Prediction-Constrained POMDPs

Joseph Futoma
Harvard SEAS

Michael C. Hughes
Dept of. Computer Science, Tufts University

Finale Doshi-Velez
Harvard SEAS

Abstract

We propose prediction-constrained (PC) training for POMDPs, simultaneously yielding high-reward policies while explaining observed histories well. PC training allows effective model learning even in settings with misspecified models, as it can ignore observation patterns that would distract classic two-stage training.

1 Motivation and Background

The partially observed Markov decision process (POMDP) [Monahan, 1982, Kaelbling et al., 1998] is a popular approach for learning to act in partially-observable domains. When the parameters of the POMDP are unknown (as is typical in reinforcement learning settings), a standard approach to identifying the optimal policy involves two stages: first, we fit transition and observation models given the data, and then we solve the learnt POMDP to obtain a policy [Chrisman, 1992]. However, if not all of the signal in the observations is relevant for decision-making, this two-stage process can result in the first stage wasting modeling effort and the second stage learning inferior policies.

We propose a novel POMDP training objective that balances two goals: providing accurate explanations of the data through a generative model (the POMDP), and learning a high-value policy. This two-term objective ensures that we do not waste computation developing an accurate model for parts of the problem that are irrelevant to decision making. As our method is model-based it will tend to be more sample efficient than alternative model-free deep learning methods, e.g. Hausknecht and Stone [2015]. This is a particular advantage in domains of limited data availability, such as healthcare.

POMDP Background and Notation. We consider POMDPs with K discrete states, A discrete actions, and continuous D-dimensional observations. Let $\tau_{ajk} \equiv p(s' = k|s = j, a = a)$ denote the probability of transitioning from state j to k after taking action a, with $\sum_k \tau_{ajk} = 1$. For each dimension $d \in \{1, 2, \dots D\}$, we independently sample observation $o_d \sim \mathcal{N}(\mu_{kad}, \sigma^2_{kad})$, where k identifies the state just entered (s') and a is the action just taken.

Let $\theta = \{\tau, \mu, \sigma\}$ denote the collection of model parameters. These parameters define an input-output hidden Markov model (IO-HMM) [Bengio and Frasconi, 1995], where the likelihood $p(o|a, \theta)$ of observations given actions can be evaluated via dynamic programming. Given θ and a learned reward function $r(s, a)$ (we do not include r in the likelihood), we can solve the POMDP for the optimal policy. We use point based value iteration (PBVI) [Pineau et al., 2003, Shani et al., 2013], an efficient solver for the small and medium-sized POMDPs we interested in. We adapt ideas from Hoey and Poupart [2005] to handle continuous observations.

2 Proposed Method: Prediction-Constrained POMDP

Unlike existing two-stage methods [Chrisman, 1992, Koenig and Simmons, 1998], which learn θ by maximizing an IO-HMM likelihood alone, our new training objective learns θ by maximizing both the likelihood *and* an estimated value of the policy $\pi(\theta)$ given by PBVI:

$$\max_{\theta} \frac{1}{D(\sum_n T_n)} \sum_{n \in \mathcal{D}^{\text{expl}}} \log p(o_{n,1:T_n}|a_{n,1:T_n-1}, \theta) + \lambda \cdot \text{value}^{\text{CWPDIS}}(\pi(\theta), \pi_{\text{beh}}, \mathcal{D}_{beh}, r, \gamma). \quad (1)$$

The first term is the IO-HMM data likelihood, while the second term is an off-policy estimate of the value of the optimal policy under the model parameters θ (see details below). Conceptually,

