Compositional Policy Priors

David Wingate, Carlos Diuk, Timothy O'Donnell,
Joshua Tenenbaum, and Samuel Gershman

CSAIL

Compositional Policy Priors

David Wingate, Lyric Labs
Carlos Diuk, Department of Psychology, Princeton University
Timothy J. O'Donnell, BCS, MIT
Joshua B. Tenenbaum, BCS and CSAIL, MIT
Samuel J. Gershman, BCS, MIT

April 10, 2013

Abstract

This paper describes a probabilistic framework for incorporating structured inductive biases into reinforcement learning. These inductive biases arise from *policy priors*, probability distributions over optimal policies. Borrowing recent ideas from computational linguistics and Bayesian nonparametrics, we define several families of policy priors that express compositional, abstract structure in a domain. Compositionality is expressed using probabilistic context-free grammars, enabling a compact representation of hierarchically organized sub-tasks. Useful sequences of sub-tasks can be cached and reused by extending the grammars nonparametrically using Fragment Grammars. We present Monte Carlo methods for performing inference, and show how structured policy priors lead to substantially faster learning in complex domains compared to methods without inductive biases.

1 Introduction

In many reinforcement learning (RL) domains, optimal policies have a rich internal structure. Learning to cook, for example, is inherently *hierarchical*: It involves mastering a repertoire of simple sub-tasks and composing them into more complicated action sequences. These compositions are often reused in multiple recipes with different variations. Thus, acquiring the underlying hierarchical structure of a cooking policy, together with a model of inter-recipe variability, can provide a powerful, abstract inductive bias for learning new recipes.

1

We approach the problem of discovering and exploiting abstract, compositional policy structure from the perspective of probabilistic inference. By defining a stochastic generative model by which optimal policies are generated and in turn give rise to observed rewards, we can treat the hypothesis that a policy is optimal as a latent variable, and invert the generative model (using Bayes' rule) to infer posterior probability of the hypothesis. In essence, this is a Bayesian version of policy search, an old idea from the optimization literature (Jones et al., 1998; Kushner, 1963; Mockus, 1982) that has been recently revived in machine learning and robotics (Martinez-Cantin et al., 2009; Srinivas et al., 2010; Wilson et al., 2010); a recent review can be found in Brochu et al. (2010).

While policy search methods avoid the problem of building a model of the environment, they are traditionally plagued by poor sample complexity, due to the fact that they make wasteful use of experience. Rather than building a model of the environment, we propose to build a model of optimal policy structure. Our key innovation is to define *compositional policy priors* that capture reusable, hierarchically-organized structure across multiple tasks. We show that, compared to unstructured policy priors, structured priors allow the agent to learn much more rapidly in new tasks.

The policy priors we propose are parameterized such that they can be used to *discover* the task hierarchy from experience. This addresses a long-standing problem in hierarchical reinforcement learning (Barto and Mahadevan, 2003), where most methods take as input a hand-designed hierarchy. By viewing the task hierarchy as a probabilistic "action grammar," we effectively recast the hierarchy discovery problem as grammar induction. We capitalize on recent developments in computational linguistics and Bayesian nonparametrics to specify priors over grammars that express policy fragments—paths through the parse tree representing reusable compositions of sub-tasks.

As an additional contribution, we sharply define what a policy prior *means*, and define a principled likelihood function that explicitly states what it means for a policy to be optimal, accounting for uncertainty in policy evaluation. This is contrast to other attempts to define likelihoods, which are often ad hoc terms like $p(\pi) \propto \exp\{V(\pi)\}$.

In Section 2 we discuss the relationship between planning and inference, and in Section 3 we define our likelihood. Section 4 presents the meat of our compositional policy priors using fragment grammars and Section 5 presents our inference methods. Section 7 discusses action selection, and Section 8 presents our experimental results. Section 9 discusses related work and Section 10 concludes.

2   Planning as inference

We begin by discussing the relationship between planning and inference. We assume that we are an agent in some domain, and must select a *policy,* or mapping from states to

actions, in order to maximize some measure of reward.

Formally, let $\pi \in \Pi$ denote an open-loop policy, defined as a sequence of T actions: $\pi = a_1 \cdots a_T$, where $\mathcal{A}$ is the set of all possible actions. The *sample average reward* under policy π is defined as:

$$R(\pi) = \frac{1}{T} \sum_{t=1}^{T} r_t, \tag{1}$$

where r_t is the reward observed at time t. Because each r_t is a random variable, so is $R(\pi)$; we say that $R(\pi) \sim \mu(\pi)$, where μ is some distribution. Note that we do not make Markovian assumptions about the sequence of observations. A policy is average-reward optimal if it maximizes the *expected average reward* $V(\pi) = \mathbb{E}_\mu[R(\pi)]$:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \, V(\pi) \tag{2}$$

We shall refer to $V(\pi)$ as the *value* of the policy.[1] We shall assume that the underlying value function is latent, though the agent has access to an estimate of $V(\pi)$ for any policy π. In our experiments, we use rollouts with an approximate forward model to construct a Monte Carlo estimator (Bertsekas, 1997). Importantly, we do not expect this estimator to be perfectly accurate, and in Section 3 we show how one can explicitly model uncertainty in the value estimates.

