Towards Global Reinforcement Learning

Milen Pavlov
University of Waterloo
Waterloo, ON, Canada
milen.pavlov@gmail.com

Pascal Poupart
University of Waterloo
Waterloo, ON, Canada
ppoupart@cs.uwaterloo.ca

Abstract

There has been substantial progress in the field of Reinforcement Learning (RL) in recent years. Many RL applications have achieved remarkable success by taking advantage of as much prior knowledge (PK) as possible. However, they often do so through problem-specific heuristics, due to the lack of a general framework capable of incorporating broad PK. We take the first steps towards a global RL framework - one that incorporates PK over multiple learning components, to learn about all of them simultaneously. Specifically, we consider PK over both model dynamics and policy. We construct a prior belief that reflects both components of PK, use this belief to select actions and update the belief upon new evidence.

1 Introduction

There has been substantial progress in the field of Reinforcement Learning (RL) in recent years. A remarkable example was the successful application of RL to the control logic of a biped robot, enabling it to quickly learn to walk on virtually any terrain [1]. Another was using RL to control a helicopter, performing complex maneuvers in real-time [2]. A common aspect of several success stories is the implicit incorporation of domain knowledge into the learning problem. Domain knowledge (also referred to as *prior knowledge* (PK)) helps by restricting the learning space and thus accelerating the learning process. In the helicopter story, symmetries about the position and orientation of the helicopter are exploited to reduce the complexity of the theoretical model. In the biped story, PK is specified in terms of dependencies between control parameters, which are used to reduce the number of unknown control parameters, which in turn greatly reduces learning time.

While it is clear that incorporating PK in RL problems is beneficial, discovering how to do so is not always straightforward. Due to the lack of a general RL framework to allow the explicit specification of all available PK, RL applications often resort to implicitly incorporating PK through problem-dependent heuristics. This severely limits the generalizability of these approaches.

There exist several frameworks (e.g., Bayesian RL [3], Inverse RL [4] and policy gradient algorithms [5]) that allow for explicit specification of *some* PK. However, in all cases they only allow specification of a single PK component (e.g., model dynamics, reward functions and policies, respectively) and only allow learning about that single component. We take the first steps towards a framework that facilitates the specification and incorporation of PK over *all* components to simultaneously learn about all. The challenge is to combine all pieces of information, keeping in mind that they are related in different ways and, since specified independently, they may present inconsistencies. In this paper, we develop two approaches based on constraint satisfaction techniques to incorporate information about model dynamics and policy, and learn about both simultaneously. While the first approach does not generally converge to an optimal policy, the second approach can be used with any existing Bayesian RL algorithm and offers the same convergence guarantees as the chosen Bayesian RL algorithm.

2 Background

A Markov Decision Process (MDP) is defined as a tuple $\langle S, A, T, R \rangle$, where S is the set of states, A is the set of actions, $T(s, a, s') = \Pr(s' \mid s, a)$ is a transition function that specifies the probability of reaching state s' as a result of performing action a in state s and $R(s, a)$ is a reward function that encodes the rewards associated with performing a in s. A policy $\pi : S \rightarrow A$ is a mapping from states to actions. The goal of RL is to find an optimal policy π^* for an MDP with unknown transition parameters. An optimal policy is defined as a policy that achieves maximum long-term expected rewards from any state. Long-term expected rewards are given using a value function $V^\pi(s)$, which is computed by Bellman's equation: $V^\pi(s) = R(s, \pi(s)) + \gamma \Sigma_{s'} T(s, \pi(s), s') V^\pi(s')$. Therefore, $\pi^*(s) \equiv \operatorname{argmax}_a R(s, a) + \gamma \Sigma_{s'} T(s, a, s') V^{\pi^*}(s')$.

We consider Bayesian RL (BRL) since it offers a principled approach to incorporate a wide range of PK in a belief over the unknown MDP parameters. If we denote each unknown transition value $T(s, a, s')$ with $\theta_{sas'}$ then the belief B is a distribution over all $\theta_{sas'}$. $B : \boldsymbol{\theta} \rightarrow [0, 1]$. The belief is updated after each transition, to take into account the new evidence about T. Specifically, after observing a transition (s, a, s'), the posterior belief B' is computed by Bayes' rule: $B'(\boldsymbol{\theta}) = kB(\boldsymbol{\theta})\theta_{sas'}$. In practice, belief update is made easy if both the prior and posterior belong to the same family of distributions. This is true for beliefs encoded with mixtures of Dirichlets, since Dirichlet distributions are conjugate priors of multinomials [6]. A Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\theta}, \boldsymbol{n}) = k\Pi_i \theta_i^{n_i - 1}$ parameterizes the probability distribution over a multinomial $\boldsymbol{\theta}$ using positive *hyperparameters* n_i, where n_i can be thought of as the number of times the θ_i-probability event has been observed. A Dirichlet mixture $\mathrm{DirMix}(\boldsymbol{\theta}, \boldsymbol{c}, \boldsymbol{n}) = \Sigma_i c_i \Pi_j \theta_j^{n_{ij} - 1}$ is parameterized by weights c_i and hyperparameters $\boldsymbol{n}_i$ that define single Dirichlets. The normalization constants k of single Dirichlets are subsumed by the weights. We can show that belief update using Dirichlet mixtures is performed in closed form using Bayes rule. After a transition $(\hat{s}, \hat{a}, \hat{s}')$, we have

