Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations

Finale Doshi-Velez*
Harvard University
Cambridge, MA 02138
finale@seas.harvard.edu

George Konidaris*
Duke University
Durham, NC 27708
gdk@cs.duke.edu

Abstract

Control applications often feature tasks with similar, but not identical, dynamics. We introduce the Hidden Parameter Markov Decision Process (HiP-MDP), a framework that parametrizes a family of related dynamical systems with a low-dimensional set of latent factors, and introduce a semiparametric regression approach for learning its structure from data. We show that a learned HiP-MDP rapidly identifies the dynamics of new task instances in several settings, flexibly adapting to task variation.

Many control applications involve repeated encounters with domains that have similar, but not identical, dynamics. An agent that swings bats may encounter several bats with different weights or lengths, while an agent that manipulates cups may encounter cups with different amounts of liquid. An agent that drives cars may encounter many different cars, each with unique handling characteristics.

In all of these scenarios, it makes little sense of the agent to start afresh when it encounters a new bat, a new cup, or a new car. Exposure to a variety of related domains should correspond to faster and more reliable adaptation to a new instance of the same type of domain, via transfer learning. If an agent has already swung several bats, for example, we would hope that it could easily learn to swing a new bat. Why? Like many domains, swinging a bat has a low-dimensional representation that affects its dynamics in structured ways. The agent's prior experience should allow it to both learn *how* to model related instances of a domain—such as via the bat's length, which smoothly changes in the bat's dynamics—and *what* specific model parameters (e.g., lengths) are likely.

We introduce the Hidden Parameter Markov Decision Process (HiP-MDP) as a formalization of these types of domains, with two important features. First, we posit that there exist a bounded number of latent parameters that, if known, would fully specify the dynamics of each individual task. Second, we assume that the parameter values remain fixed for a task's duration (e.g. the bat's length will not change during a swing), and the agent will know when a change has occurred (e.g. getting a new bat).

The HiP-MDP parameters encode the minimum learning required for the agent to adapt to a new domain instance: they are a sufficient statistic for that instance's dynamics. Given a generative model of how the latent parameters affect domain dynamics, an agent could rapidly identify the dynamics of a particular domain instance by maintaining and updating its distribution (or *belief*) over the latent parameters. Instead of learning a new policy for each domain instance, it could synthesize a parametrized control policy [Kober *et al.*, 2012; da Silva *et al.*, 2012] based on a point estimate of the parameter values, or plan in the belief space over its parameters [Poupart *et al.*, 2006; Silver and Veness, 2010; Ross *et al.*, 2008; Guez *et al.*, 2012; Bai *et al.*, 2013].

We present a method for learning HiP-MDPs from data. Our generative model uses Indian Buffet Processes [Griffiths and Ghahramani, 2011] to model what latent parameters are relevant for what state space variables and Gaussian processes [Rasmussen and Williams, 2005] to model the dynamics functions. We do not require knowledge of a system's kinematics equations, nor must we specify the number of latent parameters in advance. Our HiP-MDP model rapidly identifies dynamics and near-optimal control policies for novel instances of the acrobot [Sutton and Barto, 1998] and bicycle [Randlov and Alstrom, 1998] domains.

1 Background

Bayesian Reinforcement Learning The reinforcement learning (RL) setting consists of a series of interactions between an agent and an environment. From some state s, the agent chooses an action a which transitions it to a new state s' and provides reward r. Its goal is to maximize its expected sum of rewards, $E[\sum_t \gamma^t r_t]$, where $\gamma \in [0, 1)$ is a discount factor that weighs the relative importance of near-term and long-term rewards. This series of interactions can be modeled as a Markov Decision Process (MDP), a 5-tuple $\{S, A, T, R, \gamma\}$ where S and A are sets of states s and actions a, the transition function $T(s'|s, a)$ gives the probability of the next state being s' after performing action a in state s, and the reward function $R(s, a)$ gives the reward r for performing action a in state s. We refer to the transition function $T(s'|s, a)$ as the *dynamics* of a system.

