. The belief at time $t+1$ can be computed from the previous belief, b_t, the most recent action a, and the most recent observation o, by applying Bayes rule:

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

where $\Pr(o|b_t,a) = \sum_{s' \in S} \Omega(s',a,o) \sum_{s \in S} T(s,a,s') b_t(s)$. If the goal is to maximize the expected discounted reward $E[\sum_{t=1}^{\infty} \gamma^t r_t]$, then the optimal policy is given by the Bellman equation:

$$V^*(b_t) = \max_{a \in A} Q^*(b_t,a), \tag{2}$$

$$Q^*(b_t,a) = R(b_t,a) + \gamma \sum_{o \in O} \Pr(o|b_t,a) V^*\left(b_t^{a,o}\right), \tag{3}$$

where $R(b_t,a) = \sum_s R(s,a) b_t(s)$ and $\Pr(o|b_t,a) = \sum_s \Pr(o|s,a) b_t(s)$. The optimal value function $V^*(b_t)$ 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 and then acting optimally. The exact solution to Eq. (3) is typically only tractable for tiny problems, so we use a point-based solver to approximate the solution.

3. Bayesian model of POMDP uncertainty

We assume that the cardinalities of the sets S, A, O, and R are known. The POMDP learning problem is to sufficiently constrain the distribution of the parameters T, Ω, and R so that the agent may act near-optimally. Specifically, we note that the agent may not need to know the exact model values of T, Ω, and R to act sensibly, as long as all likely models direct the agent to take the correct action. We begin by expressing our domain knowledge through a prior over the model parameters, and improve upon this prior with experience. This Bayesian approach is attractive in many real-world settings because system designers may often have strong notions regarding certain parameters, but the exact values of all the parameters may be unknown.

Let $M = T \times \Omega \times R$ be the space of all valid values of the model parameters, and $m \in M$ be a particular model. To build a POMDP that incorporates the model parameters into the hidden state, we consider the joint state space $S' = S \times M$. Although S' is continuous and high dimensional, the transition model for M is simple if the true model is static. Fig. 1 shows graphical models for the original POMDP and the model-uncertainty POMDP (as in [10,23]). In the standard POMDP (Fig. 1(a)), the world state is hidden, but how the state changes given each action is known. In the model-uncertainty POMDP (Fig. 1(b)), both the world state and the world model are hidden, and we have a different transition dynamic: the transitions of the hidden world state depend on the hidden world model. We assume that the hidden world model is stationary. Thus, the model-uncertainty POMDP is simply a larger POMDP in which the state now consists of two factors, the discrete world state and a world model with continuous parameters. While any POMDP solution technique could now be applied to solve the model-uncertainty POMDP, in this article we derive an approximation that is particularly well-suited to the specific structure of the model-uncertainty POMDP.

Our prior considers a joint belief $p_M(m)$ over T, Ω, and R; that is, we allow the agent's belief over the transitions, observations, and rewards to be coupled. (We use p_M for the belief over models to avoid confusion with $b_m(s)$, the belief over the hidden state given model m.) The model parameters are continuous, and information from observations and policy queries introduces complex dependencies between the parameters (so the belief p_M does not factor). Except for very small or specialized model classes, conjugate models do not exist to update the belief p_M in closed form. We choose to approximate the belief over models as a set of samples:

$$p_M(m) \approx \sum w_i \delta(m_i, m)$$

where w_i represents the weight associated with model m_i and $\delta(m_i, \cdot)$ is a delta-function at model m_i. The collection of samples, each a model, represents the agent's uncertainty about the true POMDP model. The weights w_i and the models m_i are updated via sequential importance sampling [24] as the agent acts in the world. (We describe the belief update in greater detail in Section 5.2.)

While fully general, one difficulty with the sample-based representation is that if the model space M is large, many samples may be required to provide an accurate representation of the belief. If we want to weigh samples based on aspects of their policies—for example, how often they agree with an expert's action choices—each sample will require a significant computational investment because each sample contains a POMDP model to be solved. In Section 5.2.2, we describe how we address this issue through importance sampling.

4. Policy queries

The formulation in Section 3 makes the agent aware of the uncertainty in the model parameters, and by trying various actions, the agent can reduce uncertainty both in its state and in the parameters. However, the model information provided by the actions may be weak, and we would like the agent to be able to reduce model uncertainty in a safe and explicit manner. We augment the action set A of our original POMDP with a set of policy queries $\{\psi\}$ to enable active learning. We refer to the actions in the original action set A as the domain actions.

The policy queries consult an oracle (that is, 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 underlying world dynamics, and thus can advise the agent on the optimal action at any particular time. Unlike the oracle of [13], the policy queries ask for information about the correct action, not the underlying state, which can be important since model-reduction techniques to improve the efficiency of POMDP solvers often make the state space unintuitive to the user (e.g., [25]). For example, in dialog management scenarios, a user may find it more convenient to respond to a question of the form "I think you want me to go to the coffee machine. Should I go there?" rather than "Please enter exactly your most recent statement."

Depending on the form of the interaction, the agent may ask for the policy information through a series of yes–no questions ("I think a_i is the best action. Should I do a_i?") or list ("Here are actions and my confidence in them. Which should I do?"). The key in either case is giving the user a sense of the agent's uncertainty so they can advise the agent appropriately.[3] In an ideal setting, the user would have sufficient information to provide the optimal action for the agent's current belief over world states.

We can think of these policy queries ψ simply as additional actions and attempt to solve the model-uncertainty POMDP with this augmented action space. However, solving the model-uncertainty POMDP without policy queries is already intractable; the additional complexity of directly incorporating policy queries into the action space only makes solving the model-uncertainty POMDP more challenging. In the next section, we introduce an approximation for solving the model-uncertainty POMDP that treats a policy query as a special action to be taken only if other actions are too risky. Specifically, our approximation assigns a fixed cost $-\xi$ for each policy query and only ask a policy query if the expected loss of all other actions exceeds $-\xi$.

5. Solution approach

As with all POMDPs, acting optimally in a model-uncertainty POMDP requires two capabilities. First, given a history of actions and observations, the agent must be able to select the next action. Second, the agent must be able to update its distribution over the model parameters given the selected action and the subsequent observation from the environment. In the most general case, both steps are intractable via standard POMDP solution techniques. (Analytic updates are possible if the distributions take certain forms [26], but even in this case function approximation is needed to keep the solutions to a tractable size.)

We assume that our problem domains consist of episodic tasks, that is, that they are periodically restarted. An episode may end when the agent reaches a goal, or, if the task has no explicit goal, once the history reaches a certain length. Table 1

[3] If the expert is another agent, the format of these queries would, of course, be adapted to whatever input representation is required for the expert agent to provide a near-optimal action.

summarizes our approach and outlines the following sections. Sections 5.1 and 5.2 focus on the aspects of the POMDP-solving process that are unique to the model-uncertainty POMDP: we describe how actions are selected given a belief over models in Section 5.1 and how to update the belief over models in Section 5.2. Each of these steps requires approximations, and we provide bounds on the approximation error in Sections 6.1 and 6.2. For planning and belief monitoring in standard discrete POMDPs, we refer the reader to classic work such as [27].

5.1. Bayes-risk action selection

Given a belief $p_M(m)$ over models,[4] each with a belief $b_m(s)$ over states, the optimal—or highest-valued—action in the model-uncertainty POMDP can be found through an application of the Bellman equation from Section 2:

$$V^*\big(p_M, \{b_m\}\big) = \max_{a \in A} Q^*\big(p_M, \{b_m\}, a\big), \tag{4}$$

$$Q^*\big(p_M, \{b_m\}, a\big) = \int_M p_M(m) R\big(b_m(s), a\big) \tag{5}$$

$$+ \int_M p_M(m) \gamma \sum_{o \in O} Pr\big(o | b_m(s), a, m\big) V^*\big(p_M^{a,o}, \{b_m^{a,o}\}\big), \tag{6}$$

where we use $\{b_m\}$ to represent the set of beliefs $b_m(s)$ for all $m \in M$. In general, the integral over model space is intractable; even new online planning methods [28] cannot achieve the search depths necessary for planning in the model-uncertainty POMDP. In this section we describe how the agent can select actions robustly without solving the intractable model-uncertainty POMDP.