$$\begin{aligned} B'(\boldsymbol{\theta}) &= k\theta_{\hat{s},\hat{a},\hat{s}'} \Sigma_i c_i \Pi_{sas'} (\theta_{sas'})^{n_{isas'}} \\ &= \mathrm{DirMix}(\boldsymbol{\theta}, \boldsymbol{c'}, \boldsymbol{n} + \delta([s, a, s'] = [\hat{s}, \hat{a}, \hat{s}'])) \end{aligned}$$

where $\boldsymbol{c'}$ are the renormalized weights and $\delta(a = b)$ is the Kronecker delta that returns 1 if $a = b$ and 0 otherwise. In practice, belief update amounts to simply incrementing the hyperparameters corresponding to the observed transition and renormalizing.

There are numerous Bayesian algorithms for RL that take advantage of PK to perform efficient learning. For instance, the BEETLE algorithm [7] uses PK over model dynamics to compute a policy mapping from belief states to actions. Bayesian Q-learning [8] takes PK over a measure of the *value of information* that is expected to be gained by performing an exploratory action. This can be thought of as a prior over value functions. Inverse BRL [4] uses PK over reward functions to learn the true reward function. Bayesian policy gradient algorithms [5] take advantage of PK over policy gradients to learn about policies. In a recent paper [9], Doshi *et al.* make use of domain knowledge over both transition dynamics and policy, however, while transition information is taken in the form of a prior distribution in the usual way, policy information is used through an oracle, which may not always be available in learning problems. Note that existing approaches are all limited in the types of PK they can incorporate. In fact, in most cases they only consider PK over a single MDP component, which allows them to learn only about that component. Our work takes the first steps towards a framework that incorporates PK over multiple unknown components and simultaneously learns about all of them.

3 The Global Reinforcement Learning Framework

We present a reinforcement learning framework that incorporates broad types of PK. Specifically, we consider three types of knowledge. The first is a prior probability distribution over transition models, denoted $\bar{P}(\boldsymbol{\theta})$. It is often the case that we have some knowledge about how some actions will affect the world even before starting to learn - such knowledge is stored in $\bar{P}(\boldsymbol{\theta})$. The second is a prior over (stochastic) policies, denoted $\bar{P}(\boldsymbol{\pi})$.[1] Similarly, in many cases we could have information about

[1]This prior can be thought as a belief about the optimality of each policy. While this interpretation is flawed when multiple policies are optimal, this is fine since our work does not distinguish between priors that differ only in their probabilities for optimal policies.

which actions are preferable, even if we do not fully understand their effects - such information is stored in $\bar{P}(\boldsymbol{\pi})$. Finally, the third piece is the knowledge implicit in Bellman's equation. We derive from it a loss function $L(\boldsymbol{\theta}, \boldsymbol{\pi})$ that encodes the degree of inconsistency between a model and a policy (assumed to be optimal).[2]

$$L(\boldsymbol{\theta}, \boldsymbol{\pi}) = \max_s V^*(\boldsymbol{\theta}, s) - V^{\boldsymbol{\pi}}(\boldsymbol{\theta}, s) \tag{1}$$

This third piece of knowledge is present in every learning problem and links $\bar{P}(\boldsymbol{\theta})$ and $\bar{P}(\boldsymbol{\pi})$.

A Bayesian reinforcement learning framework must specify how learning agents perform the following three steps: **1)** incorporating PK into an initial belief, **2)** using this belief to select actions and **3)** updating the belief according to the effects of the performed actions. We explore two ways of implementing the above steps. One encodes the agent's belief as a joint distribution over transition models and policies, while the other - as a marginal distribution over models only.

3.1 Agent's Belief as a Joint Distribution

Step 1. Construct an initial belief distribution from PK. The inputs are $\bar{P}(\boldsymbol{\theta})$, $\bar{P}(\boldsymbol{\pi})$ and $L(\boldsymbol{\theta}, \boldsymbol{\pi})$. The output is the joint distribution $B(\boldsymbol{\theta}, \boldsymbol{\pi})$. We encode this distribution as a joint mixture of Dirichlets, as this presents a convenient way of representing, and later, updating the belief.

$$B(\boldsymbol{\theta}, \boldsymbol{\pi}) = \Sigma_{ij} c_{ij} \Pi_{sas'} (\theta_{sas'})^{n_{isas'}} \Pi_{sa} (\pi_{sa})^{m_{jsa}} \tag{2}$$