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

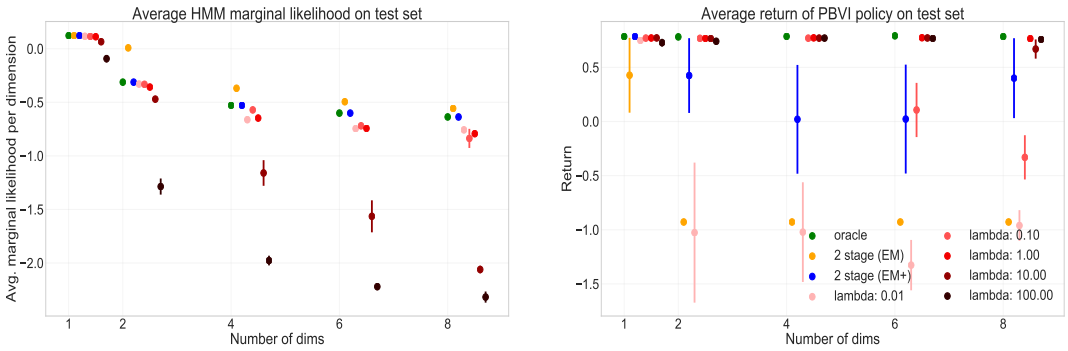

Figure 1: Tiger results as distraction dimensions grow, with $K = 2, \sigma = 0.2$. PC training with $\lambda > 0$ outperforms two-stage EM+PBVI in policy value, while still having reasonable likelihoods. "Oracle" is an ideal θ that models dimension 1 well enough such that $\pi(\theta)$ finds the safe door. EM+ is a heuristic improvement to two-stage training so dimensions correlated with rewards are constrained to have lower σ than other dimensions.

Eq. (1) trades off *generative* and *reward-seeking* properties of the model parameters. The tradeoff scalar $\lambda > 0$ controls how important the reward-seeking term is. We term our approach *prediction-constrained* (PC) training for POMDPs. Our PC-POMDP method extends recent PC objectives for supervising topic models and mixture models [**?**Hughes et al., 2018] to reinforcement learning.

It remains to determine how to quantify the quality of the generative model and the quality of the policy. We optimize the likelihood model on sequences $\mathcal{D}^{\text{expl}}$ collected under an exploration policy. This set covers many possible state-action histories and thus allows better estimation of all entries in the transition and emission parameters $\tau, \mu, \sigma \in \theta$.

For the value of the policy, we cannot simply use the estimated value from the POMDP solver, as a misspecified set of parameters θ could choose to hallucinate an arbitrarily high reward. One choice would be via Monte Carlo roll-outs of the policy. To reuse rollouts, we turn to off-policy estimation. Specifically, we collect rollouts under a reference behavior policy π_{beh} (known in advance). We then use consistent weighted per-decision importance sampling (CWPDIS) [Thomas, 2015] to reweight observed rewards from data collected under π_{beh} to yield a consistent estimate of the long-term value of our model policy $\pi(\theta)$ under discount factor $\gamma \in (0, 1)$. Crucially, this estimator is a *differentiable* function of the model parameters θ, and thus Eq. (1) can be optimized via first-order gradient ascent.

3 Synthetic Experiment: Noisy Tiger Problem

We evaluate our PC approach on a challenging extension of the classic POMDP tiger problem Kaelbling et al. [1998]. A room has K doors; only one door is safe while the remaining $K - 1$ have tigers behind them. The agent has $A = K + 1$ actions: either open one of the doors or listen for noisy evidence of which door is safe to open. Revealing a tiger gives -5 reward, while the safe door yields $+1$ reward, and listening incurs -0.1 reward. Observations o_{nt} have $D \geq 1$ dimensions. Only the first dimension signals the safe door via its mean $i_{\text{safe}} \in \{1, \ldots, K\}$: $o_{nt1} \sim \mathcal{N}(i_{\text{safe}}, \sigma^2)$ where $\sigma = 0.2$. The remaining dimensions are irrelevant, each with random mean $i \sim \text{Unif}(\{1, \ldots, K\})$ and narrow Gaussian standard deviation of 0.1 (less than $\sigma = 0.2$). This environment is designed to confuse the two-stage method that fits θ via an expectation-maximization (EM) procedure that maximizes likelihood only, as this first-stage will prefer explaining the irrelevant but low-noise dimensions, rather than the relevant but higher-noise first dimension. Note K^D states are needed to perfectly model the data but only K states are needed to learn an optimal PBVI policy $\pi(\theta)$. Our proposed PC-POMDP approach with only K states will, given a large enough emphasis on the reward term, favor parameters that focus on the signal dimension and reap better rewards.