Specifically, let the model-specific loss $L_m(a, a^*; b_m)$ of taking action a in model m be $Q_m^*(b_m, a) - Q_m^*(b_m, a^*)$, where a^* is the optimal action in model m given belief b over the hidden states. Augmented with the belief $p_M(m)$ over models, the expected loss $E_M[L]$ is the Bayes risk:

$$BR\big(a; \{b_m\}\big) = \int_M \big(Q_m^*(b_m, a) - Q_m^*\big(b_m, a_m^*\big)\big) p_M(m), \tag{7}$$

where M is the space of models, b_m is the belief over states according to model m, and a_m^* is the optimal action in belief b_m according to model's value function Q_m^*. The first term within the integral, $(Q_m^*(b_m, a) - Q_m^*(b_m, a_m^*))$, is the loss incurred by taking action a instead of the optimal action a_m^* with respect to model m. In general, different models may have different optimal actions a_m^*; the Bayes risk is the expected loss after averaging over models. Let $a' = \arg\max_{a \in A} BR(a)$ be the action with the least risk.

The action a' represents the least risky action from our domain actions. However, if the risk of a' is large, the agent may still incur significant losses; the agent should be sensitive to the absolute magnitude of the risks it takes. Therefore we consider both performing a'—the best option from the domain actions—and using a policy query ψ to determine the optimal action. We assume that the cost $-\xi$ of a policy query is a constant fixed across all models (with value based on the cost of getting policy information in the given domain). In the active learning scenario, the agent performs a policy query if $BR(a')$ is less than $-\xi$, that is, if the least expected loss is more than the cost of the policy query. The policy query will lead the agent to choose the correct action and thus incur no risk. In Section 7, we empirically compare the performance of an agent that uses active learning to a passive agent that always selects the least risky action.

5.1.1. Relationship to the Q_{MDP} heuristic

The Bayes-risk criterion selects the least risky action now and assumes that the uncertainty over models will be resolved at the next time step. To see this interpretation, we first rearrange Eq. (7) to get:

$$BR(a) = \int_M Q^*(b_m, a) p_M(m) - \int_M Q^*\big(b_m, a_m^*\big) p_M(m). \tag{8}$$

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

$$V_{BR} = \max_a \int_M Q(b_m, a) p_M(m). \tag{9}$$

As expressed by Eq. (9), the Bayes-risk criterion is similar to the Q_{MDP} heuristic [29], 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

[4] At any point in time, the belief over models is given with respect to the history of domain actions, observations and the policy queries. For clarity, in this section, we consider choosing actions given some arbitrary belief $p_M(m)$ over models.

$p_M(m)$, and the action-value function over states $Q(s,a)$ is replaced by an action-value function over beliefs $Q(b_m,a)$. The Q_{MDP} heuristic encodes the assumption that the uncertainty over states will be resolved after the next time step. The Bayes-risk criterion may be viewed as similar to Q_{MDP} but assumes that the next action will resolve the uncertainty over models rather than states.

Despite this similarity, the Bayes-risk action selection criterion differs from Q_{MDP} in two important ways. First, our actions come from POMDP solutions and thus we do fully consider the uncertainty in the POMDP state. Therefore, our approach will take actions to reduce state uncertainty: 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. The agent will take actions to reduce state uncertainty regardless of whether the agent is passive (that is, it does not ask policy queries ψ) or active.

The second difference is explicitly in the active learning setting, when the agent *does* use policy queries ψ. Without policy queries, the agent may take actions to resolve state uncertainty, but it will never deliberately take domain actions to reduce model uncertainty. However, policy queries allow the agent to decide to reduce model uncertainty by helping it prune away unlikely models. Policy queries also ensure that the agent rarely (with probability δ) takes a less than ξ-optimal action in expectation. Thus the policy queries make the learning process robust from the start and allow the agent to resolve model uncertainty.[5]

5.2. Updating the model distribution

The second part of solving the model-uncertainty POMDP is updating the agent's belief over models once it has received an observation. Specifically, we need to track the belief $p_M(m|h,\Psi)$, where h is the history of domain actions and observations and Ψ is the set of history-policy query response pairs. We accomplish this task using importance sampling, with a proposal distribution $p_M(m|h)$ that ignores the expert-provided information, to more efficiently draw samples from the much more complex distribution $p_M(m|h,\Psi)$.

In this section, we first outline the various factors involved in the belief update for the model-uncertainty POMDP. We next provide a quick review of importance sampling and discuss the challenges particular to the model-uncertainty POMDP before detailing our approach.

5.2.1. Updates to the model distribution from domain history

As our agent interacts with the environment, it receives two sources of information to update its prior: a history h of domain actions and observations and a history of policy queries (and responses) Ψ. Given h and Ψ, the posterior $p_{M|h,\Psi}$ over models is:

$$p_{M|h,\Psi}(m|h,\Psi) \propto p(\Psi|m)p(h|m)p_M(m), \tag{10}$$

where Ψ and h are conditionally independent given m because they are both computed from the model parameters. The history h is the sequence of domain actions and observations since p_M was last updated. The set $\Psi = \{(h_i, a_i^*)\}$ consists of *all* pairs of histories h_i and the policy query responses a_i^* collected by the agent. Each of these information sources, h and Ψ, poses a different challenge when updating the posterior.

We place Dirichlet priors over the transition, observation, and reward distributions. If the agent were to have access to the hidden underlying state, then it would be straightforward to compute $p_{M|h}(m|h) \propto p(h|m)p_M(m)$: the updates to each part of the model would factor, and each update could be computed in closed form using Dirichlet-multinomial conjugacy. 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.

5.2.2. Updates to the model distribution from policy queries

Incorporating history information is a standard step in any POMDP belief update. The policy query information poses a unique challenge for updating the model-uncertainty POMDP belief: the questions provide information about the policy, but our priors are over the model parameters. Any particular model, when solved, will produce a policy that may or may not be consistent with a policy query. If the oracle answering the policy queries was perfect, any model producing a policy inconsistent with a policy query would have zero likelihood: either the model m produces a policy that is consistent with the policy query ($p(\Psi|m) = 1$), or it does not ($p(\Psi|m) = 0$). In practice, such hard constraints make the inference difficult by quickly rendering most models infeasible (e.g., Fig. 2); it also may be reasonable to assume that the user or expert will occasionally make mistakes. We soften this hard constraint by modeling the probability of seeing k consistent responses in $n_{\texttt{total}}$ trials as a binomial variable with parameter p_e, such that $p(\psi|m) = p_e$ if m is consistent with the policy query ψ and $p(\psi|m) = 1 - p_e$ otherwise. Whether or not the oracle is assumed to be perfect, policy queries make the inference more challenging because a policy is typically derived from a complex, nonlinear combination of the model's

[5] An interesting area for future work might be to compute tighter risk values using different underlying approximations, such as the fast informed bound [30].