The belief is constructed by computing the hyperparameters $\boldsymbol{n}_i$ and $\boldsymbol{m}_j$ of the single Dirichlet distributions, as well as the weights c_{ij} of the mixture. In theory, this belief should satisfy the following constraints:

$$\left| \begin{array}{l} \forall \boldsymbol{\theta} : \int_{\boldsymbol{\pi}} B(\boldsymbol{\theta}, \boldsymbol{\pi}) d\boldsymbol{\pi} = \bar{P}(\boldsymbol{\theta}) \\ \forall \boldsymbol{\pi} : \int_{\boldsymbol{\theta}} B(\boldsymbol{\theta}, \boldsymbol{\pi}) d\boldsymbol{\theta} = \bar{P}(\boldsymbol{\pi}) \\ \forall \boldsymbol{\theta}, \boldsymbol{\pi}, \boldsymbol{\pi'} : L(\boldsymbol{\theta}, \boldsymbol{\pi}) \geq L(\boldsymbol{\theta}, \boldsymbol{\pi'}) \Rightarrow B(\boldsymbol{\theta}, \boldsymbol{\pi}) \leq B(\boldsymbol{\theta}, \boldsymbol{\pi'}) \end{array} \right. \tag{3}$$

However, the variables in the above constraints (namely, the hyperparameters and weights) are expressed in terms of exponents and multiplicative factors, which makes solving this formulation of the problem too difficult. Instead, since the Dirichlet mixture can be thought of as a polynomial, we approximate it by first obtaining basis functions that satisfy the first two constraints independently, then we weigh the basis functions so that all three constraints are satisfied. The latter is made easy by the fact that the only variables are the weights, which are expressed in linear terms.

We consider the unweighted single Dirichlets in (2) as components for building basis functions. Specifically, a basis function $\beta_{ij}(\boldsymbol{\theta}, \boldsymbol{\pi})$ is constructed from its components $\beta_i(\boldsymbol{\theta})$ and $\beta_j(\boldsymbol{\pi})$ in the following way:

$$\beta_{ij}(\boldsymbol{\theta}, \boldsymbol{\pi}) = \beta_i(\boldsymbol{\theta}) \beta_j(\boldsymbol{\pi})$$
$$\beta_i(\boldsymbol{\theta}) = \Pi_{sas'} (\theta_{sas'})^{n_{isas'}}, \quad \beta_j(\boldsymbol{\pi}) = \Pi_{sa} (\pi_{sa})^{m_{jsa}}$$

We construct each set of components iteratively, using the appropriate constraint from (3). We use the first constraint to construct $\beta_i(\boldsymbol{\theta})$. This constraint says that the marginal over models should be equal (or approximately equal) to the prior over models. We achieve this by minimizing some error terms $\varepsilon_{\boldsymbol{\theta}}$, such that

$$\forall \boldsymbol{\theta} : \frac{1}{\varepsilon_{\boldsymbol{\theta}}} \leq \frac{1}{\bar{P}(\boldsymbol{\theta})} \left[\int_{\boldsymbol{\pi}} \Sigma_{ij} c_{ij} \beta_i(\boldsymbol{\theta}) \beta_j(\boldsymbol{\pi}) d\boldsymbol{\pi} \right] \leq \varepsilon_{\boldsymbol{\theta}}$$

We proceed incrementally, starting with the case when $i = j = 1$. Then, there is only one basis function and the above equation reduces to

$$\forall \boldsymbol{\theta} : \frac{1}{\varepsilon_{\boldsymbol{\theta}}} \leq \frac{1}{\bar{P}(\boldsymbol{\theta})} \left[\int_{\boldsymbol{\pi}} c_{11} \Pi_{sas'} (\theta_{sas'})^{n_{1sas'}} \Pi_{sa} (\pi_{sa})^{m_{1sa}} d\boldsymbol{\pi} \right] \leq \varepsilon_{\boldsymbol{\theta}}$$

After pushing the integral to the right and taking the log, we obtain

$$\forall \boldsymbol{\theta} : -\log \varepsilon_{\boldsymbol{\theta}} \leq \log z_1 + \Sigma_{sas'} n_{1sas'} \log \theta_{sas'} - \log \bar{P}(\boldsymbol{\theta}) \leq \log \varepsilon_{\boldsymbol{\theta}}$$

where z_1 is some constant, which subsumes the weight c_{11} and the integral over policies (itself given by the multinomial Beta function). It is not possible to enforce the above constraint over *all* models

[2]While we use the max-norm over s in Eq. 1, any of the L_p norms is fine too.