Outlook. We anticipate PC training for POMDPs will have advantages when models are misspecified, so $\lambda \gg 0$ encourages rewards to guide parameters while still learning a good HMM. We plan future applications in the medical domain where our approach is uniquely suited to the combination of noisy, possibly irrelevant observations in a batch setting. Our joint-training paradigm also allows us to learn from *semi-supervised* data, where some sequences are missing rewards.

Acknowledgments

FDV and JF acknowledge support from NSF Project 1750358. JF additionally acknowledges Oracle Labs and a Harvard CRCS fellowship.

References

Yoshua Bengio and Paolo Frasconi. An input output HMM architecture. In *Advances in neural information processing systems*, 1995.

Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI*, 1992. URL https://www.aaai.org/Papers/AAAI/1992/AAAI92-029.pdf.

M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. *AAAI*, 2015.

J. Hoey and P. Poupart. Solving pomdps with continuous or large discrete observation spaces. *IJCAI*, 2005.

M. C. Hughes, G. Hope, L. Weiner, T. H. Mccoy, R. H. Perlis, E. Sudderth, and F. Doshi-Velez. Semi-supervised prediction-constrained topic models. In *AISTATS*, 2018.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Sven Koenig and Reid G Simmons. Xavier: A robot navigation architecture based on partially observable markov decision process models. In *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*. Citeseer, 1998.

George E Monahan. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

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

G. Shani, J. Pineau, and R. Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

Philip S. Thomas. *Safe Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, 2015. URL https://people.cs.umass.edu/~pthomas/papers/Thomas2015c.pdf.

Supplement: Prediction-Constrained POMDPs

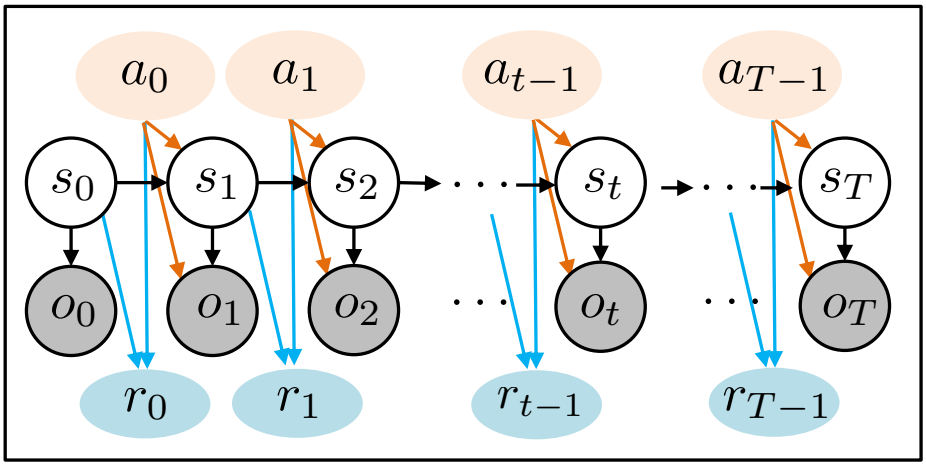

Figure 2: Illustration of the sequential generative process for a sequence of hidden states s, actions a, observations o, and rewards r under a POMDP. Circles with dark outlines represent random variables whose densities are explicitly modeled.

A Additional Details of POMDP model

Fig. 2 illustrates the important sequential relationships between the POMDP variables at each timestep $t = \{0, 1, 2, \dots T\}$ of a sequence.

Initial conditions. Before the agent can act, it must known something about the state of the environment. We assume that at the first timestep $t = 0$, the state $s_0 \in \{1, 2, \dots K\}$ is drawn from a Discrete prior with probability vector τ_0 (length K vector of positive values that sums to one).

$$s_0 \sim p(s_0) \triangleq \text{Discrete}(\tau_{01}, \tau_{02}, \dots \tau_{0K}), \tag{2}$$

We might also see some initial observation $o_0 \in \mathbb{R}^D$, before taking any actions. These are sampled:

$$o_0 \sim p(o_0 | s_0 = k) \triangleq \text{Normal}(\mu_{0k}, \text{diag}(\sigma^2_{0k})) \tag{3}$$

This initial-step D-dimensional multivariate normal likelihood has mean $\mu_{0k} \in \mathbb{R}^D$ and a diagonal covariance matrix whose diagonal is set via the element-wise squaring of standard deviation vector $\sigma_{0k} \in \mathbb{R}^D_+$.