Our goal is to infer the posterior probability that some new policy π' is the optimal policy π^* given "data" $\mathbf{v} = \{V(\pi_1), \ldots, V(\pi_N)\}$ consisting of past policies:

$$p(\pi' = \pi^* | \mathbf{v}) \propto p(\mathbf{v} | \pi' = \pi^*) p(\pi' = \pi^*), \tag{3}$$

where $p(\mathbf{v} | \pi' = \pi^*)$ is the likelihood and $p(\pi' = \pi^*)$ is a policy prior. The likelihood encodes how consistent observed policy values are with the hypothesis that π' is the optimal policy. Formally, it represents the probability that $x_n = V(\pi') - V(\pi_n)$ is greater than 0 for $n = 1, \ldots, N$, integrating over uncertainty about V.

It is important to note that π^* does not correspond to the *globally* optimal policy. Rather, π^* represents a policy that is superior to all other previously sampled policies. While the goal of performing inference over the globally optimal policy is clearly desirable, it presents formidable computational obstacles. Nonetheless, under certain exploration conditions the policy posterior presented above will eventually converge to the global optimum.

A very general way to derive the likelihood term is to represent a conditional distribution over the random variable x_n given the data and then compute the cumulative probability

---

[1]Note that this definition of value departs from the traditional definition in the RL literature (e.g., Sutton and Barto, 1998), where value is defined over states or state-action pairs. Here, the value of a policy is implicitly an expectation with respect to the distribution over state sequences. Thus, we ignore an explicit representation of state in our formalism.
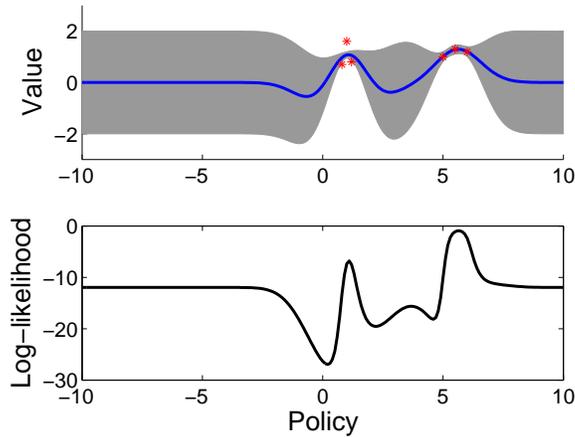
Figure 1: **GP policy likelihood**. (*Top*) Evaluated policies (red crosses) and posterior value function defined over a one-dimensional policy space with a squared-error covariance function. Error bars represent 95 percent confidence intervals. (*Bottom*) Policy log-likelihood.

that x_n is greater than 0 for each value of n. The joint probability that $V(\pi')$ is greater than the value of all the previously evaluated policies is then given by:

$$p(\mathbf{v}|\pi' = \pi^*) = \int_0^\infty p(\mathbf{x}|\mathbf{v})d\mathbf{x} \tag{4}$$

The next question is what form the conditional distribution $p(\mathbf{x}|\mathbf{v})$ takes. We propose one simple and analytically tractable approach in the next section.

3   A Gaussian process policy likelihood

In this section, we show how to derive a policy likelihood of the form (4) by placing a prior over value functions. Each value function induces a different setting of x_n (or a distribution over x_n, if the value function estimator is noisy); integrating out the value function gives us the desired conditional distribution:

$$p(x_n|\mathbf{v}) = \int_V p(x_n|V)p(V|\mathbf{v})dV, \tag{5}$$

where $p(V|\mathbf{v}) \propto p(\mathbf{v}|V)p(V)$ is the value function posterior.