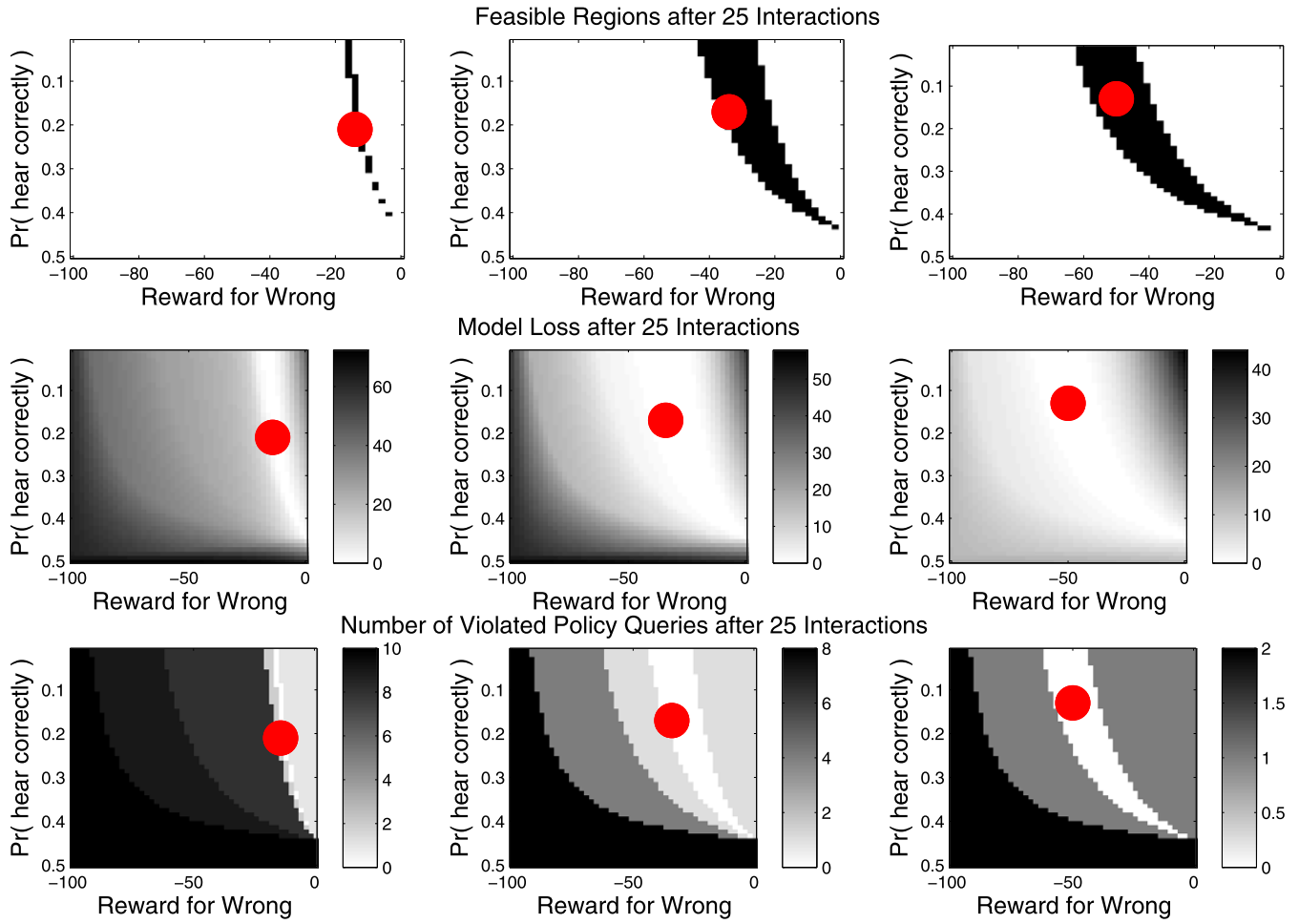


Fig. 2. The first row plots feasible regions after 25 interactions for three different TIGER trials: the black area represents the space of feasible models (as constrained by policy-queries), and the red dot shows the true model. Depending on how the interactions proceeded and how many policy queries were asked, we see that the feasible region can quickly become quite small. The second and third rows show the incurred loss and number of inconsistent policy query responses, respectively.

transition, observation, and reward parameters. Thus, a constraint on the policy introduces dependencies between the model parameters, and the transition, observation, and reward distributions can no longer be considered independently.

Recall from Section 3 that we maintain a distribution over parameterized models, so we represent the posterior over models by a set of samples. To update the posterior, we must therefore apply the belief update (Eq. (1)) to our sample-based representation of the belief. The most general case, importance sampling [24], dictates that for samples m from some proposal distribution $K(m)$ to be an unbiased approximation of the desired posterior $p_{M|h,\Psi}$, the samples must be weighted by

$$w(m) = \frac{p_{M|h,\Psi}}{K(m)}. \tag{11}$$

It is straightforward to derive the sequential weight update for a set of samples at time t with weights w_t:

$$w_{t+1}(m) = w_t(m)\frac{p_{M,t+1}(m)}{p_{M,t}(m)} \tag{12}$$

where p_M is the probability of the model m under the true posterior.

With only mild conditions on $K(m)$—it must have support wherever $p_{M|h,\Psi}$ has support—the weighted samples are an unbiased approximation of the true posterior. However, a good choice of proposal will reduce the variance in the approximation, which is important when relatively few samples are being used. Indeed, because the model-uncertainty space is so large, and samples—models—are relatively expensive, a good proposal distribution is critical to our task [24]. In particular, a single policy query can render a large part of the model space infeasible in a single time-step. After some interactions with the environment, only a small part of that space may have significant probability mass. If the agent's belief consists of only a few samples, then it is possible that a few policy queries may make all of the samples infeasible.

5.2.3. Example of model inference

For illustration, we show a toy example using the canonical TIGER problem [29]. In the TIGER problem, a tiger hides behind one of two doors. At each time step, the agent may try to open a door or listen for the tiger; the agent receives a large reward for opening the door without the tiger behind it. The TIGER problem has several parameters, but for the purposes of this discussion, we assume that only two are unknown, the penalty for opening the door with the tiger and the probability of hearing the tiger correctly, so the model space M can be represented in two dimensions.

Fig. 2 shows various aspects of the distribution space after the agent has tried opening the door 25 times on 3 independent trials (in red is the true model). The top row shows the size of the feasible region, that is, the region of model space consistent with the policy queries that have been asked. Each of the 3 trials have slightly different feasible regions because different policy queries have been asked on each run, and we see that oftentimes the regions can be quite small.

The small regions produced by this binary threshold—whether the model is consistent with all policy queries—makes searching for models with non-zero probability difficult. The second two rows of Fig. 2 show ways of softening the hard constraint. For each model m, the second row shows the total loss $\sum_h Q^*_m(b_h, a^*_{m^*}) - Q^*_m(b_h, a^*_m)$ for taking the action a^*_m that is optimal under m instead of the action $a^*_{m^*}$ that is optimal under the true model m^*, summed over the histories associated with each policy query. The loss function penalizes models for choosing actions that result in lower values, rather than simply choosing actions different from the expert. The smooth gradients in the second row give a sense of which models may still have good performance although they are not fully consistent with the policy queries. Although the Q^* functions are smooth (and thus possibly well-suited for gradient-based optimizations), computing them is intractable in all but the smallest domains—such as the restricted TIGER domain below. In contrast, we see that the third row, which shows for how many policy queries each model m is inconsistent, is very non-smooth and thus difficult to optimize. As in the first row, a small set of models in the third row are consistent with all of the policy queries. However, among the models that provide inconsistent responses for at least one policy, looking at the number of inconsistent responses a model gives can help us distinguish models closer to the true model.

We can define the "feasible" region as the region of the model space where the corresponding policies have little to no disagreement with the policy query responses (the light gray or white regions of Fig. 2). Then, in this low-dimensional case, one might imagine using non-uniform grids to iteratively refine the feasible region. However, as the dimensionality of the model space increases (recall that each POMDP parameter is a dimension), grid-based techniques quickly become intractable. The case of the TIGER problem illustrates that for our belief update to be computationally robust, two factors are particularly important. First, as policy queries can drastically affect the likelihood of large parts of the model space, a belief update that makes only local adjustments to samples—such as gradient ascent—is unlikely to be successful.[6] Second, we see that a hard feasibility constraint can make finding probable models extremely difficult, but quantities such as a model's loss or inconsistent responses provide smoother measures of a model's quality. Softening the feasibility constraint not only simplifies the problem computationally, but it also corresponds to the belief that the policy oracle and the POMDP solver are occasionally fallible—both true in real-world situations.

5.2.4. Resampling models between tasks

Eq. (11) describes how samples from a proposal distribution $K(m)$ can be used to provide an unbiased representation for our desired distribution $p_{M|h,\Psi}$. We call this step the resampling step because we are drawing new models (instead of adjusting weights on fixed models). Resampling models is important because, over time, even a re-weighted set of samples 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. In this work, we resample models at regular experience intervals (e.g., every 100 interactions with the environment) as well as if the model weights fall below a threshold (suggesting that none of the models are probable).

We define $K(m)$ as the posterior over models given only the histories h (not the policy queries Ψ), that is, $K(m) = p_M(m|h)$, noting that for a Dirichlet-multinomial prior, this distribution will have full support over the model space. We use a Markov chain Monte Carlo approach, forward-filtering backward-sampling (FFBS) [31], to sample dynamics models from this proposal distribution $K(m)$. FFBS alternates between sampling potential state sequences and potential observation and transition matrices. To sample the state sequence, FFBS first computes the standard forward pass of the Viterbi algorithm [32]:

$$\alpha_{t+1}(s) = \Omega(s, a_t, o_t) \sum_{s'} \alpha_t(s') T(s, a_t, s')$$

where $\alpha_0(s)$ is initialized to the initial belief state of the POMDP. However, instead of computing the marginal distributions of each state on the backward pass, FFBS samples a possible state sequence. The final state s_T is sampled from $\alpha_T(s)$. Starting with s_{T-1}, the remaining states are sampled according to

$$s_t \sim \alpha_t(s) T(s, a_t, s_{t+1}) \Omega(s_{t+1}, a_t, o_t).$$

[6] In the limit of a large number of samples, of course, feasible regions can always be found—even if they are small.

Given a sequence of visited states, the observation and transition matrices are easily sampled if we use a Dirichlet prior for each observation distribution $\Omega(s,a,\cdot)$ and transition distribution $T(s,a,\cdot)$. For example, given a Dirichlet prior with parameters $\{c_1, c_2, \ldots, c_{|O|}\}$ and counts $\{n_{o_1}, n_{o_2}, \ldots, n_{o_{|O|}}\}$ of how often each observation is observed following state s and action a in our sampled sequence, the observation distribution $\Omega(s,a,\cdot)$ is distributed as a Dirichlet with parameters $\{c_1 + n_{o_1}, c_2 + n_{o_2}, \ldots, c_{|O|} + n_{o_{|O|}}\}$. By repeating the process of resampling state sequences and resampling the transition and observation matrices several times, FFBS draws samples from the true posterior over observation and transition matrices given the history. Each sampled model has equal weight.