$\boldsymbol{\theta}$, since there are infinitely many of them. However, we can sample some $\boldsymbol{\theta}$ uniformly at random and make sure it holds for those samples. Let $\boldsymbol{\theta}^k$ denote the k-th of K samples. Then, we have a set of K *linear* constraints to satisfy, while minimizing some norm of $\log \boldsymbol{\varepsilon} = \bigcup_{\boldsymbol{\theta}} \log \varepsilon_{\boldsymbol{\theta}}$. We can see that the expression between the two inequality signs above is equivalent to the difference between vectors $\boldsymbol{Ax}$ and $\boldsymbol{b}$, where

$$\begin{aligned}
\boldsymbol{x} &= \begin{pmatrix} \log z_1 & n_{1s_1a_1s_1} & n_{1s_1a_1s_2} & \cdots & n_{1s_{|S|}a_{|A|}s_{|S|}} \end{pmatrix}^\mathsf{T} \\
\boldsymbol{b} &= \begin{pmatrix} \log \bar{P}(\boldsymbol{\theta}^1) & \log \bar{P}(\boldsymbol{\theta}^2) & \cdots & \log \bar{P}(\boldsymbol{\theta}^K) \end{pmatrix}^\mathsf{T} \\
\boldsymbol{A} &= \begin{pmatrix} 1 & \log(\theta^1_{s_1a_1s_1}) & \log(\theta^1_{s_1a_1s_2}) & \cdots & \log(\theta^1_{s_{|S|}a_{|A|}s_{|S|}}) \\ 1 & \log(\theta^2_{s_1a_1s_1}) & \log(\theta^2_{s_1a_1s_2}) & \cdots & \log(\theta^2_{s_{|S|}a_{|A|}s_{|S|}}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \log(\theta^K_{s_1a_1s_1}) & \log(\theta^K_{s_1a_1s_2}) & \cdots & \log(\theta^K_{s_{|S|}a_{|A|}s_{|S|}}) \end{pmatrix}
\end{aligned}$$

We solve for $\boldsymbol{x}$ by minimizing the Euclidean distance between $\boldsymbol{Ax}$ and $\boldsymbol{b}$. The solution is given by solving the linear system $\boldsymbol{A}^\mathsf{T}\boldsymbol{Ax} = \boldsymbol{A}^\mathsf{T}\boldsymbol{b}$. Finally, from $\boldsymbol{x}$ we can extract the hyperparameters $\boldsymbol{n}_1$, which give us the basis function component $\beta_1(\boldsymbol{\theta})$.

Now we look at the difference between $\boldsymbol{Ax}$ and $\boldsymbol{b}$. This difference is known as the *residual*. Let us denote the positively-shifted residual (formally defined below) with $\boldsymbol{\psi}^1$. We will attempt to fit another monomial to approximate $\boldsymbol{\psi}^1$. To do this we use the same matrix $\boldsymbol{A}$ as above, but we set $\boldsymbol{b} = \log \boldsymbol{\psi}^1$. The minimization is performed in the same way as above to obtain $\boldsymbol{n}_2$. In general, to find hyperparameters $\boldsymbol{n}_i$ we need to determine the value of each ψ^i_k,

$$\begin{aligned}
&\psi^i_k = \bar{P}(\boldsymbol{\theta}^k) - \Sigma_{j=1}^{i-1} z_j \Pi_{sas'}(\theta_{sas'})^{n_{jsas'}} - \text{shift}_{i-1}, \\
&\text{where } \text{shift}_{i-1} = \min_{1\le k\le K} \left\{ \bar{P}(\boldsymbol{\theta}^k) - \Sigma_{j=1}^{i-1} z_j \Pi_{sas'}(\theta_{sas'})^{n_{jsas'}} \right\} - 1
\end{aligned}$$

and minimize the Euclidean distance between $\boldsymbol{Ax}$ and the new $\boldsymbol{b}$,

$$\boldsymbol{b} = \begin{pmatrix} \log \psi^i_1 & \log \psi^i_2 & \cdots & \log \psi^i_K \end{pmatrix}^\mathsf{T}$$

Note that taking the log of ψ^k_i is always allowed, since the shift term ensures $\boldsymbol{\psi}^i$ is strictly positive. In particular, the -1 term in the shift ensures $\boldsymbol{\psi}^i$ is never zero, while the rest ensures it is never negative. The iteration continues until the residual drops below a certain threshold, meaning our desired level of approximation has been achieved.

Applying the same methodology to the second constraint in (3) we can find the hyperparameters $\boldsymbol{m}_j$. In this way, we construct all basis functions $\beta_{ij}(\boldsymbol{\theta}, \boldsymbol{\pi})$. Next, we must weigh them properly so as to satisfy all three constraints. Using basis function notation, this means