If we place a Gaussian process (GP) prior over value functions, then the posterior over value functions after observing $\mathbf{v}$ is also a Gaussian process (Rasmussen and Williams, 2006). Since any subset of policy values will then be jointly Gaussian-distributed, and the difference between two Gaussian random variables is also a Gaussian-distributed random

4

variable, then $p(\mathbf{x}|\mathbf{v})$ is Gaussian with mean $\boldsymbol{\mu} = m(\pi') - \mathbf{m}(\boldsymbol{\pi})$ and variance $\boldsymbol{\Sigma} = s(\pi')\mathbf{I}_N + \mathbf{S}(\boldsymbol{\pi})$, where:

$$\mathbf{m}(\boldsymbol{\pi}) = \mathbf{K}^\intercal(\mathbf{K} + \tau^2\mathbf{I})^{-1}\mathbf{v} \tag{6}$$

$$\mathbf{S}(\boldsymbol{\pi}) = \mathbf{K} - \mathbf{K}^\intercal(\mathbf{K} + \tau^2\mathbf{I})^{-1}\mathbf{K} \tag{7}$$

and τ^2 is the observation noise variance (i.e., the error in the value function estimator). The function $k : \Pi \times \Pi \to \mathbb{R}$ is a *covariance function* (or *kernel*) that expresses our prior belief about the smoothness of the value function. Here we use the notation $\mathbf{K}$ for the matrix of covariances for each pair of evaluated policies, such that $\mathbf{K}_{ij} = k(\pi_i, \pi_j)$. The vector $\mathbf{k}$ represents the covariances between the test policy and the N evaluated policies.

Given that $p(\mathbf{x}|\mathbf{v})$ is a Gaussian distribution, it is now straightforward to calculate the policy likelihood:

$$p(\mathbf{v}|\pi' = \pi^*) = 1 - \Phi(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{8}$$

where Φ is the multivariate Gaussian CDF.[2] Martinez-Cantin et al. (2009) have investigated a similar idea (though without policy priors), where the GP policy likelihood is closely related to what is known as the *expected improvement function* (see also Jones et al., 1998; Locatelli, 1997; Mockus, 1982).

An example of the policy likelihood function is shown in Figure 1, where we have used a one-dimensional, continuous policy space and a squared-error covariance function. Notice that the highest-valued policy does *not* have the highest likelihood. The reason is that it is flanked by low-valued policies, and the GP implements a form of local regularization. Thus, policies will have high likelihood to the extent that nearby policies are highly valued.

Our examples in the next section will focus on discrete, multidimensional (sequential) policies. One suitable covariance function for this policy space is the *sparse spatial sample kernel* (SSSK), which measures the similarity of two policies based on the number of common action subsequences (Kuksa et al., 2008). Let Σ^n denote the set of n-length subsequences and let $\mathcal{D} = \{1, \ldots, d\}$ denote the set of possible distances separating subsequences. The feature mapping $\phi_u(\pi)$ underlying the SSSK counts the number of times feature $u = (a_1, d_1, \ldots, d_{j-1}, a_j)$ occurs in policy π, with $a_i \in \Sigma^n$ and $d_i \in \mathcal{D}$. The SSSK is defined as the scalar product of feature mappings for two policies:

$$k(\pi, \pi') = \sum_u \phi_u(\pi)\phi_u(\pi'). \tag{9}$$

The SSSK matches policies at multiple spatial resolutions, encoding not only the number of matching subsequences, but also their spatial configuration.

---

[2]In practice, we found it more numerically stable to use a diagonal approximation of the full covariance matrix.

4   Compositional policy priors

We now return to the central theme of this work: the definition of compositional policy priors that are capable of capturing abstract, reusable structure in a domain.

The policy prior $p(\pi' = \pi^*)$ encodes the agent's inductive bias that policy π' is optimal. Another way to understand the policy prior is as a generative model for the optimal policy. This prior can be specified in many different ways; for ease of exposition, we shall consider a succession of increasingly sophisticated compositional priors. In what follows, we assume that the action space is discrete and finite: $\mathcal{A} = \{1, \ldots, A\}$.

4.1   Probabilistic context-free grammars

We begin with policy priors that can be expressed as probabilistic context-free grammars (PCFGs). A PCFG policy prior is a tuple $\langle \mathcal{C}, c_0, \mathcal{A}, \mathcal{R}, \theta \rangle$, where

- $\mathcal{C}$ is a finite set of *non-terminal symbols*.

- $c_0 \in \mathcal{C}$ is a distinguished non-terminal called the *start symbol*.

- $\mathcal{A}$ is a finite set of *terminal symbols* (taken here to be the set of actions).

- $\mathcal{R}$ is a finite set of production rules of the form $c \to c'$, where $c \in \mathcal{C}$, $c' \in (\mathcal{C} \cup \mathcal{A})^*$ and the * superscript denotes Kleene closure.

- θ is a collection of multinomial production probabilities, where $\theta_{c \to c'}$ is the probability of using the production $c \to c'$ to expand the non-terminal c. Typically, we place a Dirichlet(α) prior on θ and integrate out θ to obtain a marginal distribution over productions.

A policy is drawn from a PCFG by starting with c_0 and stochastically expanding symbols until the expression consists only of terminal symbols (actions). This generative process assumes that each sub-tree is drawn independently—the process has no memory of previously generated sub-trees. In the next section, we describe a generalization of the PCFG that relaxes this assumption.