The reward model remains to be sampled. Assuming that the rewards are discrete—that is, we have a set of possible reward values specified beforehand—we can place a Dirichlet prior over reward models; "evidence" can be added to the prior in the form of counts.[7] Starting from the prior, we apply a proposal distribution of the following form:

- With probability p_k, use the prior over reward models as the base distribution.
- With probability $1 - p_k$, use a base distribution that adds the weighted mean reward model (from the previous samples) as a set of counts to the prior.
- Randomly perturb the base distribution with counts chosen from $[-d, d]$.
- Sample a reward model from the perturbed distribution

where p_k is the proportion of policy queries with which the best sampled model is inconsistent (so as the models improve, we are more likely to trust their reward models as good starting points for sampling). For our experiments, we set d to 0.5.

Finally, once we have sampled a model m from this proposal distribution $K(m)$, we must assign it a weight. Our choice of $K(m)$ results in a particularly simple form of the policy-query set's likelihood as the weight

$$w(m) = \frac{p_{M|h,\Psi}(m|h,\Psi)}{p_{M|h}(m|h)} \propto p(\Psi|m,h) \tag{13}$$

where the likelihood $p(\Psi|m,h)$ can be computed by solving each sampled model m, playing forward the history leading to each policy query, and computing the action the POMDP model would have suggested. Specifically, 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 policy query due to the approximate solver. Assigning the model the appropriate importance sampling weight corrects the fact that the rewards were not chosen from the true posterior and ensures that the set of models m is an unbiased representation of the true posterior.

5.2.5. Updating weights during a task

Resampling models is a computationally-intensive process; therefore we only resample models after the agent has gained a substantial, new chunk of experience or if none of its current models have high weights. Between these resampling periods, we update the weights on the models so that they continue to represent the current belief over models. If a policy query ψ occurs at time t, then $p_{M,t}(m) \propto p(\psi|m)p_{M,t-1}(m)$, and the weight update reduces to

$$w_t(m) = w_{t-1}(m)p(\psi|m) \tag{14}$$

where $p(\psi|m)$ is p_e if model m is consistent with the policy query ψ and $1 - p_e$ otherwise.

6. Performance and policy query bounds

6.1. Performance bounds

The procedure in Section 5.1 defines how an agent can act in the model-uncertainty POMDP using the Bayes-risk action selection criterion. A reasonable question is how well we can expect this procedure to perform compared to an agent that always queries the expert (and therefore acts optimally). We now show formally in what sense the agent's behavior is robust. Let V^* be the value of the optimal policy under the true (unknown) model. We first show that we can ensure that the expected loss at each action is no more than ξ with high probability. Next, we show how bounding the expected loss gives us a bound on overall performance.

6.1.1. Expected loss for a single action

Recall from Section 3 that the agent represents its belief over POMDP models through a set of samples. Thus, the integral in Eq. (7) reduces to a sum over the agent's sample of POMDPs from the space of models:

$$\hat{BR}(a) \approx \sum_i \left(\hat{Q}^*_{m_i}(b_i,a) - \hat{Q}^*_{m_i}\left(b_i,a^*_i\right)\right)p_M(m_i). \tag{15}$$

[7] Continuous rewards could be modeled in the same framework using, for example, Beta priors, but inference would be much more challenging as the prior and the likelihood would no longer be conjugate.

The Bayes risk value $\hat{BR}(a)$ is an approximation to the true Bayes risk $BR(a)$ for two reasons. First, we are approximating the integral in Eq. (7) with a set of samples. Second, we are approximating the true action-value function Q_{m_i} with a point-based approximation $\hat{Q}_{m_i}$. We bound the error due to each of these approximations to derive an overall bound for the error in $\hat{BR}$.

We first consider the error due using an approximate action-value function $\hat{Q}_{m_i}$ rather than the true action-value function Q_{m_i}. Let the true risk r_i be $(Q(b_i, a) - Q(b_i, a_i^*))$ and the approximate risk $\hat{r}_i$ be $(\hat{Q}(b_i, a) - \hat{Q}(b_i, a_i^*))$. Then error $|r_i - \hat{r}_i|$ due to the point-based approximation is given by:

Lemma 6.1. *Let δ_B be the density of the belief points, defined to be the maximum distance between any reachable belief to the nearest sampled belief point. Then the absolute difference $|r_i - \hat{r}_i|$ may have an error of up to*

$$\epsilon_{PB} = \frac{2(R_{\max} - R_{\min})\delta_B}{(1-\gamma)^2}.$$

Proof. Lemma 6.1 follows directly from the PBVI error bound [27]. □

Now, suppose that no approximation was needed to compute the action-value function, that is, Q_{m_i} could be computed exactly. Let $\hat{BR}'$ be the approximate Bayes risk computed by substituting the action-value function in Eq. (15):

$$\hat{BR}(a)' \approx \sum_i \left(Q^*_{m_i}(b_i, a) - Q^*_{m_i}\left(b_i, a_i^*\right)\right) p_M(m_i).$$

The quality of the approximation $\hat{BR}'$ can be guaranteed with high probability if a sufficient number of sampled models m_i are used:

Lemma 6.2. *To bound the error of the approximation $|BR - \hat{BR}'| < \epsilon_s$ with probability $1-\delta$, n_m independent samples are needed from p_M, where*

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

Proof. The maximum value of the difference $Q(b_i, a) - Q(b_i, a_i^*)$ is trivially lower bounded by $-\frac{R_{\max} - \min(R_{\min}, \xi)}{1-\gamma}$ and upper bounded by zero (note that the risk is always non-positive). Next we apply the Hoeffding bound with sampling error ϵ_s and confidence δ. □

Finally, we can combine the results from Lemmas 6.1 and 6.2 to bound the error in the Bayes risk $|\hat{BR}(a) - BR(a)|$ for any particular action a with high probability:

Lemma 6.3. *Let the number of samples n_m used to approximate the Bayes risk be given by*

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

where δ_B is the sampling density of the belief points used to approximate the action-value function Q_{m_i}. Then the deviation $|\hat{BR}(a) - BR(a)|$ will be less than ϵ with probability $1-\delta$.

Proof. We want to bound the total error by ξ. Lemma 6.1 gives deterministic bound of ϵ_{PB} on any particular value of the difference $|r_i - \hat{r}_i|$. To get an overall bound of ξ, we set $\epsilon_s = \xi - \epsilon_{PB}$ when computing the number of samples needed n_m. □

Before deriving a performance bound from this specific bound, we note that the Markov chain Monte Carlo inference presented in Section 5.2.4 does not produce independent samples m_i: the previous sample is used to propose the next sample. To apply the bounds from Lemma 6.3, one would have to compute the effective number of samples, which would depend on characteristics of the underlying Markov chain. However, the Hoeffding bounds used to derive this approximation are already quite loose; for example in the benchmark POMDP problems presented in Section 7, we obtained good results with 50 samples, whereas Lemma 6.3 suggested over 3000 samples may have been necessary even with a perfect POMDP solver.

6.1.2. Overall performance bound

Lemma 6.3 bounds the risk associated with a single action to be less than ξ. The following theorem provides a bound on the difference between the Bayes-risk agent and an optimal agent:

Theorem 6.4. *Suppose that an agent performs an action whose value is within ξ of the optimal action at every time step with probability δ. Then the expected value V' is lower bounded by*

$$V' > \eta\left(V^* - \frac{\xi}{1-\gamma}\right) + (1-\eta)\frac{R_{\min}}{1-\gamma}, \tag{16}$$

where

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

Proof. From Lemma 6.3, we have that the agent will suffer a loss greater than ξ at time t with probability δ. When the Bayes-risk approximation fails, there are no guarantees on the reward the agent receives; in the worst case it could end up in an absorbing state in which it receives $R_{\min}$ forever.

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 $R_{\min}$. Eq. (17) describes the transitions in this simple chain and the values of the states:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} R-\xi \\ R_{\min} \end{bmatrix} + \begin{bmatrix} 1-\delta & \delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}. \tag{17}$$

The first action puts the agent in state 1 with probability $1-\delta$ and in state 2 with probability δ. Solving the system gives the desired lower bound on $V' = (1-\delta)V_1 + \delta V_2$. □

In general, the actual performance will be much better than the bound in Eq. (16), since most problems do not have an absorbing $R_{\min}$-state from which the agent can never escape.

Finally, we note that all of the performance guarantees in this section are with respect to the prior: we are treating the unknown model m as a random variable, and the optimal policy V is one that makes the best decisions *given the uncertainty p_M in the models*. Theorem 6.4 provides a bound on the expected value V' when making decisions regarding model uncertainty using the Bayes-risk action-selection criterion from Section 5.1 instead of solving the full model-uncertainty POMDP.