The transition function T or the reward function R must be learned from experience. Bayesian approaches to reinforcement learning [Poupart *et al.*, 2006; Ross *et al.*, 2008; Silver and Veness, 2010] place a prior over the transition

*Both authors are primary authors.

function T (and sometimes also R), and refine this prior with experience. Thus, the problem of learning an unknown MDP is transformed into a problem of planning in a known *partially*-observable Markov Decision Process (POMDP). A POMDP [Kaelbling *et al.*, 1998] consists of a 7-tuple $\{Y, A, O, \tau, \Omega, R, \gamma\}$, where Y, A, and O are sets of states y, actions a, and observations o; $\tau(y'|y, a)$ and $R(y, a)$ are the transition and reward functions; and the observation function $\Omega(o|y', a)$ is the probability of receiving an observation o when taking action a to state y'. Bayesian RL learns an MDP by planning in a POMDP with states $y_t = \{s_t, T\}$: the fully-observed "world-state" s_t and the hidden dynamics T.

However, despite recet advances [Silver and Veness, 2010; Kurniawati *et al.*, 2008; Guez *et al.*, 2012], solving POMDPs in high-dimensional, continuous state spaces remains challenging. Our approach simplifies the Bayesian RL challenge by using instances of related tasks to find a low-dimensional representation of the transition function T.

Indian Buffet Processes and Gaussian Processes Our specific instantiation of the HiP-MDP uses two models from Bayesian nonparametric statistics. The first is the Indian Buffet Process (IBP). The IBP is a prior on 0-1 matrices $M(n, k)$ with a potentially unbounded number of columns k. To generate samples from the prior, we first use a Beta process to assign a probability p_k to each column k such that $\sum_k p_k$ is bounded. Then, each entry $M(n, k)$ is set to 1 independently with probability p_k. We use the IBP as a prior on which latent parameters are relevant for predicting each state transition.

The second model we use is the Gaussian Process (GP). A GP is a prior over continuous functions $y = f(x)$ where the prior probability of outputs $\{y_1, ..., y_t\}$ given a set of inputs $\{x_1, ..., x_t\}$ is a multivariate Gaussian $N(m, K)$, where $m(x_i)$ is the mean function of the Gaussian process and the covariance matrix has elements $K(x_i, x_j)$ for some positive definite kernel function K. We will use Gaussian processes as priors over the basis functions for our transition functions.

2 Hidden Parameter Markov Decision Processes

We focus on learning the dynamics T and assume that the reward function $r = R(s, a)$ is fixed across all instances. Let b denote each instance of a domain. The Hidden Parameter Markov Decision Process (HiP-MDP) posits that the variation in the dynamics of different instances can be captured through a set of hidden parameters θ_b.

A HiP-MDP is described by a tuple: $\{S, A, \Theta, T, R, \gamma, P_\Theta\}$, where S and A are the sets of states s and actions a, and $R(s, a)$ is the reward function. The dynamics T for each instance b depends on the value of the hidden parameters θ_b: $T(s'|s, \theta_b, a)$. We denote the set of all possible parameters as Θ and let $P_\Theta(\theta)$ be the prior over these parameters. Thus, a HiP-MDP describes a *class* of tasks; a particular instance b of that class is obtained by fixing the parameter vector $\theta_b \in \Theta$.

As pointed out by Bai *et al.* [2013] in a similar setting, we can consider the HiP-MDP a type of POMDP where the hidden state is the parameter vector θ_b. However, a HiP-MDP makes two assumptions that are stronger than those of a POMDP. First, each *instance* of a HiP-MDP is an MDP—conditioned on θ_b, there is no hidden state. Thus, we could always learn to solve each HiP-MDP instance as its own distinct MDP. Second, the parameter vector θ_b is fixed for the duration of the task, and thus the hidden state has no dynamics. This assumption considerably simplifies the procedure for inferring the hidden parametrization.

In special cases, we may be able to derive analytic expressions for how θ_b affects the dynamics $T(s'|s, \theta_b, a)$: for example, in a manipulation domain we might be able to derive an equation for how the cup will respond to a force given a particular mass of liquid. However, in most situations, the simplifications required to derive these analytical forms for the dynamics will be brittle at best. The IBP-GP prior for the HiP-MDP, presented next, describes a semiparametric approach for modeling the dynamics that places few assumptions on the form of the transition function $T(s'|s, \theta_b, a)$ while still maintaining computational tractability.