4.2   Pitman-Yor adaptor grammars

Adaptor grammars (AGs) (Johnson et al., 2007) are a generalization of PCFGs that allows reuse of generated sub-trees. The motivation for this generalization is the notion that the optimal policy makes use of "macro-actions" that correspond to particular sub-trees.

6

Formally, a Pitman-Yor adaptor grammar is a tuple $\langle \mathcal{P}, \boldsymbol{\gamma}, \boldsymbol{b} \rangle$, where $\mathcal{P}$ is a PCFG and $\boldsymbol{\gamma}$ and $\boldsymbol{b}$ are vectors of parameters for a set of Pitman-Yor processes (PYP; Ishwaran and James, 2003; Pitman, 1995; Pitman and Yor, 1997), one for each non-terminal in $\mathcal{P}$. In an adaptor grammar, a non-terminal c can be expanded in one of two ways:

1. SUB-TREE REUSE (**R**): With probability $\frac{J_z - b_c}{\gamma_c + \sum_z J_z}$ the non-terminal c is expanded to sub-tree z (a tree rooted at c whose yield is a policy), where J_z is the number of times tree z was previously generated. Here $\gamma_c \geq -b_c$ is the *concentration parameter* and $b_c \in [0, 1]$ is the *discount parameter* of the PYP for c.

2. NEW SUB-TREE CREATION (**N**): With probability $\frac{J_z + Y_c b_c}{\gamma_c + \sum_z J_z}$, where Y_c is the number of unique sub-trees rooted at c, c is expanded by first sampling a production rule from $\mathcal{P}$ and, then, recursively expanding that rule's right-hand side. The new tree that results is stored and reused as in **R**.

Adaptor grammars instantiate a "rich get richer" scheme in which macro-actions (sub-trees) that have been used in the past tend to be reused in the future.

4.3   Fragment grammars

One limitation of adaptor grammars is that reuse is restricted to sub-trees with only terminal (action) leaf nodes. More general, partial tree fragments with "variables" representing categories of macro-actions cannot be learned by an adaptor grammar. To address this issue, O'Donnell et al. (2009) introduced *fragment grammars* (FGs), a generalization of adaptor grammars that allows reuse of arbitrary tree fragments (see also O'Donnell, 2011; O'Donnell et al., 2011). The generative process for fragment grammars is identical to that for adaptor grammars with two crucial differences. First, when expanding a non-terminal c according to **R** in Section 4.2, it is possible to reuse a *partial* tree fragment with one or more non-terminals at its yield. These non-terminals are expanded recursively. Second, when constructing a new tree fragment according to **N** in Section 4.2, a non-terminal c was expanded by first sampling a production rule from $\mathcal{P}$, and, then, recursively expanding that rule's right-hand side. In adaptor grammars the result of this computation was stored for later reuse. In fragment grammars, an additional random choice is made for each non-terminal on the right-hand side of the rule sampled from $\mathcal{P}$ which determines whether or not the corresponding sub-tree is stored, or whether a partial tree fragment with a variable at that non-terminal is created.

5   Approximate inference

The policy posterior (Eq. 3) is intractable to compute exactly due to the exponential number of terms in the normalization constant. We therefore resort to a sample-based approximation based on Monte Carlo methods. We begin by decomposing the policy posterior as follows:

$$p(\pi' = \pi^*|\mathbf{v}) \propto \int_\omega p(\pi' = \pi^*|\mathbf{v}, \omega)p(\omega|\boldsymbol{\pi})d\omega, \tag{10}$$

where $\boldsymbol{\pi} = \{\pi_n\}$, ω represents the parameters of the policy prior; we shall refer to ω generically as the *rule set*. For example, in a PCFG, $\omega = \theta$, the production probabilities. In an adaptor grammar, $\omega = \langle \theta, J, Y \rangle$. This decomposition allows us to split the inference problem into two conceptually distinct problems:

1. Infer $p(\omega|\boldsymbol{\pi}) \propto p(\boldsymbol{\pi}|\omega)p(\omega)$. In the case of grammar-based policy priors, this is a standard grammar induction problem, where the corpus of "sentences" is comprised of past policies $\boldsymbol{\pi}$. Note that this sub-problem does not require any knowledge of the policy values.

2. Infer $p(\pi' = \pi^*|\mathbf{v}, \omega) \propto p(\mathbf{v}|\pi' = \pi^*)p(\pi' = \pi^*|\omega)$. This corresponds to the reinforcement learning problem, in which ω parameterizes an inductive bias towards particular policies, balanced against the GP likelihood, which biases the posterior towards high-scoring policies.