$$\begin{aligned}
\text{minimize} \quad & \varepsilon \\
\text{subject to} \quad & \forall \boldsymbol{\theta}^k : -\varepsilon \le \int_{\boldsymbol{\pi}} \Sigma_{ij} c_{ij} \beta_{ij}(\boldsymbol{\theta}^k, \boldsymbol{\pi}) d\boldsymbol{\pi} - \bar{P}(\boldsymbol{\theta}^k) \le \varepsilon \\
& \forall \boldsymbol{\pi}^{k'} : -\varepsilon \le \int_{\boldsymbol{\theta}} \Sigma_{ij} c_{ij} \beta_{ij}(\boldsymbol{\theta}, \boldsymbol{\pi}^{k'}) d\boldsymbol{\theta} - \bar{P}(\boldsymbol{\pi}^{k'}) \le \varepsilon \\
& \forall \boldsymbol{\theta}^k, \boldsymbol{\pi}^{k'}, \boldsymbol{\pi}^{k''} : L(\boldsymbol{\theta}^k, \boldsymbol{\pi}^{k'}) \ge L(\boldsymbol{\theta}^k, \boldsymbol{\pi}^{k''}) \Rightarrow \Sigma_{ij} c_{ij} \beta_i(\boldsymbol{\theta}^k) \left[\beta_j(\boldsymbol{\pi}^{k'}) - \beta_j(\boldsymbol{\pi}^{k''}) \right] \le \varepsilon
\end{aligned}$$

Note that the only variables are the the weights c_{ij}, which are expressed in linear terms. We use the Simplex method [10] to solve for them. The values of the weights and hyperparameters together define the agent's initial belief B.

Step 2. Action selection is done by sampling a policy from the belief's marginal. The belief's marginal is given by

$$\int_{\boldsymbol{\theta}} B(\boldsymbol{\theta}, \boldsymbol{\pi}) = \int_{\boldsymbol{\theta}} \Sigma_{ij} c_{ij} \beta_i(\boldsymbol{\theta}) \beta_j(\boldsymbol{\pi}) d\boldsymbol{\theta} = \Sigma_j z_j \Pi_{sa}(\pi_{sa})^{m_{jsa}} \tag{4}$$

where z_j is a normalization constant, computed by the multinomial Beta function. We can see that the marginal has the form of a Dirichlet mixture. The agent samples a policy from this mixture and selects an action according to that policy. This action selection method naturally progresses from high exploration to high exploitation as the marginal belief over policies converges.

Step 3. Suppose an agent with current belief B is in state $\hat{s}$, performs action $\hat{a}$ and observes next state $\hat{s}'$. Then, the updated belief B' is computed as follows.

$$\begin{aligned}
B'(\boldsymbol{\theta}, \boldsymbol{\pi}) &= p(\boldsymbol{\theta}, \boldsymbol{\pi} \mid \hat{s}, \hat{a}, \hat{s}') \\
&= \frac{p(\boldsymbol{\theta}, \boldsymbol{\pi})\, p(\hat{s}, \hat{a}, \hat{s}' \mid \boldsymbol{\theta}, \boldsymbol{\pi})}{p(\hat{s}, \hat{a}, \hat{s}')} \\
&= z\, B(\boldsymbol{\theta}, \boldsymbol{\pi})\, \theta_{\hat{s},\hat{a},\hat{s}'} \\
&= z\, \Sigma_{ij} c_{ij} \Pi_{sas'} (\theta_{sas'})^{n_{isas'} + \delta([s,a,s'] = [\hat{s},\hat{a},\hat{s}'])} \Pi_{sa} (\pi_{sa})^{m_{jsa}}
\end{aligned}$$

where the normalization factor z can be factored into the weights c_{ij} and computed using the multinomial Beta function for each corresponding Dirichlet. Note that in practice the update is performed simply by incrementing the hyperparameters corresponding to the observed transition.

Known Limitations. There is one major limitation of the approach discussed in this section. Recall that learning about the policy is performed by increasing the weights of policy basis function components β_j that correspond to high-utility policies. However, the components β_j are pre-computed and their number - finite. When the agent has converged to the underlying model of the environment, he will increase the weight of the best component β_j^* and will act according to policies sampled from that component. However, this is not equivalent to using the optimal policy. In fact, to guarantee convergence to the optimal policy one would have to include in the agent's initial belief all possible components β_j, of which there are infinitely many. It is clear that encoding the agent's belief as a joint mixture of Dirichlets, while convenient, does not generally guarantee optimal rewards.

3.2 Agent's Belief as a Marginal Distribution

Another approach we consider is one where the belief is encoded as a marginal mixture of Dirichlets over models only. As we will see, this allows the agent to incorporate Bellman's equation in a way that avoids the problematic need for infinitely many basis functions.