6.2. Policy query bounds

Next we consider bounds on the number of policy queries asked. First, we show that the agent will only ask a finite number of policy queries. Second, we show that while it is difficult to characterize exactly how the observation and transition models will evolve over time, we can check if an additional k interactions are sufficient to ensure that the probability of asking a policy query—or equivalently, the probability of encountering a high-risk situation—is below some parameter p_q. We emphasize that the procedures described below are for the purpose of analysis and not implementation. They are *not* meant to be used to determine how long that process will take.

6.2.1. The number of policy queries asked is bounded

First, we argue that the agent will ask only a finite number of policy queries, that is, the number of policy queries is bounded:

Theorem 6.5. *An agent following the algorithm in Table* 1 *will ask only finite number of policy queries over an infinite learning trial.*

Proof. If the transition, observation, and reward models are drawn from finite sets, then the total number of possible models is finite. Since each policy query invalidates at least one POMDP—we would not have requested a query if all the sampled models agreed—we must eventually stop asking policy queries.

A similar argument holds even if the rewards are not drawn from a discrete set. Here, the cost of a policy query limits the model resolution. Suppose a POMDP m has an optimal policy π with value V_m. If we adjust all the model parameters by some small ϵ to create a new model $\hat{m}$, then the value $V_{\hat{m}}$ of the same policy π will be close to V_m [33, Theorem 1]:

$$\|V_m - V_{\hat{m}}\|_\infty \leqslant \frac{\|R-\hat{R}\|_\infty}{1-\gamma} + \frac{\gamma\|\hat{R}\|_\infty}{(1-\gamma)^2} \sup_{s\in S, a\in A} \|T_{s,a} - \hat{T}_{s,a}\|_1.$$

Thus, we can lower-bound the value of the optimal policy under model m in some nearby model $\hat{m}$; specifically, we can create a ball around each model m for which the policies are within ξ. In this way, the value ξ imposes a minimal level

Table 1
Approach for active learning in POMDP models.

Active Learning with Bayes Risk
• Sample POMDP models m from a prior distribution $p_M(m)$. • During an episode, choose actions based on Bayes risk $BR(a; \{b_m\})$, where $\{b_m\}$ is the belief over states held by each model m: – Use the POMDP samples to compute the action with minimal Bayes risk (Section 5.1). – If the risk is larger than a given ξ, perform a policy query (Section 5.1) and augment Ψ with the most recent query and its answer. – Otherwise, perform the domain action with the highest expected reward, and augment h with the action and subsequent observation. – Update each POMDP sample's belief $p_M(m\|h, \Psi)$ based on the new h or Ψ (Section 5.2). • Once an episode is completed, update the belief over models based on the new history (Section 5.2): – Sample POMDP models from a proposal distribution $p_M(m\|h)$ incorporating the action-observation history h. – Weight POMDPs based on policy query history Ψ.
Performance and termination bounds are in Sections 6.1 and 6.2.

of discretization over the reward space. Thus, by invalidating a model m, a policy query invalidates a particular ball in the model space and the number of policy queries will be bounded. □

6.2.2. Probability of a policy query

The specific convergence rate of the active learning will depend heavily upon the problem. Here, we show that one could (in theory) test if k additional interactions reduce the probability of making a policy query to some value p_q with confidence δ_q. The core idea is that each interaction corresponds to a count when computing the Dirichlet posteriors. Once the counts are sufficiently large, the variance of the Dirichlet will be small. Thus, even if the mean of the distribution shifts with time, no additional policy queries will be asked.

Theorem 6.6. *Given a set of state–action visit counts C from the FFBS algorithm, the following procedure will test if k additional interactions reduce the probability of making a policy query to p_q with confidence δ_q:*

1. *Update the counts C with k interactions spread out so as to maximize the entropy of the posterior. Sample n_m models from this posterior, where n_m is computed from Lemma* 6.3.
2. *Compute $p''_q = p_q(1 - 2(1 - \delta))$, where δ is the confidence used in the approximation of the Bayes risk* (*Lemma* 6.3).
3. *Sample $n_b > -27/4 \cdot (p''_q)^{-3} \log \delta_q$ beliefs b uniformly from the belief space.*
4. *Compute the Bayes risk for each belief b. If less than a p''_q-proportion of beliefs require policy queries, then, with confidence δ_q, k is an upper bound on the number of interactions required until probability of asking a policy query is p''_q.*

Proof. We go through each part of the procedure step by step and show that the final computation satisfies the desired bound.

1. **Computing Bayes risk from a conservative posterior**. 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). The maximum-entropy Dirichlet is conservative in that, compared to any other Dirichlet posterior, it spreads out models as much as possible and thus over-estimates the Bayes risk.
2. **Correction for approximate Bayes risk**. Lemma 6.3 gave the number of models n_m required to estimate the Bayes risk within ξ with confidence δ. This bound implies that we may misclassify the risk associated with a belief b with probability δ. Thus, if the true fraction of beliefs requiring policy queries is p_q, we will only observe a fraction $(1-\delta)p_q$ in expectation. Since the beliefs are uniformly sampled, we can apply a Chernoff bound to this expectation to state that with probability $1 - \delta$, no than $2(1 - \delta)n_b$ beliefs will be misclassified.[8]
   Thus, we set our empirical test proportion to $p''_q = p_q(1 - 2(1 - \delta))$.
3. **Sampling a sufficient number of beliefs**. The last step of the procedure involves computing the Bayes risk for n_b beliefs sampled uniformly and accepting the value of k if the empirical proportion of policy queries $\hat{p}_q = n_q/n_b < p_q$, where n_q is the number of beliefs b that require policy queries according to the Bayes-risk computation.
   Minimizing the number of sample n_b needed to guarantee that $\hat{p}_q$ is near p_q with high probability requires two steps. First, we require that $\hat{p}_q$ be within ϵ_q of $p'_q = p_q - \epsilon_q$ with probability δ_q. Next we optimize the associated Chernoff bound $\delta_q = e^{-n_b p'_q \epsilon_q^2/3}$; setting ϵ_q to $2/3p_q$ gives us the desired expression $n_b > -27/4 \cdot (p_q)^{-3} \log \delta_q$. Finally, we substitute p''_q for p_q to account for the approximation in computing the Bayes risk. □

[8] 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.

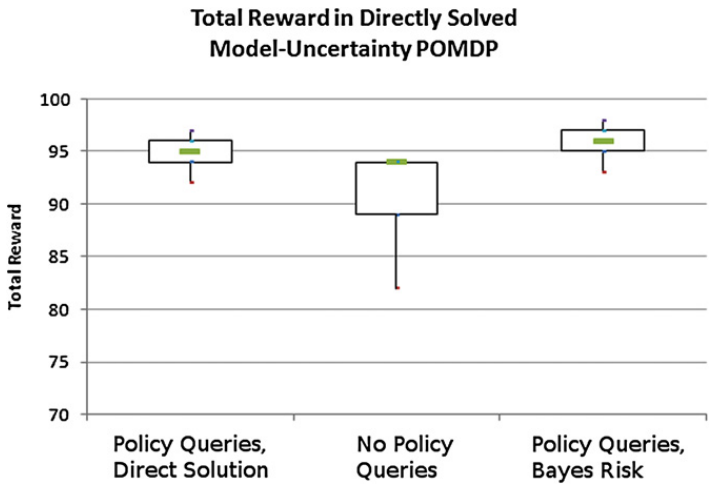


Fig. 3. Box plot of performance for three POMDP learning approaches when learning over a discrete set of four possible models. The medians of the policies are comparable, but the active learner (left) makes fewer mistakes than the passive learner (center). The Bayes-risk action selection criterion (right) does not cause the performance to suffer.

Table 2
Problem descriptions.

Problem	Number of states	Number of actions	Number of obs.
Tiger [29]	2	3	2
Network [29]	7	4	2
Shuttle [35]	8	3	5
Bulkhead [29]	10	6	6
Gridworld (adapted from [29])	25	4	16
Follow (adapted from [36])	26	5	6

7. Results

We first solve a discretized model-uncertainty POMDP to show the utility of policy queries. We next combine the policy queries with our Bayes-risk criterion for learning with continuous-valued unknown parameters. All of the approaches were coded in reasonably-optimized Matlab; depending on the size of the domain, a complete set of learning trials took between an hour to several hours.