Note that the initial observation *can* be missing; none of the rest of the sequence conditions on this (or any) observations.

Taking actions and reaping rewards. At the first timestep, the agent selects some action a_0 according to its policy. This choice immediately produces some scalar reward $r_0 \in \mathbb{R}$ via a *deterministic* function that depends on the initial state s_0 and the first action a_0: $r_0 \triangleq r(s_0, a_0)$. Similarly, at all subsequent timesteps $t > 0$, after taking action a_t from state s_t, the agent receives reward $r_t \triangleq r(s_t, a_t)$ from the same reward function,

State transitions. Given a current state $s_t = j$ and action $a_t = a$, for any time step $t \geq 0$, the next state s_{t+1} at time $t + 1$ is generated given its immediate past history via the distribution:

$$s_{t+1} \sim p(s_{t+1} | s_t = j, a_t = a) \triangleq \text{Discrete}(\tau_{aj1}, \tau_{aj2}, \dots \tau_{ajK}). \tag{4}$$

Here, each transition probability vector τ_{aj} has K positive entries that sum to one.

Generating observations. Once we've sampled the next state $s_{t+1} = k$, the corresponding observation o_{t+1} is then generated by sampling from a distribution that conditions on the next state and current action:

$$o_{t+1} \sim p(o_{t+1}|s_{t+1} = k, a_t = a) \triangleq \text{Normal}(\mu_{ak}, \text{diag}(\sigma_{ak}^2)). \tag{5}$$

Each action,state pair a, k has its own D-dimensional multivariate normal likelihood with mean vector $\mu_{ak} \in \mathbb{R}^D$ and a diagonal covariance matrix whose diagonal is set via the element-wise squaring of standard deviation vector $\sigma_{ak} \in \mathbb{R}_+^D$.

Termination. When the agent reaches a special terminal state, the POMDP generative process is terminated. We denote the final timestep as T.

A.1 Computing forward beliefs.

Within a POMDP setting, the states remain *hidden*. However, given past actions and observations, we can define a *belief vector* b_t giving the probability of each of the K possible values for state s_t at time t.

For the initial timestep $t = 0$, we have:

$$b_0 = [b_{01},\ b_{02},\ \ldots\ b_{0K}], \quad b_{0k} \triangleq p(s_0 = k|o_0) \tag{6}$$

Then, for subsequent timesteps $t = \{1, 2, \ldots T\}$, we have:

$$b_t = [b_{t1},\ b_{t2},\ \ldots\ b_{tK}], \quad b_{tk} \triangleq p(s_t = k|o_{0:t}, a_{0:t-1}) \tag{7}$$

We call such belief vectors *forward* beliefs, because they are computed by looking "forward" at timestep t given observations and actions from all previous timesteps.

Given an estimate of model parameters $\theta = \tau, \mu, \sigma$, We can easily compute these beliefs via a forward recusion formula defined below. Applying these recursions sequentially across timesteps $t = 0, 1, 2, \ldots T$ is known as the *forward* algorithm in the hidden Markov model literature [**?**]. This is an instance of dynamic programming, where belief values at time $t + 1$ are computed from belief values at time t.

Base case: $t = 0$**.** The initial belief vector is computed via first computing a non-normalized vector $\tilde{b}_0$, and then normalizing to get a b_0 that sums to one:

$$\tilde{b}_{0k} = \text{NormalPDF}(o_0|\mu_{0k}, \text{diag}(\sigma_{0k}^2))\tau_{0k} \qquad \text{for each } k \in \{1, 2, \ldots K\}. \tag{8}$$

$$b_{0k} = \frac{\tilde{b}_{0k}}{\sum_{\ell=1}^K \tilde{b}_{0\ell}} \qquad \text{for each } k \in \{1, 2, \ldots K\}. \tag{9}$$