Step 1. The inputs are the same as before: $\bar{P}(\boldsymbol{\theta})$, $\bar{P}(\boldsymbol{\pi})$ and $L(\boldsymbol{\theta}, \boldsymbol{\pi})$. However, the output consists of two distributions: a marginal $B(\boldsymbol{\theta})$ over transition models and a conditional distribution $P(\boldsymbol{\pi} \mid \boldsymbol{\theta})$ that tells us how consistent a given policy is with a given transition model. Using those two distributions, we can obtain the joint distribution trivially: $B(\boldsymbol{\theta}, \boldsymbol{\pi}) = B(\boldsymbol{\theta}) P(\boldsymbol{\pi} \mid \boldsymbol{\theta})$. The conditional distribution $P(\boldsymbol{\pi} \mid \boldsymbol{\theta})$ is constructed directly from the loss function.

$$P(\boldsymbol{\pi} \mid \boldsymbol{\theta}) = \frac{e^{-L(\boldsymbol{\theta}, \boldsymbol{\pi})t}}{\Sigma_{\boldsymbol{\pi}} e^{-L(\boldsymbol{\theta}, \boldsymbol{\pi})t}}$$

where t is a temperature parameter that specifies how peaked the resulting distribution will be. This method is similar to Boltzmann exploration. Since the belief $B(\boldsymbol{\theta})$ is taken only with respect to transition models, we can encode it using a single mixture of Dirichlets.

$$B(\boldsymbol{\theta}) = \Sigma_i c_i \Pi_{sas'} (\theta_{sas'})^{n_{isas'}} \tag{5}$$

Then, we need to find the weights and hyperparameters that satisfy the following constraints.

$$\left| \begin{array}{l} \forall \boldsymbol{\theta} : B(\boldsymbol{\theta}) = \bar{P}(\boldsymbol{\theta}) \\ \forall \boldsymbol{\pi} : \int_{\boldsymbol{\theta}} P(\boldsymbol{\pi} \mid \boldsymbol{\theta}) B(\boldsymbol{\theta}) d\boldsymbol{\theta} = \bar{P}(\boldsymbol{\pi}) \end{array} \right.$$

As before, since we cannot enforce the constraints for all $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$, we sample K transition models $\boldsymbol{\theta}^k$ and K' policies $\boldsymbol{\pi}^{k'}$ uniformly at random, and attempt to satisfy the constraints for those samples. Using the same sampled points, we can also approximate the integral in the second constraint with a sum. However, we must be careful to ensure the probabilities of sampled transition models sum up to one. We introduce auxiliary variables $p_k \in [0, 1]$ (one for each sampled $\boldsymbol{\theta}^k$) and rewrite the constraints as follows.

$$\left| \begin{array}{l} \forall \boldsymbol{\theta}^k : \Sigma_i c_i \Pi_{sas'} (\theta^k_{sas'})^{n_{isas'}} = \bar{P}(\boldsymbol{\theta}^k) \\ \forall \boldsymbol{\pi}^{k'} : \Sigma_k P(\boldsymbol{\pi}^{k'} \mid \boldsymbol{\theta}^k)\, p_k = \bar{P}(\boldsymbol{\pi}^{k'}) \end{array} \right.$$

$$p_k = \frac{\Sigma_i c_i \Pi_{sas'} (\theta^k_{sas'})^{n_{isas'}}}{\Sigma_{k'} p_{k'}}$$

We proceed by using the second constraint alone to compute the p_k values. This is straightforward, since the constraint represents a linear system. Next, we satisfy the remaining constraints to obtain the values of the weights and hyperparameters of the belief $B(\boldsymbol{\theta})$. This is done by building and weighing basis functions in the same way as described in the previous section.

This approach has the advantage of building only basis functions corresponding to transition models and using them, together with the conditional distribution $P(\boldsymbol{\pi} \mid \boldsymbol{\theta})$, to reason about optimal policies. Since the basis functions are constantly updated through the closed-form belief update, even in the case where erroneous PK is given the framework will still be able to converge to the true model of the environment (albeit more slowly) and select optimal actions from then on.

Steps 2 and 3. Note that the belief constructed in the previous step incorporates three types of PK, as discussed before. Nevertheless, it is much like conventional beliefs in the Bayesian reinforcement learning literature, in the sense that it is encoded as a probability distribution over transition models. Therefore, any existing approach can be used for the action selection and belief update steps. In our experiments, we consider the BEETLE algorithm [7].

4 Experiments and Results

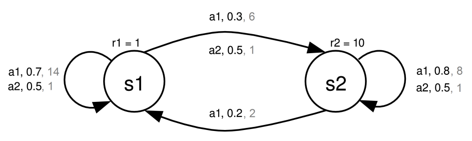
We perform four tests to assess the performance of the GRL framework: (1) we test whether the two proposed approaches converge to the optimal policy, (2) whether constructing a belief by incorporating PK over policies and model dynamics is at all worthwhile, (3) how varying the parameters of the model affects accuracy and (4) how the framework scales with increasing problem size.