7.1. Learning from a limited set of models

In domains where model uncertainty is limited to a few, discrete parameters, we may be able to solve the model-uncertainty POMDP (in which both the world state and the world model are hidden variables) using standard POMDP methods. We consider a simple POMDP-based dialog management task [34] where the reward is unknown. We presume the model contains one of four (discrete) possible reward levels and that the policy query had a fixed associated cost. Fig. 3 compares the performance of the optimal policy—that is, the solution to the model-uncertainty POMDP—*with* policy queries (left column), an optimal policy *without* policy queries (middle column), and our Bayes risk policy *with* policy queries (right column). The difference in median performance is small, but the variance reduction from the policy queries is substantial.[9]

Unfortunately, simply discretizing the model space does not scale when trying to solve model-uncertainty POMDP; adding one more level of discretization, which increases the number of models from four to 48, resulted in poor-quality global solutions when using standard techniques. Next, we present results using our Bayes-risk action selection criterion where we allow the parameters to take on many more values.

7.2. Learning from a general set of models

We applied our approach to learn the complete model—transitions, observations, and rewards—for several standard problems summarized in Table 2. For each problem, 50 POMDP samples were initially taken from a flat $\alpha = 1$ prior over models. The sampled POMDPs were solved very approximately, using relatively few belief points (500) and only 25 backups (see sensitivity analysis in Appendix A for details on the effects of these parameter choices). The policy oracle used a solution to

[9] Although the total reward obtained by 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.

the true model with many more belief points (1000–5000) and 250 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 domain action or asking a policy query ($\xi = -1$) and then taking the supplied optimal action. The probability of expert error p_e was set to 0.3 in all tests. POMDPs were resampled every 100 interactions and the learning process consisted of 5000 total interactions. The entire learning process was repeated 10 times per problem.

The passive learner resampled its POMDP set after updating its prior over transitions and observations using FFBS. Its reward model was drawn from the prior, and it chose the action with the least Bayes risk (regardless of how risky that action was in absolute terms). The active learner used FFBS to sample models given the action–observation histories and re-weighted the samples based on the consistency of their responses to the policy queries. In addition to the Bayes-risk criterion, we avoided common sample-death issues associated with particle filtering by using the importance sampling approach of Section 5.2. We also forced the active learner to ask a policy query if all model weights fell below a threshold. None of the systems received explicit reward information, but the active learner used policy queries to infer information about the reward model. Depending on the problem, tasks required an average of 10 to 30 actions to complete under a near-optimal policy.

Fig. 4 compares the results of the passive learner and the active learner on several standard problems. In general, the active learner maintained a relatively steady level of mean rewards: initially it used the policy queries to choose good actions; later the agent performed well because it had enough data to reason about the model. Thus, the policy queries "covered" for the agent, ensuring good performance while it was still learning. The plots below each performance plot show that the number of policy queries generally decreased. Thus, our agent eventually learned to perform well on its own but used expert knowledge to maintain good performance from the start.

Compared to the results for our previous work [19], this inference approach generally resulted in better relative performance for the active learner. For example, in our previous work active learning improved the passive learner's final rewards by 16.4 in tiger and 80.6 in gridworld; with the new inference the final rewards differed by 60 in tiger and 125 in gridworld. Passive performance also sometimes improves: for example, in shuttle, the passive learner nearly matched the active learner's performance in the end.

Inference in this model is still difficult, however, and problems that require specific sequences of moves—such as BULKHEAD or SHUTTLE—provide a challenge to our approach. The combination of constrained transition spaces with non-smooth reward spaces results in many local optima from which it is challenging to extract excellent solutions, resulting in occasional performance dips during the learning process. The exact reasons for these dips—aside from the fact that the posterior space is generally complex—is difficult to ascertain; developing more robust inference techniques is a subject for future work.

8. Related work

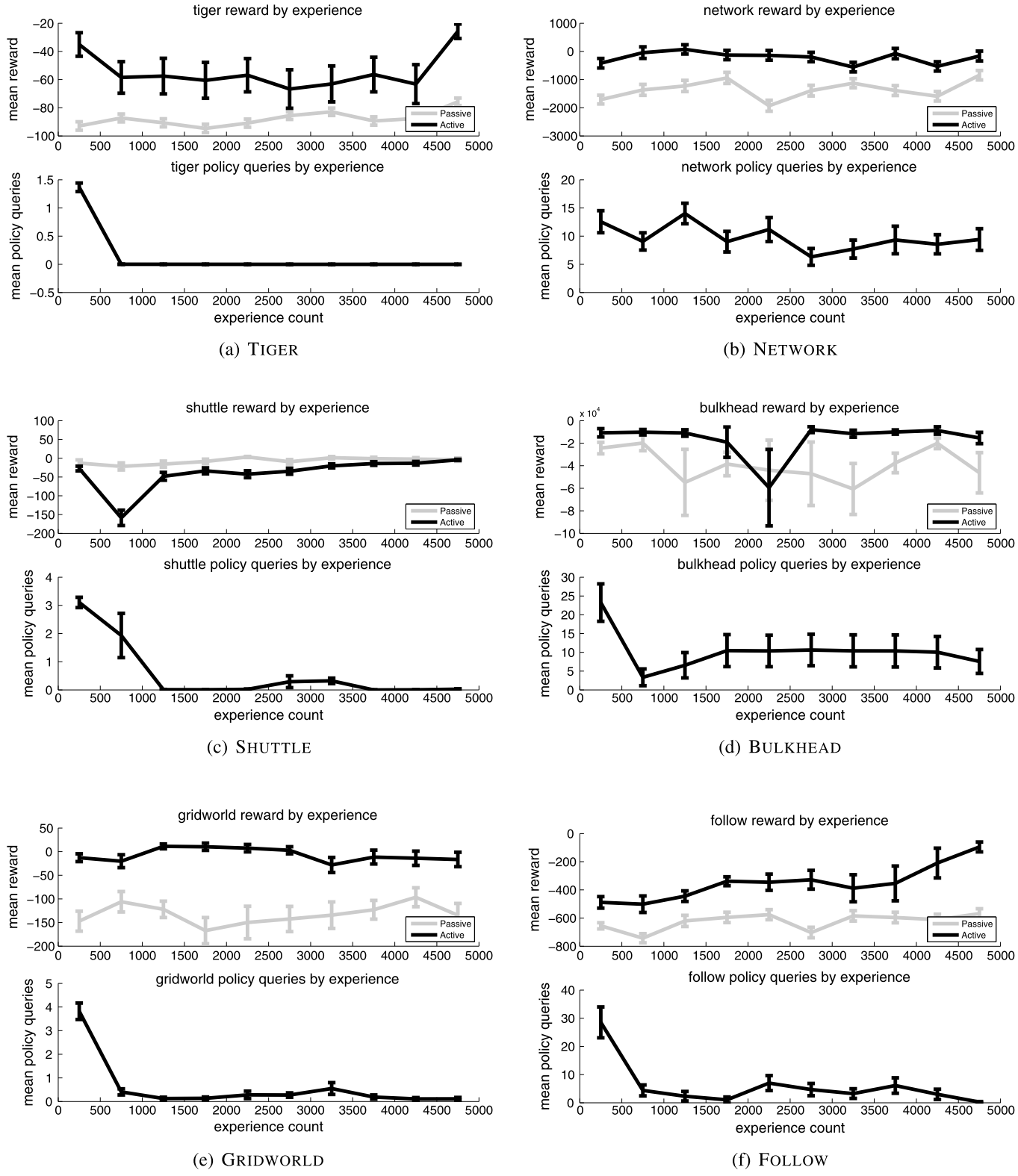
Using Bayesian methods for reinforcement learning is an active area of research. One approach to MDP model learning, the Beetle algorithm [12], converts a discrete MDP into a continuous POMDP with state variables for each MDP parameter. However, their analytic solution does not scale to handle the entire 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. The authors have extended Beetle to partially observable domains [26], providing similar analytic solutions to the POMDP case. The work outlines efficient approximations but results are not provided. Another analytic approach to POMDP learning [37] extends concepts from utile suffix memory to online reinforcement learning. All of these approaches require the rewards to be directly observed—not indirectly inferred from policy queries—and thus are not applicable to our setting.

Related work in MDP and POMDP learning has also considered sampling to approximate a distribution over uncertain models. Dearden et al. [10] discuss several approaches for representing and updating priors over MDPs using sampling and value function updates. Strens [11] shows that for MDP problems, 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 recent work in Bayesian MDPs have provided a formal characterization of the sample complexity required to achieve near-optimal reward [38–40], including cases where the learner has access to a teacher (similar to an oracle) to achieve efficient apprenticeship learning [41]. Their analysis uses some similar assumptions as the results in this paper, but it does not cover partially observable domains.

Extending Bayesian methods to the case of POMDPs, Medusa [13] avoids the problem of knowing how to update the prior by occasionally requesting the true state (queries are made based on model-uncertainty heuristics). It converges to the true model but may make several mistakes before convergence. More recently, Bayes-adaptive POMDPs [36,22] learn POMDPs by incorporating parameter statistics such as Dirichlet counts and Gaussian means into the state space and leveraging the relatively simple iterative updates when applying forward planning. All of these methods assume that the reward function is known.[10]

[10] A somewhat related area of work is planning with imprecise parameters [42,43], but unlike learning approaches, this area assumes that additional interactions with the environment cannot reduce model-uncertainty.