Recursive case: $t + 1 > 0$**.** We are given the previous belief b_t, the chosen action $a_t = a$, where $a \in \{1, 2, \ldots A\}$, and then next observation o_{t+1}. Then we compute b_{t+1} from b_t via the following

$$\tilde{b}_{t+1,k} = \text{NormalPDF}(o_{t+1}|\mu_{ak}, \text{diag}(\sigma_{ak}^2)) \sum_{j=1}^K b_{tj}\tau_{ajk} \qquad \text{for each } k \in \{1, 2, \ldots K\}. \tag{10}$$

$$b_{t+1,k} = \frac{\tilde{b}_{t+1,k}}{\sum_{\ell=1}^K \tilde{b}_{t+1,\ell}}, \qquad \text{for each } k \in \{1, 2, \ldots K$$

Thus, we have a simple procedure for updating our current belief about time t given a chosen action a_t and next observation o_{t+1}. We can write this recursive update concisely as a function belief_update:

$$b_{t+1} \leftarrow \text{belief\_update}(b_t, a_t, o_{t+1}). \tag{11}$$

A.2 POMDP value functions.

Within a POMDP, a policy π is a mapping from each possible state value s_t to a corresponding discrete action $a_t \in \{1, 2, \ldots A\}$. The agent seeks a policy that optimizes the expected infinite stream of future discounted rewards, where discount factor $\gamma \in (0, 1)$ indicates how much to downweight

rewards far in the future. Given a known state,action to next-state transition distribution T, the expected value of a policy π that starts at state s_0 is:

$$V(\pi, s_0) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{s_{t+1} \sim T(\cdot|s_t, \pi(s_t))} \big[r(s_t, \pi(s_t)) \big] \tag{12}$$

However, with POMDPs we do not know the initial state (or any state) exactly. We thus need to develop a value function $V(\pi, b_0)$ which computes expected rewards given initial *beliefs*. Using the beliefs defined above, the value function can then be written:

$$V(\pi, b_0) = \sum_{t=0}^{\infty} \gamma^t \, \mathbb{E}_{\substack{s_{t+1} \sim T(\cdot|s_t, \pi(b_t)) \\ o_{t+1} \sim O(\cdot|s_{t+1}, \pi(b_t))}} \big[R(b_t, \pi(b_t)) \big], \tag{13}$$

where we have defined the expected reward given belief as $R(b_t, a_t) \triangleq \sum_{k=1}^{K} b_{tk} r(k, a_t)$. Each b_t is computed via the belief_update procedure in Eq. (10).

Given the per-belief value function $V(\pi, b)$ in Eq. (13), we can derive a *greedy* policy for how to act when the current belief vector is b:

$$\pi(b) = \arg\max_a R(b, a) + \gamma \sum_{o \in \mathcal{O}} p(o|b, a) V(\pi, \text{belief\_update}(b, a, o))$$

Here, we define the probability density of the next observation given current belief b_t and current action a_t as $p(o_{t+1}|b_t, a_t) \triangleq \sum_{j=1}^{K} b_{tj} \sum_{k=1}^{K} p(s_{t+1} = k | s_t = j, a_t) p(o_{t+1}|s_{t+1} = k, a_t)$. In this equation, we assume that observations come from a discrete space $\mathcal{O}$, but it is easy to translate to a continuous space by replacing sums with integrals.

It has been long recognized [**?**] that the value function for a POMDP can be modeled arbitrarily closely as the upper envelope of a finite set of linear functions of the belief vector. We denote this set of piecewise functions as $\mathcal{P}$, which has P total pieces. Following common notation, we refer to each linear piece (indexed by p) via a corresponding vector $\alpha_p \in \mathcal{R}^K$ of coefficients.

$$V(\alpha, b) = \max_{\alpha_p \in \mathcal{P}} b^T \alpha_p \tag{14}$$

Each α_p-vector in the set $\mathcal{P}$ has a corresponding action a_p associated with it, where $a_p \in \{1, 2, \ldots A\}$. Thus, the given an accurate piecewise upper envelope, we can define the policy as $\pi(b) = a_{p^*(b)}$, where $p^*(b) = \arg\max_p b^T \alpha_p$. That is, always execute the action associated with the maximizing α-vector.

B Additional Details on Point-Based Value Iteration

Point based value iteration (PBVI) is an algorithm for efficiently solving POMDPs [Pineau et al., 2003]. See [Shani et al., 2013] for a thorough survey of related algorithms and extensions in this area.