Each of these sub-problems is still analytically intractable, but they are amenable to Monte Carlo approximation. We use a Markov Chain Monte Carlo algorithm for inferring $p(\omega|\boldsymbol{\pi})$. This algorithm returns a set of M samples $\omega^{(1:M)}$, distributed approximately according to $p(\omega|\boldsymbol{\pi})$. Due to space constraints, we do not describe this algorithm here (see O'Donnell et al., 2009, for more details). For each sample, we draw a policy $\pi^{(i)}$ from the policy prior $p(\pi' = \pi^*|\omega^{(i)})$, and weight it according to the policy likelihood:

$$w^{(i)} = \frac{p(\mathbf{v}|\pi^{(i)} = \pi^*)}{\sum_{j=1}^{M} p(\mathbf{v}|\pi^{(j)} = \pi^*)}. \tag{11}$$

The weights can be evaluated efficiently using the GP to interpolate values for proposed policies. We then approximate the policy posterior by:

$$p(\pi' = \pi^*|\mathbf{v}) \approx \sum_{i=1}^{M} w^{(i)}\delta[\pi^{(i)}, \pi']. \tag{12}$$

This corresponds to an *importance sampling* scheme, where the proposal distribution is $p(\pi' = \pi^*|\omega^{(i)})$. Generally, the accuracy of importance sampling approximations depend
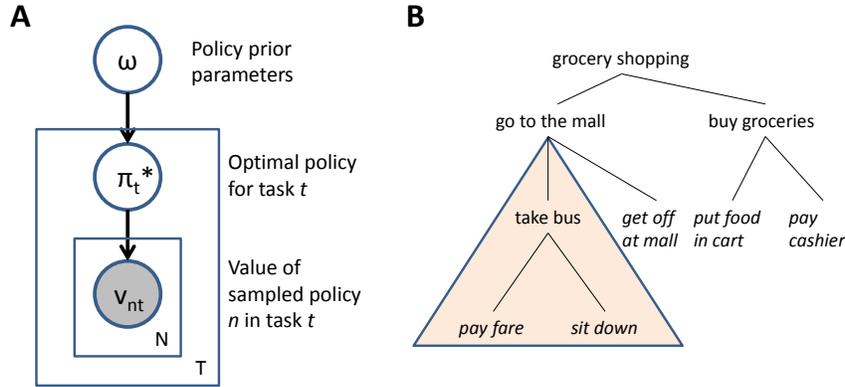
Figure 2: **Model schematic**. (*A*) Graphical model for multi-task learning. T denotes the number of tasks. Sampled policies are not shown. (*B*) Grocery shopping grammar. A fragment that could be reused for going to the bank is highlighted.

on how close the posterior and proposal distributions are (MacKay, 2003). One way to improve the proposal distribution (and also dramatically reduce computation time) is to run MCMC only on high-value past policies. This sparse approximation to the rule prior $p(\omega|\pi)$ is motivated by the fact that the posterior modes will tend to be near the maximum-likelihood (i.e., high-value) policies, and discarding low-value past policies will lead to rules that put higher probability mass on the maximum-likelihood policies. Another motivation for the sparse approximation is as an *empirical Bayes* estimate of the prior, which is sometimes considered more robust to prior misspecification (Berger and Berliner, 1986).

6   Multi-task learning

In the setting where an agent must learn multiple tasks, one way to share knowledge across tasks in the policy prior framework is to assume that the tasks share a common rule set ω, while allowing them to differ in their optimal policies. This corresponds to a hierarchical generative model in which the rule prior $p(\omega)$ generates a rule set, and then each task samples an optimal policy conditional on this rule set. A graphical model of this generative process is shown in Figure 2(A).

Inference in this hierarchical model is realized by aggregating all the policies across tasks into π, and then sampling from $p(\omega|\pi)$ using our MCMC algorithm. This gives us our approximate posterior over rule sets. We then use this posterior as the proposal distribution for the importance sampler described in Section 5, the only difference being that we include *only* policies from the current task in the GP computations. This scheme has the added benefit that the computational expense of the GP computations is reduced.

A simple example of a fragment grammar is shown in Figure 2(B). Here the task is to go grocery shopping, which can be decomposed into "go to the mall" and "buy groceries" actions. The "go to the mall" action can be further decomposed into "take bus" and "get off at mall' actions, and so on. A fragment of this grammar is highlighted that could potentially be reused for a "go to the bank" task. Policy inference takes as data the primitive actions at the leaves of this tree and their associated values, and returns a distribution over optimal policies.

7   Policy selection

We have so far conditioned on a set of policies, but how should an agent select a new policy so as to maximize average reward? A general solution to this problem, in which exploration and exploitation are optimally balanced, is not computationally tractable. However, a very simple approximation, inspired by softmax (Boltzmann) exploration widely used in reinforcement learning (Sutton and Barto, 1998), is to sample from the approximate policy posterior. Using the importance sampling approximation, this corresponds to setting $\pi_{N+1} = \pi^{(i)}$ with probability $w^{(i)}$. Once a policy has been selected, we evaluate it and add this value to $\mathbf{v}$.