3 The IBP-GP HiP-MDP Model

Let the state s be some d-dimensional vector in $\mathbb{R}^d$. We propose a transition model T of the form:

$$\begin{aligned}(s'_d - s_d) &\sim \textstyle\sum_k^K z_{kad} w_{kb} f_{kad}(s) + \epsilon \\ \epsilon &\sim N(0, \sigma^2_{nad}),\end{aligned}$$

that is, we model the transition distribution as the difference between the current and next state. The transition distribution has two parts (as also illustrated in figure 1): one that is shared among all instances and one that is specific to the particular instance b. The functions $f_{kad}(s)$ can be thought of as a basis from which we will construct the transition function, and the filter parameters $z_{kad} \in \{0, 1\}$ denote whether the k^{th} latent parameter is relevant for predicting dimension d under action a—effectively switching f_{kad} on or off. These global basis functions $z_{kad} f_{kad}(s)$ are multiplied by task-instance-specific weights w_{kb} (e.g., the effect of torque on velocity depends on the weight of the bat). The task instance dynamics are therefore obtained by setting w_{kb}, the only elements of the model that vary across instances.

The generative process for the hidden variables w_{kb}, z_{kad}, and f_{kad} is given by

$$\begin{aligned}z_{kad} &\sim \text{IBP}(\alpha) \text{ for } k > 1 \\ f_{kad} &\sim \text{GP}(\psi) \\ \mu_{w_k} &\sim N(0, \sigma^2_{w_0}) \\ w_{kb} &\sim N(\mu_{w_k}, \sigma^2_w) \text{ for } k > 1,\end{aligned}$$

where α and ψ are the hyper-parameters of the IBP and GP. We set $w_{1b}, z_{1ad} = 1$ to fix the scale of the latent parameters and have the basis function f_{1ad} be the mean dynamics.

The IBP-GP prior encodes the assumption that we have an infinite number of possible basis functions f_{kad} that we could use to construct the transition functions, while using an IBP prior on the filter parameters z_{kad} implies that we expect a few latent factors to be relevant for making most predictions. That is, given a finite number of instances, we will only need

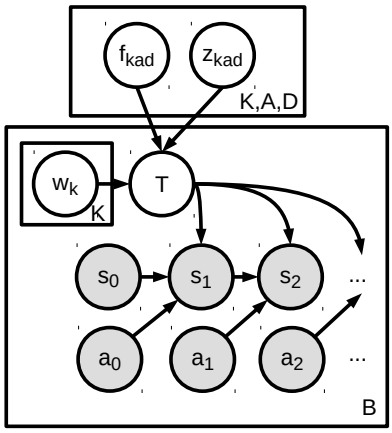

Figure 1: The graphical model for the IBP-GP HiP-MDP. The transitions between states depend on a set of batch-specific weights w_{kb} and global parameters f_{kad}, z_{kad}.

a finite number of basis functions to construct the dynamics. Specifically, we cannot have seen more than B different transition functions in a batch of B instances. Thus, when making predictions about a specific state dimension d given action a, only a few of these infinite possible latent parameters will be relevant—and we only need to infer weights w_{kb} corresponding to situations where z_{kad} is nonzero. Moreover, additional prediction tasks—such as a new action, or a new dimension to the state space—can be incorporated in a statistically consistent manner. As a regression model, IBP-GP model is an infinite version of the Semiparametric Latent Factor Model [Teh *et al.*, 2005].

4 Inference in the IBP-GP HiP-MDP

We focus on scenarios in which the agent is given a large amount of *batch* observational data from several domain instances (perhaps solved as independent MDPs) and tasked with quickly performing well on new instances. Our batch inference procedure uses the observational data to fit the filter parameters z_{kad} and basis functions f_{kad}, which are independent of any particular instance, and compute a posterior over the weights w_{kb}, which depend on each instance. These settings of z_{kad}, f_{kad}, and $P(w_{kb})$ will used to infer the instance-specific weights w_{kb} efficiently in the online setting when given a new instance.

4.1 Batch Inference

Given a batch of data containing multiple instances, we infer the filter parameters z_{kad}, the weights w_{kb}, and an approximation to the basis functions $f_{kad}(S^*)$ using a blocked Gibbs sampler.

Representing and Resampling f_{kad} The posterior over the weights w_{kb} is Gaussian given the filter settings z_{kad} and means μ_{w_k}; the basis functions f_{kad} can be marginalized out. Thus, in theory, it is not necessary to instantiate the basis functions f_{kad}.