We perform the experiments on two learning environments: a trivial 2-state 2-action abstract world and a more realistic task assistance scenario. Fig. 1a shows the 2-state world, its actions, transition probabilities and rewards. This world serves as a "proof-of-concept" for verifying basic algorithmic functionality. It is clearly not suitable for testing more complex properties, such as scalability and accuracy. Inspired by assistive technologies such as robotic assistants in nursing homes [11] and decision-theoretic models for task assistance [12], we also test our framework on a task assistance scenario. This scenario defines a set of state variables, which a person can modify in order to reach some goal configuration. The actions in the model are prompt cues, that are given to the person to assist them with making correct choices. The system must learn about the unknown behaviour of each person at different steps of the task. Prior knowledge about a specific person or the population can tremendously speed up adaptation to the person. This task assistance scenario allows us to vary the size of the problem by varying the number of state variables that define the state of the world. We vary the number of state variables from 2 to 7, which gives us problems of 36 to 8192 states and 18 to 128 unknown parameters.

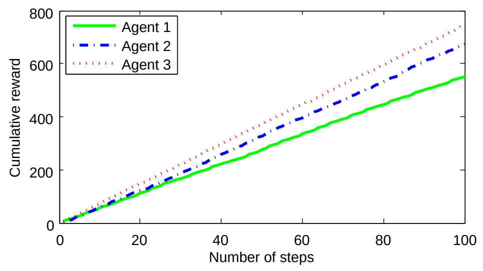
Prior Knowledge. For the 2-state problem, we specify PK using Dirichlet distributions with hyperparameters as specified in Fig. 1a (with values in grey). Intuitively, we can interpret the values as occurrence counts for particular events (e.g., "I believe performing $a1$ in $s1$ has a high chance of leading back to $s1$, as if I had observed this happen 14 times out of $14 + 6 = 20$ trials"). In this test problem, our agent has a strong sense of the effects of action $a1$, but is uncertain about $a2$'s. The optimal policy here is to perform $a2$ in $s1$ and $a1$ in $s2$. We give the agent a strong belief about the policy by setting the hyperparameters corresponding to optimal actions to 50 and the rest to 1.

The task assistance problem is modeled as a factored MDP. The structure and parameters of the conditional probability tables (CPTs) are known except for the behaviour of the user. In many applications including this one, it is reasonable to assume that the environment dynamics can be estimated from data except for human behaviours, which vary with each person. Hence, the agent learns the behaviour CPT as it interacts with the user. Since it is often the case that users can carry out most of the steps independently, we start with model and policy priors that mildly believe that not prompting is optimal and the user will exhibit the correct behaviour at each step of the task (hyperparameters set to 3 for no prompt and correct behaviour and 1 for all prompts and incorrect behaviours). Note that a caregiver familiar with the user could specify a stronger and personalized prior for the expected behaviour and the best prompting strategy.

Experiment 1: Convergence. We test whether the two approaches discussed in Sect. 3 eventually converge to the optimal policy. We consider three agents: Agent 1 acts using a joint mixture belief, Agent 2 uses a marginal mixture belief and Agent 3 uses the optimal policy. The last agent is considered as a frame of reference. We use the 2-state problem for this experiment. Cumulative rewards gained by the three agents over 100 steps are shown in Fig. 1b.



a)



b)

Figure 1: a) A simple 2-state test world. Transition probabilities are given by middle numbers of arc labels. Trailing numbers (in grey) specify PK over models in terms of hyperparameter values. b) Rewards accumulated by agents using different learning strategies. Agent 1 uses a joint mixture belief, Agent 2 - a marginal mixture belief, Agent 3 - the optimal policy.

Agent 2 converges to the optimal policy reasonably quickly - this is evident from the fact that his reward curve becomes roughly parallel to that of Agent 3. However, Agent 1 converges to a suboptimal policy. This is due to the limitation of the joint mixture approach discussed in Sect. 3. Due to this limitation, we do not consider the joint mixture approach in the following experiments.

Experiment 2: Utility of Joint Belief. We test whether incorporating PK over policies and transition models in the initial belief is beneficial. For the 2-state world, we construct the agent's belief $B(\boldsymbol{\theta})$ using the marginal mixture approach of Sect. 3. This belief takes into account the aforementioned PK. We also construct the naïve belief, $B'(\boldsymbol{\theta}) = \bar{P}(\boldsymbol{\theta})$. To see which belief offers a better representation of the true environment, we compare the average Euclidean distance between the true model of the environment $\boldsymbol{\theta}^*$ and K sampled models $\boldsymbol{\theta}_B^k$ and $\boldsymbol{\theta}_{B'}^k$ from the two beliefs B and B'. Running 50 simulations with $K = 1000$ samples from each belief yields average distances that are $||\boldsymbol{\theta}^* - \boldsymbol{\theta}_B^k|| = 0.5645$ and $||\boldsymbol{\theta}^* - \boldsymbol{\theta}_{B'}^k|| = 0.7155$. This shows a 21% improvement in belief accuracy using B, which confirms the intuitive claim that incorporating additional PK (namely, $\bar{P}(\boldsymbol{\pi})$ and $L(\boldsymbol{\theta}, \boldsymbol{\pi})$) pays off. In general, incorporating any good PK over policies will improve accuracy.