This strategy of importance-weighted policy selection has a number of attractive properties:

- The amount of exploration scales with posterior entropy: when the agent has more uncertainty about the optimal policy, it will be more likely to select policies with low predicted value under the GP.

- Under some mild assumptions, it will asymptotically converge to deterministic selection of the optimal policy.

- It requires basically no extra computation, since it reuses the weights computed during inference.

8   Experiments - Increasingly Complex Mazes

To explore the use of compositional policy fragments, we tested our ideas in a family of increasingly complex mazes. Fig. 3 (top left) shows the 11 mazes, which have been constructed such that more complex mazes are composed of bits of easier mazes. Fig. 3 (top right) also shows the base grammar for this domain: the agent's policy is composed of several "steps," which are composed of a direction and a count.

To simplify the experiment and focus on the transfer learning via fragment grammar, no feedback is given in this domain (there are no observations). Instead, policies are open-loop: the agent samples a policy (for example, "move N 5 steps, move S 3 steps, move E 2 steps"), rolls it out, and evaluates the return. The domain is therefore a "blind search" problem, which is difficult when naively done because of the infinite number of possible policies (this arises from the recursive nature of the grammar). The agent receives a reward of -0.1 each time it bumps into a wall, and incurs a terminal reward proportional to the distance between the agent's ending position and the goal. Optimal policies in each domain therefore have the value of zero.

Fig. 3 (bottom left) shows the results of transfer and inference. Three lines are shown. The red line ("baseline") represents the distribution of values of policies that are randomly sampled from the base grammar (ie, the prior). The blue line shows the distribution of the value of policies conditioned on transferring the optimal policies of previous domains. We transferred optimal policies to avoid issues with exploration/exploitation, as well as compounding errors. The green line will be explained later.

In the 'A' plot, we see that the blue and red lines exactly overlap, because there is nothing to transfer. In the 'B' plot, we see some slight negative transfer: the optimal policy is to move south 5 steps, but the agent attempts to transfer its knowledge gleaned from the optimal policy of the first task (which is to move east 5 steps). In the 'C' plot, we see the first example of positive transfer: the agent has learned that steps of length 5 are salient, and we see that the distribution of values has shifted significantly in the positive direction. In all subsequent plots, we see more positive transfer. We also see multi-modality in the distributions, because the agent learns that all useful movements are multiples of 5.

To examine the results more closely, Fig. 3 (bottom right) shows some of the learned policy fragments after experiencing mazes 'A'-'J', upon entering domain 'K'. The agent has learned a variety of abstract patterns in this family of mazes: 1) that the directions 'S' and 'E' are most salient; 2) that all movement occurs in units of 5 steps; and 3) that optimal policies often have an even number of steps in them. This abstract knowledge serves the agent well: the 'K' plot shows that the agent has a much higher likelihood of sampling high-value policies than the uninformed baseline; there is even some possibility of sampling an optimal policy from the start.