However, marginalization over the basis functions f_{kad} requires computing inverses of matrices of size $N = \sum_b N_b$, where N_b is the number of data collected in instance b. To avoid this computational cost, we choose to represent each function f_{kad} by a set of $(s^*, f_{kad}(s^*))$ pairs for states s^* in a set of support points S^*. Various optimization procedures exist for choosing the support points [Snelson and Ghahramani, 2005; Teh *et al.*, 2005]; we found that iteratively choosing support points from existing points to minimize the maximum reconstruction error within each batch was best for a setting in which a few large errors can result in poor performance.

Given a set of tuples (s, a, s', r) from a task instance b, we first created tuples $(s^*, a, \Delta_b(s^*))$ for all $s^* \in S^*$ and all actions a, all dimensions d, and all instances b. The tuple $(s_d*, a, \Delta_b(s_d*))$ can be predicted based on all other data available for that action a, dimension d pair using standard Gaussian process prediction:

$$E[\Delta(s_d*)] = K_{s^* S_{ab}}(K_{S_{ab}S_{ab}} + \sigma^2_{nad} I)^{-1} \Delta_b(S_{abd}),$$

where S_{ab} is the collection of tuples (s, a, s', r) with action a from instance b, $K_{s^* S_{ab}}$ is the vector $K(s^*, s)$ for every $s \in S_{ab}$, $K_{S_{ab}S_{ab}}$ is the matrix $K(s, s)$, and $\Delta_b(S_{abd})$ is the vector of differences $s'_d - s_d$. This procedure was repeated for every task instance b.

Now, we can proceed to sample new values for the basis function f_{kad} at the chosen support points S^*. Given z_{kad} and w_{kb}, the posterior over the outputs $f_{kad}(S^*)$ is Gaussian. Let $f_{ad}(S^*)$ be a column vector of concatenated $f_{kad}(S^*)$ vectors, and let $\Delta(S^*)$ be a column vector of concatenated $\Delta_b(S^*)$ vectors. The mean and covariance of $f_{ad}(S^*)$ is given by

$$\begin{aligned} \text{cov}(f_{ad}(S^*)) &= \sigma^2_{nad}(W^T W + \sigma^2_{nad}\mathbf{K}^{-1}_{S^*S^*})^{-1} \\ E[f_{ad}(S^*)] &= \frac{1}{\sigma^2_{nad}}\text{cov}(f_{ad}(S^*))^{-1} W^T * \Delta(S^*) \end{aligned}$$

where

$$\begin{aligned} \mathbf{K} &= I_k \otimes K_{S^*S^*} \\ W &= w_b(z_{kad}) \otimes I_{|S^*|}, \end{aligned}$$

where $\otimes$ is the Kronecker product, $K_{S^*S^*}$ is the matrix $K(s^*, s^*)$ for every $s^* \in S^*$, and we write $w_b(z_{kad})$ to be a $B \times k'$ matrix of elements w_{kb} such that $z_{kad} = 1$. We repeat this process for each action a and each dimension d.

The update equations for z_{kad} and w_{kb} follow directly from the properties of Gaussian distributions and Indian Buffet Processes.

Resampling w_{kb} Given z_{kad} and $f_{kad}(S^*)$, the posterior over w_{kb} is also Gaussian. For each instance b, the mean and covariance of $w_b(z_{kad})$ (that is, the weights not set to zero) are given by:

$$\begin{aligned} \text{cov}(w_{kb}) &= \sigma^2_n(F_b^T F_b + I_{k-1}\frac{\sigma^2_n}{\sigma^2_w})^{-1} \\ E[w_{kb}] &= \text{cov}(w_{kb})^{-1}(\frac{\mu_{w_k}}{\sigma^2_w} + \frac{1}{\sigma^2_n}F_b^T) \\ &\quad \cdot(\Delta_b(S^*) - F_{1b}(S^*)), \end{aligned}$$

for $k > 1$, where F_b is matrix with $k' - 1$ columns concatenating values for f_{kad} for all actions a and all dimensions d and excluding $k = 1$, and $\Delta_b(S^*)$ is a column vector concatenating the differences for all actions a and all dimensions d.

Resampling z_{kad} To sample z_{kad} for an already-initialized feature k, we note that the likelihood of the model given z, w, and f with some $z_{k'ad} = 0$ is Gaussian with mean and variance

$$E[\Delta s^*] = \sum_{k \neq k'} z_{kad} w_{kb} f_{kad}(s^*) \quad (1)$$

$$\text{cov}(\Delta s^*) = \sigma^2_{nad}. \quad (2)$$

If $z_{kad} = 1$, then the likelihood is again Gaussian with the same mean (here we assume that the GP prior on f is zero-mean) but with covariance

$$\text{cov}(\Delta s^*) = w_{k'} w_{k'}^T \otimes K_{S^* S^*} + \sigma^2_{nad} I, \quad (3)$$

where $w_{k'}$ is a vector of latent parameter values for all instances b. We combine these likelihoods with the prior to sample z_{kad}.