In PBVI, we do not perform full Bellman backups over the space of all possible belief points, as this is typically intractable. Instead, we will only perform backups at a fixed set of belief points. We first list an equation that computes the value at a belief b after a Bellman backup over V, where we let r_a denote the vector $R(\cdot, a)$ and we let $b^{a,o}$ denote the result of a belief update, belief(b, a, o):

$$V'(b) = \max_{a \in A} r_a \cdot b + \gamma \sum_o p(o|b, a) V(b^{a,o}) \tag{15}$$

$$= \max_{a \in A} r_a \cdot b + \gamma \sum_o \max_{\alpha \in V} b \cdot \alpha^{a,o}, \tag{16}$$

where $\alpha^{a,o}(s) = \sum_{s'} \alpha(s') p(o|s', a) p(s'|s, a)$. Then, we can use the computation of this value to efficiently compute the new α-vector that would have been optimal for b, had we ran the complete Bellman backup:

$$backup(V, b) = \text{argmax}_{\alpha_a^b : a \in A, \alpha \in V} b \cdot \alpha_a^b \tag{17}$$

$$\alpha_a^b = r_a + \gamma \sum_o \text{argmax}_{\alpha^{a,o} : \alpha \in V} b \cdot \alpha^{a,o} \tag{18}$$

PBVI: Sampling Approximation to Deal with Complex Observation Models In normal PBVI, we are limited by how complex our observation space is. The PBVI backup crucially depends on a summation over observation space (or integration, for continuous observations). Dealing with multi-dimensional, non-discrete observations is generally intractable to compute exactly.

Instead, we will utilize ideas from [Hoey and Poupart, 2005] to circumvent this issue. The main idea is to learn a partition of observation space, where we group together various observations that, conditional on a given belief b and taking an action a, would have the same maximizing α-vector. That is, we want to learn $\mathcal{O}_\alpha = \{o | v = \text{argmax}_{\alpha \in V} \alpha \cdot b^{a,o}\}$. We can then treat this collection of $\mathcal{O}_\alpha$ as a set of "meta-observations", which will allow us to replace the intractable sum/integral over observation space into a sum over the number of α-vectors, by swapping out the $p(o|b,a)$ term from PBVI in (8) with $p(\mathcal{O}_\alpha|b,a)$, the aggregate probability mass over all observations in the "meta-observation". In particular, we can express the value of a belief by the following now:

$$V(b) = \text{max}_a r_a \cdot b + \gamma \sum_\alpha p(\mathcal{O}_\alpha|b,a) V(b^{a,\mathcal{O}_\alpha}) \tag{19}$$

$$p(\mathcal{O}_\alpha|b,a) = \sum_s b(s) \sum_{s'} p(s'|a,s) p(\mathcal{O}_\alpha|s',a) \tag{20}$$

$$b^{a,\mathcal{O}_\alpha} \propto p(\mathcal{O}_\alpha|a,s') \sum_s b(s) p(s'|s,a) \tag{21}$$

$$p(\mathcal{O}_\alpha|a,s') = \sum_{o \in \mathcal{O}_\alpha} p(o|a,s'). \tag{22}$$

We will make use of a sampling approximation that admits arbitrary observation functions in order to approximate these $\mathcal{O}_\alpha$ and (15), the aggregate probability of each "meta-observation".

To do this, first we sample k observations $o_k \sim p(o|s',a)$, for each pair of states and actions. Then, we can approximate $p(\mathcal{O}_\alpha|a,s')$ by the fraction of sampled o_k where α was the optimal α-vector, ie

$$p(\mathcal{O}_\alpha|a,s') \approx \frac{|\{o_k : \alpha = \text{argmax}_{\alpha \in V} \alpha \cdot b^{a,o_k}\}|}{k}, \tag{23}$$

where ties are broken by favoring the α-vector with lowest index. Using this approximate discrete observation function, we can perform point-based backups for V at a set of beliefs B as before. Our backup operation is now:

$$backup(V,b) = \text{argmax}_{\alpha_a^b : a \in A, \alpha \in V} b \cdot \alpha_a^b \tag{24}$$

$$\alpha_a^b = r_a + \gamma \sum_{\alpha'} \underset{\alpha^{a,\mathcal{O}_{\alpha'}}}{\text{argmax}}\ b \cdot \alpha^{a,\mathcal{O}_{\alpha'}} \tag{25}$$

$$\alpha^{a,\mathcal{O}_{\alpha'}}(s) = \sum_{s'} \alpha(s') p(s'|s,a) p(\mathcal{O}_{\alpha'}|a,s'). \tag{26}$$