The green line in each plot represents a different experiment. Instead of transferring policy knowledge from easy mazes to hard mazes, we asked: what would happen if we ran the process in reverse, and tried to transfer from hard mazes to easy ones? Thus, the 'K' maze is the first domain tried; in the 'K' plot, the green and red lines overlap, because there is nothing to transfer. In the second maze ('J') we see negative transfer, just like the second domain of the forward transfer case. We see that this idea is generally ineffective; while some information is transferred, performance on the difficult mazes (such as I and J) is worse when transferring in the reverse direction rather than the forward direction. (We see some positive transfer to the easy mazes – but this is arguably less important since
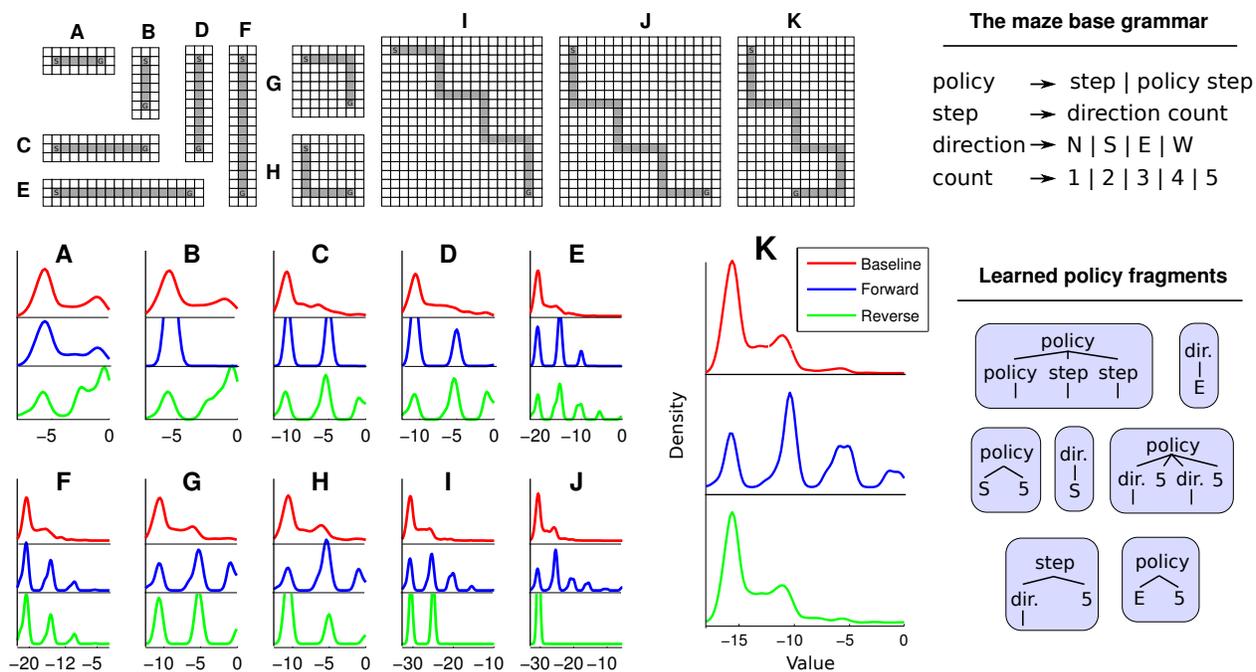
Figure 3: **Results on the maze experiment**. (*Top left*) Eleven mazes of increasing complexity. More difficult mazes are composed of pieces of previous mazes. (*Top right*) The base maze grammar. (*Bottom left*) Transfer results. Shown are distributions over policy values for three conditions: baseline (no transfer), forward transfer (policy fragments are transferred from easier mazes to harder mazes), and reverse transfer (policy fragments are transferred from harder mazes to easier ones). (*Bottom right*) Some representative policy fragments upon entering maze 'K', before seeing any data. The fragments are learned from optimal policies of previous domains.

they're so easy!).

This validates the intuition that transfer can be used to gradually shape an agents knowledge from simple policies to complex ones.

9   Related work

Our work intersects with two trends in the RL and control literature. First, our planning-as-inference approach is related to a variety of recent ideas attempting to recast the planning problem in terms of probabilistic inference (e.g., Botvinick and An, 2009; Hoffman et al., 2009; Theodorou et al., 2010; Toussaint and Storkey, 2006; Vlassis and Toussaint, 2009), in addition to the Bayesian policy search papers cited earlier. While these papers exploit a formal equivalence between inference and planning, they do not harness the full power of the Bayesian framework—namely, the ability to coherently incorporate prior knowledge about good policies into the inference problem. Mathematically, it was previously unclear whether policy priors even have a sensible probabilistic interpretation, since the formal equivalence between inference and planning typically relies on using a uniform prior (as in Toussaint and Storkey, 2006) or an arbitrary one, where it has no effect on the final solution (as in Botvinick and An, 2009). Our framework is a first step towards providing policy priors with a better probabilistic foundation.

Policy priors have previously been used by Doshi-Velez et al. (2010) to incorporate information from expert demonstration. In that work, the probabilistic semantics of policy priors were defined in terms of beliefs about expert policies. Here, we develop a more general foundation, wherein policy priors arise from a generative model of optimal policies. These can, in principle, include beliefs about expert policies, along with many other soruces of information.

In Wilson et al. (2010), prior knowledge was built into the mean-function of the GP; in contrast, we assumed a zero-valued mean function. It is difficult to see how structured prior knowledge can be incorporated in the formulation of Wilson et al. (2010), since their mean function is a simple Monte Carlo estimator of past reward. Our goal was to develop parameterized policy priors with a rich internal structure that can express, for example, hierarchical organization.

This leads us to the second trend intersected by our work: hierarchical RL (Barto and Mahadevan, 2003), in which agents are developed that can plan at multiple levels of temporal abstraction. The fragment grammar policy prior can be seen as a realization of hierarchical RL within the planning-as-inference framework (see also Brochu et al., 2010). Each fragment represents an abstract sub-task—a (possibly partial) path down the parse tree. The policy posterior learns to adaptively compose these fragments into structured policies. Most interestingly, our model also *learns* the fragments, without the need to hand-design

the task hierarchy, as required by most approaches to hierarchical RL (e.g., Dietterich, 2000; Sutton et al., 1999). However, it remains to be seen whether the fragments learned by our model are generally useful. The policy prior expresses assumptions about the latent policy structure; hence, the learned fragments will tend to be useful to the extent that these assumptions are true.