Initializing a new latent parameter k'—that is, a k' for which we do not already have values for the weights $w_{k'b}$—involves computing the marginal likelihood $P(\Delta(S^*)|z_{k'ad}, w_{kb}, f_{kad})$, which is intractable. We approximate the likelihood by sampling N_w sets of new weights $w_{k'b}$. Given values for the weights $w_{k'b}$ for each instance b, we can compute the likelihood with the new basis function $f_{k'ad}$ marginalized out. We average these likelihoods to estimate the marginal likelihood of $z_{k'ad} = 1$ for the new k'. (For the $z_{k'ad} = 0$ case, we can just use the variance from the prior.) If we do set $z_{k'ad} = 1$ for the new k', then we can sample a set of weights $w_{k'b}$ from our N_w samples based on their importance weight (marginal likelihood).

Finally, given the values of the weights w_{kb} for a set of instances b, we can update the posterior over μ_{w_k} with a standard conjugate Gaussian update:

$$\text{var}(\mu_{w_k}) = (\frac{1}{\sigma^2_{w_0}} + \frac{B}{\sigma^2_w})^{-1}$$

$$E[\mu_{w_k}] = \frac{\sum_b w_{kb}}{\sigma^2_w} \text{var}(\mu_{w_k})^{-1}.$$

We use this posterior over the weight means μ_{w_k} when we encounter a new instance b'.

4.2 Online Filtering

Given values for the filter parameters z_{kad} and the basis functions f_{kad}, the posterior on the weights w_{kb} is Gaussian. Thus, we can write the parametrized belief $b_t(w_{kb})$ at time t with (h_w, P_w), where P_w is the inverse covariance Σ_w^{-1} and h_w is the information mean $\mu_w P_w$. Then the update given an experience tuple (s, a, s', r) is given by

$$h_w(t+1) = h_w(t) + F_a^T \Sigma_{na}^{-1} \Delta(s)$$

$$P_w(t+1) = P_w(t) + F_a^T \Sigma_{na}^{-1} F_a,$$

where F_a is a $d \times k$ matrix of basis values $f_{kad}(s)$, Σ_{na} is a $d \times d$ noise matrix with σ^2_{nad} on the diagonal (note that therefore the inverse Σ_{na}^{-1} is trivially computed), and $\Delta(s)$ is a d-dimensional vector of $s'_d - s_d$. If updates are performed only every n time steps, we can simply extend F, Σ, and $\Delta(s)$ to be $nd \times k$, $nd \times nd$, and nd respectively.

Computing F_a requires computing the values of the basis functions at the point s. Since we only have the values computed at pseudo-input points $s^* \in S^*$, we use standard GP prediction to interpolate the value for this new point:

$$E[f_{kad}(s)] = K_{sS^*}(K_{S^*S^*} + \sigma^2_{nad} I)^{-1} f_{kad}(S^*),$$

where K_{sS^*} is the vector $K(s, s^*)$ for every $s^* \in S^*$, $K_{S^*S^*}$ is the matrix $K(s^*, s^*)$, and $f_{kad}(S^*)$ is the vector $f_{kad}(s^*)$. Using only the mean value $f_{kad}(s^*)$ ignores the uncertainty in the basis function f_{kad}. While incorporating this variance is mathematically straight-forward—all updates remain Gaussian—it adds additional computation to the online calculation, and we found that using the means already provided significant gains in learning in practice.

5 Results

In this section, we describe results on three benchmark problems: cartpole, acrobot [Sutton and Barto, 1998], and bicycle [Randlov and Alstrom, 1998]. Cartpole has a relatively simple policy; we use it to visualize how HiP-MDPs compress the dynamics across different parameter settings into a latent space. Acrobot and bicycle are challenging domains when the agent must rapidly learn a control strategy for a new set of parameters. All of our tests use an anisotropic squared-exponential kernel with length and scale parameters approximated for each action a and state dimension d.