C Additional Details on Prediction-Constrained POMDPs

Given fixed model parameters θ, we use PBVI to obtain a policy defined by a collection of α-vectors. However, we still need to estimate the value of this implied policy. We do this by using an off-policy estimator, consistent weighted per-decision importance sampling (CWPDIS) [Thomas, 2015].

We assume in this initial work that we have a collection of data, $\mathcal{D}_{beh}$ collected by following a known behavior policy that takes action a_t with probability $\pi_{\text{beh}}(a_t|o_{1:t}, a_{1:t-1})$. We use a similar notation to denote the action probabilities implied by our learned policy, $\pi_\theta(a_t|o_{1:t}, a_{1:t-1})$. Then the CWPDIS estimate of the value of π_θ is:

$$\text{value}^{\text{CWPDIS}}(\pi_\theta, \pi_{\text{beh}}, \mathcal{D}_{\text{beh}}, r, \gamma) \equiv \sum_{t=1}^{T} \gamma^{t-1} \frac{\sum_{n \in \mathcal{D}_{beh}} r_{nt} \prod_{j=1}^{t} \frac{\pi_\theta(a_{nj}|o_{n,1:j}, a_{n,1:j-1})}{\pi_{beh}(a_{nj}|o_{n,1:j}, a_{n,1:j-1})}}{\sum_{n \in \mathcal{D}_{beh}} \prod_{j=1}^{t} \frac{\pi_\theta(a_{nj}|o_{n,1:j}, a_{n,1:j-1})}{\pi_{beh}(a_{nj}|o_{n,1:j}, a_{n,1:j-1})}} \tag{27}$$

where $T = \max\{T_n\}$. Since this objective is a differentiable function of the model parameters θ, it can be optimized via first-order gradient ascent methods. Perturbing θ affects the action probabilities

for our learned policy π_θ in two ways: by changing our current beliefs due to a different model of the environment, and by changing the α-vectors that dictate which action to take. In order to take gradients, we replace the argmax operations in the PBVI backups with softmax operations (in Eqs 16, 17, 18). Likewise we use a stochastic softmax policy, replacing the argmax over α-vectors when applying our policy and choosing an action with a softmax. We start each of these four softmaxes with a temperature of 1 that slowly anneals to a final temperature of 0.1.

Hyperparameters and other settings In this initial work, we optimize the objective in Eq 1 via batch gradient ascent, using backtracking line search to choose the step size. To help improve convergence to a good solution, at the start of optimization for both our PC method and the EM baseline (2 stage EM+ in Figure 1) we decrease the variance of observation dimensions with high correlation with observed rewards (i.e. dimension 1) and increase the variance of the other dimensions, in order to encourage the optimizer to find the desired solution. We also run vanilla EM without this addition (2 stage EM in Figure 1), but it performs much worse. For EM, we try 250 different random initializations, and choose the one with best likelihood on a held-out validation set of $10,000$ trajectories. For our PC approach we try 25 different random initializations, and choose the one with the best objective value (Eq 1) after a small number of gradient steps. During each gradient computation, we perform 10 iterations of PBVI-based backups at a fixed set of 35 belief points evenly spaced between $(0.01, 0.99)$ (this is possible because $K = 2$). We use 100 samples to approximate the likelihood in Eq 18. We learn the reward function separately from the other model parameters by EM even for our PC methods.

Results in Figure 1 are means and standard errors averaged over 25 random seeds. We use $\gamma = 0.9$, and use a sample size of $10,000$ for both $\mathcal{D}^{beh}$ and $\mathcal{D}^{expl}$; both are generated according to random policies that listen at each time with a fixed probability of 0.9, resulting in an average trajectory length of 10. To evaluate the HMM we use a held-out test set of $10,000$, and to evaluate the learned policies, we use $2,500$ trajectories generated using the same random number generator. We implemented our methods using autograd (https://github.com/HIPS/autograd).