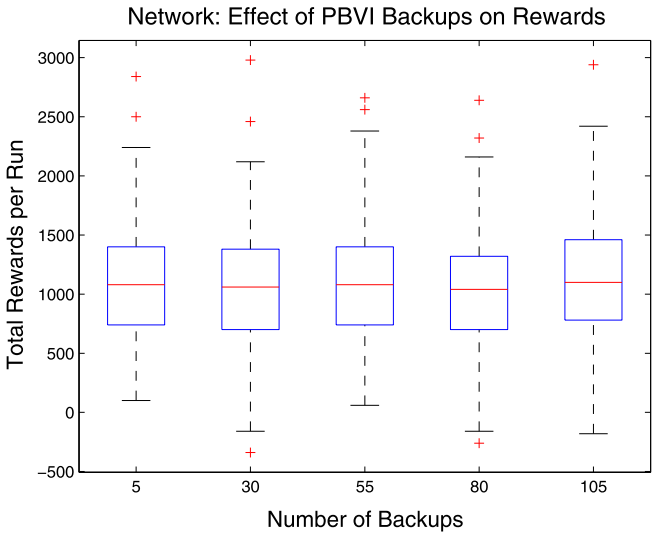
(a) TIGER (b) NETWORK

(c) SHUTTLE (d) BULKHEAD

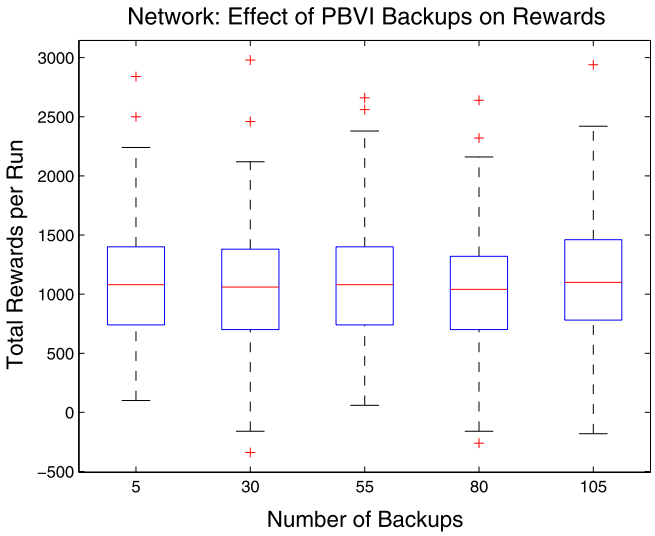
(e) GRIDWORLD (f) FOLLOW

Fig. 4. Performance of the passive learner (gray) and active learner (black) on several standard problems. Error bars show the standard error of the mean reward in each block of 500 counts of experience. Plots below each performance plot show the number of policy queries used in a trial.

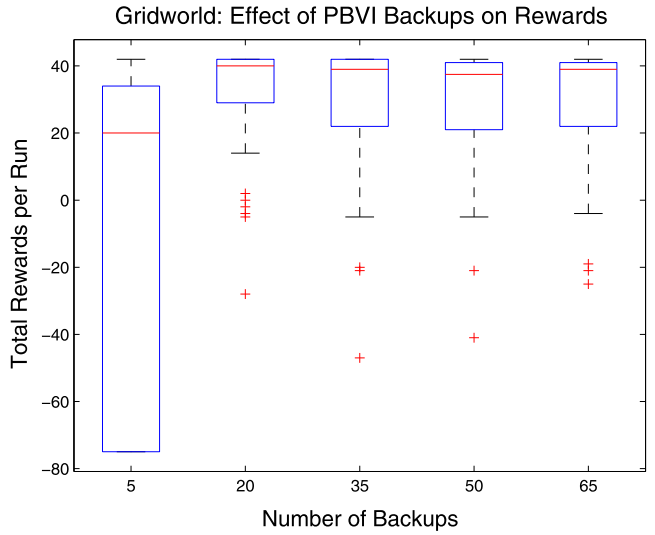
Learning rewards—or preferences—is widely studied in the human-computer and human–robot interaction literature. The preference elicitation literature [44,45] focuses on learning preferences based on queries, but there, learning the rewards is itself the objective: the agent does not have to balance between learning a model and performing a task. An exception is [46], where the transition and observation parameters of a POMDP are considered known, but the reward parameters are unknown. Much of the work in inverse reinforcement learning assumes that states are fully observable [18,47–50]. Extensions to the partially observable case are less common; one recent example showed how to use entire trajectory



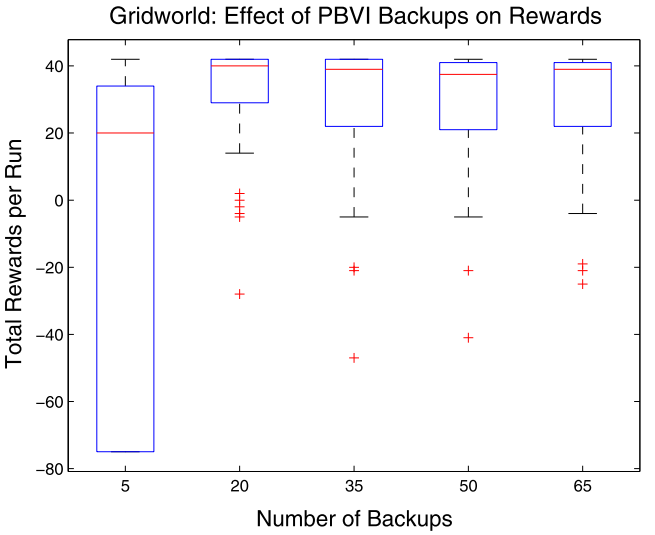
(a) Effect of Belief Points in Network



(b) Effect of Backup Count in Network



(c) Effect of Belief Points in Gridworld



(d) Effect of Backup Count in Gridworld

Fig. 5. Effect of number of belief points and number of backups on two of the test domains. We see that overall, relatively few belief points—around 200—and relatively few backups—around 25—already achieve good performance.

demonstrations as input [51]. Finally, there is a rich body of work on learning by demonstration; for the most part, the objective is quite different, in that the goal is to achieve expert-like behavior, but not necessarily to correctly infer the underlying model.

9. Discussion and conclusion

We presented an approach to POMDP model learning that is both principled and flexible enough for domains requiring online adaptation. Unlike the approaches described in Section 8, our risk-based heuristic and policy queries provide correctness and convergence guarantees throughout the learning process. The risk-based action-selection allows us to choose actions robustly in a computationally tractable manner. The policy queries help us address the issue of robustness by allowing the agent to ask questions when it is confused. The policy queries also allow us to side-step issues of numerical feedback when interacting with humans. To demonstrate our approach on problems from the POMDP literature, we use a sample-based representation of the model posterior. Using importance sampling for the belief update allows us to incorporate heuristic information—such as the approximated model posterior—in a principled manner that reduces the amount of computation required and yet remains faithful to the true posterior.

Our approach for active learning in POMDPs robustly chooses actions and has the flexibility to learn models for a variety of POMDP domains. To scale further, future work should aim to develop more refined proposal distributions—for example,

ones that incorporate some of the policy-query constraints—and gradient-based approaches that can efficiently update particles. We found that in our domains, reward learning—the part of the model inferred from the policy queries—posed the most difficulties in the inference. While transition and observation models could be learned for larger domains than the 26-state, 5-action gridworld presented in this work, learning the reward models for such domains solely from policy queries started to become intractable. Ordinal reward variables, rather than discrete rewards, could also help partition the search space more efficiently.

While incorporating policy queries poses the most inference challenges, other innovations could also reduce computational complexity and thus help scale the concepts in this work to larger domains. First, using adaptive MCMC methods [52] could make the sampling more efficient. Second, since we expect that the model posterior will have smaller changes later in the learning process, heuristics for increasing the number of interactions between resampling steps—such as monitoring the effective number of samples—may reduce computational load. Using fewer samples or running fewer updates may also work well in practice, although developing principled online inference would of course be required to maintain learning guarantees. Finally, as the number of samples needed to represent the posterior grows, using POMDP solvers that allow for iterative solution refinement and methods for allocating more computation to more promising solutions will also be key. These innovations will allow POMDP learning to be deployed on larger, real-time applications.

Appendix A. Sensitivity to POMDP solution parameters

To speed up computations, we used very approximate solutions to the sampled POMDPs in Section 7.2. Here, we show the sensitivity of the approximations—specifically, the number of belief points used and the number of backups applied—in two of the test domains, network and gridworld. For each setting, we ran 10 runs of 75 iterations for 25 random samples of the belief points (for a total of 250 runs per setting). When the running sensitivity analysis on the number of belief points, the number of backups was held fixed at 50; when running the sensitivity analysis for the on the number of backups, the number of beliefs was held fixed at 500.