5.1 Cartpole

In the cartpole domain, an agent must apply forces either to the left or right of a cart to keep a pole balanced on top of it. It has a four-dimensional state space $s = \{x, \dot{x}, \theta, \dot{\theta}\}$ consisting of the cart's position x, the cart's velocity $\dot{x}$, the pole's angle θ, and the pole's angular velocity $\dot{\theta}$. At each time step of length τ, the system evolves according to:

$$\begin{aligned} x_{t+1} &= x_t + \tau \dot{x}_t \\ \dot{x}_{t+1} &= \dot{x}_t + \tau(v - ml\ddot{\theta}\cos\theta/M) \\ \theta_{t+1} &= \theta_t + \tau\dot{\theta}_t \\ \dot{\theta}_{t+1} &= \dot{\theta}_t + \tau\ddot{\theta}, \end{aligned} \quad (4)$$

where $v = \frac{f + ml\dot{\theta}_t^2 \sin\theta}{M}$, $\ddot{\theta} = \frac{(g\sin\theta - v\cos\theta)}{(l(\frac{4}{3} - m\cos\theta^2/M)}$, f is the applied force, g is gravity, M is the mass of both the cart and the pole, and m and l are the mass and length of the pole.

We varied the pole mass m and the pole length l. In each (m, l) setting, we ran Sarsa [Sutton and Barto, 1998] (using a 3rd order Fourier basis [Konidaris *et al.*, 2011]) for 5 repetitions of 30 episodes, where each episode was run for 300 steps or until the pole fell down. Data from seven (m, l) settings { (.1,.4) , (.3,.4) , (.15,.45) , (.2,.55) , (.25,.5) , (.3,.4) , (.3,.6) } were reduced to 750 support points and then the

	Batch + Train	Batch Only	Train Only	IBP-GP HIP-MDP
$\dot{x}$	4.8e-06 (1.7e-08)	4.8e-06 (1.7e-08)	9.1e-05 (2.3e-07)	**4.7e-06 (1.7e-08)**
$\ddot{x}$	1.1e-07 (1.0e-08)	1.1e-07 (1.1e-08)	7.6e-07 (1.4e-07)	**3.3e-08 (2.9e-09)**
$\dot{\theta}$	1.1e-03 (3.2e-06)	1.1e-03 (3.2e-06)	3.4e-02 (1.4e-04)	**1.0e-03 (3.2e-06)**
$\ddot{\theta}$	1.5e-05 (1.5e-06)	1.5e-05 (1.6e-06)	3.7e-05 (2.8e-06)	**2.5e-06 (1.1e-07)**
All	2.7e-04 (8.9e-04)	2.7e-04 (8.9e-04)	8.4e-03 (2.8e-02)	**2.6e-04 (8.8e-04)**

Table 1: Mean-Squared Error on Cartpole (with 95% confidence intervals).

batch inference procedure was repeated 5 times for 250 iterations. We set $\sigma_w = 4$, $\alpha = 2$, and optimized the Gaussian process parameters.

Next, 50 training points were selected from each run of (m, l) settings with $m \in \{.1, .15, .2, .25, .3\}$ and $l \in \{.4, .45, .5, .55, .6\}$. The online inference procedure was used to estimate the weights w_{kb} given the filter parameters z_{kad} and basis functions f_{kad} from the batch procedure. The quality of the predictions $\Delta(s)$ on 50 unseen test points are shown in Table 1, which compares standard regression using different combinations of input data to the HiP-MDP model. Standard regression using only the batch points, and using both batch and training, is equivalent to treating all of the task instances as a single MDP, while regression using the training points for each (m, l) setting performs no transfer and results in the highest error. Our IBP-GP model uses both the batch data and the training points for each (m, l) setting, modeling both the commonalities and differences between MDPs in the task family. It performs similarly to a single batch-trained GP for predicting the change in outputs x and θ, since these state variables do not depend on m or l (see equation 4). It significantly outperforms the other settings for outputs $\dot{x}$ and $\dot{\theta}$, whose changes *do* depend on the parameters of system. Visualizations interpreting the latent parameters are included in the appendix.

5.2 Acrobot

Cartpole is simple enough that changing m and l does not change the optimal policy—if the pole is falling to the left, the cart should be moved left, and similarly to the right. Acrobot, in which the agent must swing up a double pendulum only through applying a positive, neutral, or negative torque to the joint between the two poles, is much more challenging. The state space consists of the angle θ_1 and angular velocity $\dot{\theta_1}$ of the first segment (with mass m_1) and of the second segment (θ_2 and $\dot{\theta_2}$, with mass m_2). Small changes in mass can substantially impact the policy.