Experiment 3: Accuracy. Since our approach approximates some integrals and samples constraints over the parameter space, we verify how the quality of the approximation varies with the number of model parameters. We evaluate accuracy by computing the inverse of the average distance between the true model and sampled models from the constructed belief. Using the notation introduced in the previous experiment, we have: $\mathrm{distance} = \Sigma_k ||\boldsymbol{\theta}^* - \boldsymbol{\theta}_B^k||/K$. We perform this test on the task assistance scenario, by varying two parameters: the size of the learning problem and the number of sampled points used in belief construction. Fig. 2a shows the distance surface for $K = 1000$ in problems of 18 to 128 unknowns when the belief is constructed using 100 to 1000 samples. As expected, accuracy is reduced with increasing problem size. On the other hand, increasing the number of sampled points for belief construction seems to help up to 300 samples and does not show a significant increase any further. This indicates that using a sample size of 300 is a good choice for the task assistance problem. We also computed distances in terms of KL-divergence (instead of Euclidean norm), however the surface was similar to Fig. 2a, so it is not included here.

Experiment 4: Scalability. The bottleneck of the marginal mixture method is building the loss function, which has to compute value functions for every model-policy pair. Asymptotic (theoretical) analysis shows that the time required is quadratic in the number of samples and quadratic in the number of states. The marginal mixture method also exhibits a cubic dependency on the number of unknown parameters, due to solving linear constraints. However, in practice this is overshadowed by the computation of the loss function. Fig. 2b shows belief construction times for task assistance.

Note that we did not include any experiment to evaluate the overall performance of Bayesian RL based on the marginal mixture approach since any existing approach for belief updating and action selection can be used and these approaches have already been evaluated in the literature.

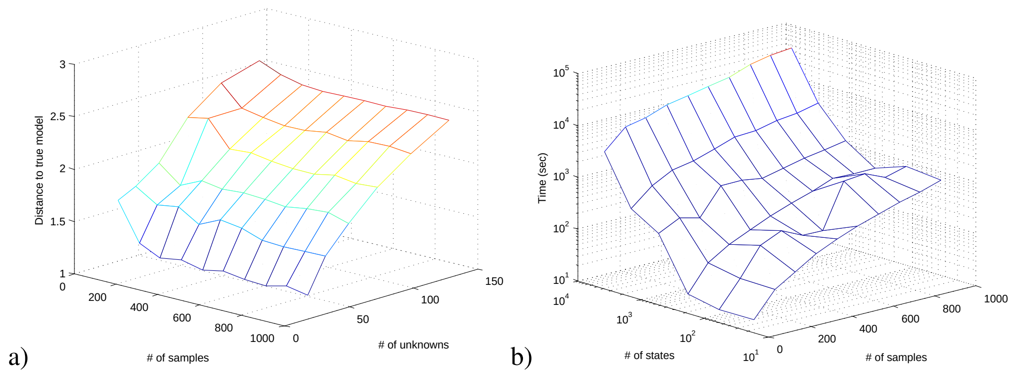


Figure 2: a) Distance surface for various model parameters. b) Belief construction time.

5 Conclusion

We proposed a framework that incorporates PK in terms of model dynamics, policy and the knowledge implicit in Bellman's equation. We discussed two methods for belief construction, of which one was shown to overcome limitations intrinsically present in the other. We empirically showed that broad PK is beneficial to incorporate and that our framework scales reasonably well with increasing problem size. This work presents the first step towards a Global reinforcement learning.

Future work. Global reinforcement learning is only in its infancy. Here we have discussed the encoding of PK about model dynamics and policy. The other obvious component we would like to incorporate PK about is the optimal value function. It may also be worthwhile to investigate other ways of specifying PK. Currently, we incorporate PK in terms of distributions over models, however, it should also be possible to specify particular constraints on actions or transition probabilities and incorporate those directly into the constraint satisfaction problems used for belief construction. Finally, the framework could be extended to partially observable domains.

References

[1] R. Tedrake, T. Weirui Zhang, and H.S. Seung. Learning to walk in 20 minutes. In *Yale Workshop on Adaptive and Learning Systems*, 2005.

[2] A.Y. Ng, H.J. Kim, M.I. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *NIPS*, 2003.

[3] M.J.A. Strens. A Bayesian framework for reinforcement learning. In *ICML*, pages 943–950, 2000.

[4] U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *ICML*, pages 35–42, 2001.

[5] M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *NIPS*, pages 457–464, 2006.

[6] M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

[7] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *ICML*, pages 697–704, 2006.

[8] R. Dearden, N. Friedman, and S.J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[9] F. Doshi, N. Roy, and J. Pineau. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *ISAIM*, 2008.

[10] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1997.

[11] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3-4):271–281, 2003.

[12] J. Hoey, P. Poupart, C. Boutilier, and A. Mihailidis. POMDP models for assistive technology. Technical report, IATSL, 2005.