10 Conclusions and future work

We have presented a Bayesian policy search framework for compositional policies that incorporates inductive biases in the form of policy priors. It is important to emphasize that this is a modeling *framework*, rather than a particular model: Each policy prior makes structural assumptions that may be suitable for some domains but not others. We described several families of policy priors, the most sophisticated of which (fragment grammars) can capture hierarchically organized, reusable policy fragments. Since optimal policies in many domains appear amenable to these structural assumptions (Barto and Mahadevan, 2003), we believe that fragment grammars may provide a powerful inductive bias for learning in such domains.

We have illustrated our method on a family of increasingly complex maze tasks, and shown that fragment grammars are capable of capturing abstract, reusable bits of policy knowledge; this provides improved inductive biases upon entering new tasks.

Our methods can be improved in several ways. First, our multi-task model imposes a rather blunt form of sharing across tasks, where a single rule set is shared by all tasks, which are allowed to vary in their optimal policies and value functions conditional on this rule set. A more flexible model would allow inter-task variability in the rule set itself. Nonetheless, the form of multi-task sharing that we adopted offers certain computational advantages; in particular, the GP computations can be performed for each task separately.

Another direction for improvement is algorithmic. Our Monte Carlo methods do not scale well with problem size. For this reason, alternative inference algorithms may desirable. Efficient variational algorithms have been developed for adaptor grammars (Cohen et al., 2010), and these may be extended to fragment grammars for use in our framework.

References

Barto, A. and Mahadevan, S. (2003). Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, 13(4):341–379.

Berger, J. and Berliner, L. (1986). Robust Bayes and empirical Bayes analysis with e-contaminated priors. *Annals of Statistics*, 14(2):461–486.

Bertsekas, D. (1997). Differential training of rollout policies. *Proc. of the 35th Allerton Conference on Communication, Control, and Computing*.

Botvinick, M. and An, J. (2009). Goal-directed decision making in prefrontal cortex: a computational framework. *Advances in Neural Information Processing Systems)*, pages 169–176.

Brochu, E., Cora, V., and de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Technical Report TR-2009-023. University of British Columbia, Department of Computer Science.*

Cohen, S., Blei, D., and Smith, N. (2010). Variational inference for adaptor grammars. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572. Association for Computational Linguistics.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Doshi-Velez, F., Wingate, D., Roy, N., and Tenenbaum, J. (2010). Nonparametric Bayesian Policy Priors for Reinforcement Learning. *Advances in Neural Information Processing Systems*.

Hoffman, M., de Freitas, N., Doucet, A., and Peters, J. (2009). An expectation maximization algorithm for continuous markov decision processes with arbitrary rewards. In *Twelfth International Conference on Artificial Intelligence and Statistics*. Citeseer.

Ishwaran, H. and James, L. (2003). Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13(4):1211–1236.

Johnson, M., Griffiths, T., and Goldwater, S. (2007). Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. *Advances in Neural Information Processing Systems*, 19:641.

Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Kuksa, P., Huang, P., and Pavlovic, V. (2008). Fast protein homology and fold detection with sparse spatial sample kernels. In *International Conference on Pattern Recognition*, pages 1–4. IEEE.

Kushner, H. (1963). A new method of locating the maximum point of an arbitrary multi-peak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106.

15

Locatelli, M. (1997). Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10(1):57–76.

MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J., and Doucet, A. (2009). A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103.

Mockus, J. (1982). The bayesian approach to global optimization. *System Modeling and Optimization*, pages 473–481.

O'Donnell, T., Tenenbaum, J., and Goodman, N. (2009). Fragment Grammars: Exploring Computation and Reuse in Language. Technical report, MIT Computer Science and Artificial Intelligence Laboratory Technical Report Series, MIT-CSAIL-TR-2009-013.

O'Donnell, T. J. (2011). *Productivity and Reuse in Language*. PhD thesis, Harvard University.

O'Donnell, T. J., Snedeker, J., Tenenbaum, J. B., and Goodman, N. D. (2011). Productivity and reuse in language. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*.

Pitman, J. (1995). Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102(2):145–158.

Pitman, J. and Yor, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211.

Theodorou, E., Buchli, J., and Schaal, S. (2010). Learning policy improvements with path integrals. In *International Conference on Artificial Intelligence and Statistics*.

Toussaint, M. and Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *Proceedings of the 23rd International Conference on Machine Learning,* pages 945–952. ACM.

Vlassis, N. and Toussaint, M. (2009). Model-free reinforcement learning as mixture learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1081–1088. ACM.

Wilson, A., Fern, A., and Tadepalli, P. (2010). Incorporating domain models into Bayesian optimization for RL. *Machine Learning and Knowledge Discovery in Databases*, pages 467–482.