The plots in Fig. 5 show the distributions of the total rewards received during each run. Very small numbers of belief points (10) or backups (5) have much lower rewards, but even moderate approximations, such as 200 belief points and 25 backups, already have good performance. Thus, our approximations with 500 belief points and 25 backups should have been fairly close to the optimal solution.

References

[1] A.R. Cassandra, L.P. Kaelbling, J.A. Kurien, Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 1996, pp. 963–972.
[2] M.T.J. Spaan, N. Vlassis, A point-based POMDP algorithm for robot planning, in: Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, Louisiana, 2004, pp. 2399–2404.
[3] B. White, An economic analysis of ecological monitoring, Ecological Modeling 189 (2005) 241–250.
[4] Y. Aviv, A. Pazgal, A partially observed Markov decision process for dynamic pricing, Management Science 51 (2005) 1400–1416.
[5] J.D. Williams, S. Young, Partially observable Markov decision processes for spoken dialog systems, Computer Speech Language 21 (2) (2007) 393–422.
[6] J. Hoey, P. Poupart, C. Boutilier, A. Mihailidis, POMDP models for assistive technology, in: Proceedings of the AAAI Fall Symposium on Caring Machines: AI in Eldercare, 2005.
[7] N. Roy, J. Pineau, S. Thrun, Spoken dialogue management using probabilistic reasoning, in: Proceedings of the 38th Annual Meeting of the ACL, Hong Kong, 2000.
[8] G. Shani, R. Brafman, S. Shimony, Forward search value iteration for POMDPs, in: International Joint Conference on AI, 2007.
[9] W.S.L. Hanna Kurniawati, David Hsu, SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces, in: Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland, 2008.
[10] R. Dearden, N. Friedman, D. Andre, Model based Bayesian exploration, in: Proceedings of Uncertainty in Artificial Intelligence, 1999, pp. 150–159.
[11] M. Strens, A Bayesian framework for reinforcement learning, in: International Conference in Machine Learning, 2000.
[12] P. Poupart, N. Vlassis, J. Hoey, K. Regan, An analytic solution to discrete Bayesian reinforcement learning, in: International Conference in Machine Learning, ACM Press, New York, NY, USA, 2006, pp. 697–704.
[13] R. Jaulmes, J. Pineau, D. Precup, Learning in non-stationary partially observable Markov decision processes, in: European Conference in Machine Learning Workshop, 2005.
[14] C. Watkins, Learning from delayed rewards, PhD thesis, Cambridge University, 1989.
[15] A.L. Strehl, L. Li, M.L. Littman, Incremental model-based learners with formal learning-time guarantees, in: Uncertainty in Artificial Intelligence, 2006.
[16] E. Even-Dar, S.M. Kakade, Y. Mansour, Reinforcement learning in POMDPs without resets, in: International Joint Conference on AI, 2005, pp. 690–695.
[17] W.B. Knox, P. Stone, Combining manual feedback with subsequent mdp reward signals for reinforcement learning, in: Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems, 2010.
[18] A. Ng, S. Russell, Algorithms for inverse reinforcement learning, in: International Conference in Machine Learning, 2000.
[19] F. Doshi, J. Pineau, N. Roy, Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs, in: International Conference on Machine Learning, vol. 25, 2008.
[20] E.J. Sondik, The optimal control of partially observable Markov processes, PhD thesis, Stanford University, 1971.
[21] J.M. Porta, N. Vlassis, M. Spaan, P. Poupart, An point-based value iteration for continuous POMDPs, Journal of Machine Learning Research.
[22] S. Ross, B. Chaib-draa, J. Pineau, Bayesian reinforcement learning in continuous POMDPs with application to robot navigation, in: International Conference on Robotics and Automation, 2008, pp. 2845–2851.
[23] F. Doshi, N. Roy, Spoken language interaction with model uncertainty: An adaptive human–robot interaction system, Connection Science 20 (4) (2008) 299–318.
[24] P.D. Moral, A. Doucet, A. Jasra, Sequential Monte Carlo samplers, Journal of The Royal Statistical Society, Series B: Statistical Methodology 68 (2002) 411–436.

[25] J. Williams, S. Young, Scaling up POMDPs for dialogue management: The "Summary POMDP" method, in: Proceedings of the IEEE ASRU Workshop, 2005.
[26] P. Poupart, N. Vlassis, Model-based Bayesian reinforcement learning in partially observable domains, in: International Symposium on AI and Mathematics, 2008.
[27] J. Pineau, G. Gordon, S. Thrun, Point-based value iteration: An anytime algorithm for POMDPs, in: International Joint Conferences on Artificial Intelligence, 2003.
[28] D. Silver, J. Veness, Monte Carlo planning in large POMDPs, in: Proceedings of the Conference on Neural Information Processing Systems, 2010.
[29] M.L. Littman, A.R. Cassandra, L.P. Kaelbling, Learning policies for partially observable environments: Scaling up, in: International Conference in Machine Learning, 1995.
[30] M. Hauskrecht, Value-function approximations for partially observable Markov decision processes, Journal of Artificial Intelligence Research 13 (2000) 33–94.
[31] S.L. Scott, Bayesian methods for hidden Markov models—recursive computing in the 21st century, Journal of the American Statistical Association 97 (2002) 337–351.
[32] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
[33] S. Ross, M.T. Izadi, M. Mercer, D.L. Buckeridge, Sensitivity analysis of POMDP value functions, in: ICMLA, 2009, pp. 317–323.
[34] F. Doshi, N. Roy, Efficient model learning for dialog management, Technical Report SS-07-07, AAAI Press, Palo Alto, CA, 2007.
[35] L. Chrisman, Reinforcement learning with perceptual aliasing: The perceptual distinctions approach, in: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 183–188.
[36] S. Ross, B. Chaib-draa, J. Pineau, Bayes-adaptive POMDPs, Advances in Neural Information Processing Systems 20 (2008) 1225–1232.
[37] G. Shani, R.I. Brafman, S.E. Shimony, Model-based online learning of POMDPs, in: European Conference on Machine Learning, 2005, pp. 353–364.
[38] J. Asmuth, L. Li, M.L. Littman, A. Nouri, D. Wingate, A bayesian sampling approach to exploration in reinforcement learning, in: Proceedings of Uncertainty in Artificial Intelligence, 2009.
[39] J.Z. Kolter, A.Y. Ng, Near-bayesian exploration in polynomial time, in: International Conference on Machine Learning, 2009.
[40] J. Sort, S. Singh, R.L. Lewis, Variance-based rewards for approximate bayesian reinforcement learning, in: Proceedings of Uncertainty in Artificial Intelligence, 2010.
[41] T.J. Walsh, K. Subramanian, M.L. Littman, C. Diuk, Generalizing apprenticeship learning across hypothesis classes, in: International Conference on Machine Learning, 2010.
[42] H. Itoh, K. Nakamura, Partially observable Markov decision processes with imprecise parameters, Artificial Intelligence 171 (2007) 452–490.
[43] A. Nilim, L. Ghaoui, Robustness in Markov decision problems with uncertain transition matrices, in: Proceedings of Neural Information Processing Systems, 2004.
[44] C. Boutilier, A POMDP formulation of preference elicitation problems, in: Eighteenth National Conference on Artificial Intelligence, American Association for Artificial Intelligence, 2002, pp. 239–246.
[45] L. Chen, P. Pu, Survey of preference elicitation methods, Tech. rep., Ecole Politechnique Federale de Lausanne (EPFL), 2004, IC/2004/67.
[46] A. Atrash, J. Pineau, A Bayesian reinforcement learning approach for customizing human–robot interfaces, in: IUI, 2009, pp. 355–360.
[47] W.B. Knox, P. Stone, Combining manual feedback with subsequent MDP reward signals for reinforcement learning, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, 2010, pp. 5–12.
[48] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems 57 (2009) 469–483.
[49] S. Chernova, M.M. Veloso, Interactive policy learning through confidence-based autonomy, Journal of Artificial Intelligence Research 34 (2009) 1–25.
[50] M. Lopes, F.S. Melo, L. Montesano, Active learning for reward estimation in inverse reinforcement learning, in: European Conference on Machine Learning, Bled, Slovenia, 2009.
[51] J. Choi, K.-E. Kim, Inverse reinforcement learning in partially observable environments, in: International Joint Conference on Artificial Intelligence, 2009, pp. 1028–1033.
[52] C. Andrieu, J. Thoms, A tutorial on adaptive MCMC, Statistics and Computing 18 (2008) 343–373.