For batch training, we used mass settings (m_1, m_2) settings of { (.7,.7) , (.7,1.3) , (.9,.7) , (.9,1.1) , (1.1,.9) , (1.1,1.3) , (1.3,.7) , (1.3, 1.3) }, again employing Sarsa (this time with a 5th order Fourier Basis). 1000 support points were chosen to minimize the maximum prediction error within each batch. Next, we inferred the filter parameters z_{kad} and obtained approximate MAP-estimates for the basis functions f_{kad}.

Table 2 shows mean-squared prediction errors using the same evaluation procedure as cartpole, with (m_1, m_2) settings of $m_1 \in \{.7, .9, 1.1, 1.3\}$ and $m_2 \in \{.7, .9, 1.1, 1.3\}$. The HiP-MDP predicts better overall, with slightly higher mean-squared errors on the angle predictions and lower errors on the angular velocity predictions. Predicting angular velocities is critical to planning in acrobot; inaccurate predictions will make the agent believe it can reach the swing-up position more quickly than is physically possible.

	Batch + Train	Batch Only	Train Only	IBP-GP HIP-MDP
$\dot{\theta_1}$	5.1e-05 (3.1e-06)	5.4e-05 (3.3e-06)	3.8e-04 (3.1e-05)	5.5e-05 (8.3e-06)
$\dot{\theta_2}$	1.4e-04 (1.2e-05)	1.5e-04 (1.2e-05)	8.8e-04 (9.6e-05)	1.3e-04 (2.1e-05)
$\ddot{\theta_1}$	7.0e-04 (4.5e-05)	7.2e-04 (4.8e-05)	1.6e-03 (1.5e-04)	3.6e-04 (4.0e-05)
$\ddot{\theta_2}$	4.2e-04 (2.5e-05)	4.3e-04 (2.7e-05)	9.1e-04 (1.0e-04)	2.3e-04 (2.1e-05)
All	3.3e-04 (3.8e-05)	3.4e-04 (3.9e-05)	9.4e-04 (8.1e-05)	2.0e-04 (2.1e-05)

Table 2: Mean-squared prediction error on the acrobot state variables (with 95% confidence intervals).

In acrobat, a policy learned with one mass setting will generally perform poorly on another. To evaluate our HiP-MDP approach in a control setting, we ran 30 trials in each of the 16 settings above, filtered the weight parameters w_k during an episode, and updated the policy at the end of each episode. As finding the full Bayesian RL solution is PSPACE-complete [Fern and Tadepalli, 2010] and offline approximation techniques are an active area of research [Bai *et al.*, 2013], we performed planning using Sarsa with dynamics based on the mean weight parameters w_k. We first obtained an initial value function using the prior mean weights, and then interleaved model-based planning and reinforcement learning, using 5 episodes of planning for each of interaction.

Results on acrobot are shown in Figure 2. We compare performance (averaged over weight settings) to planning using the true model, and to planning using an average model (learned using all the batch data at once) to obtain an initial value function. Both the average and HiP-MDP model reach performance near, but not quite as high as, using the true model. However, the HiP-MDP model is already near that performance by the 2nd episode. Using the learned bases from the batches, as well as learning the weights from the first episode, lets it quickly "snap" very near to the true dynamics. Our HiP-MDP model reaches near-optimal performance in 5 episodes, compared to 15 for the average model.

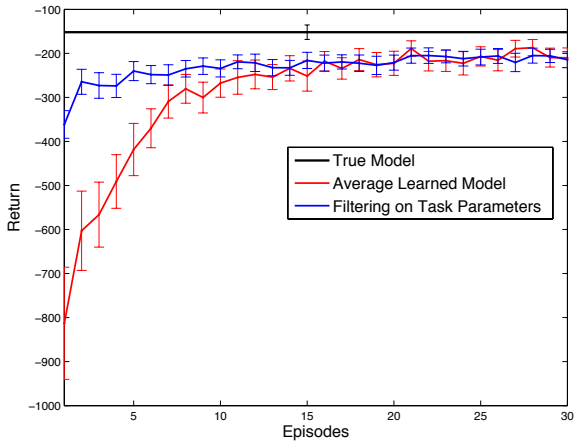

Figure 2: Acrobot performance. The HiP-MDP quickly approaches near-optimal performance.

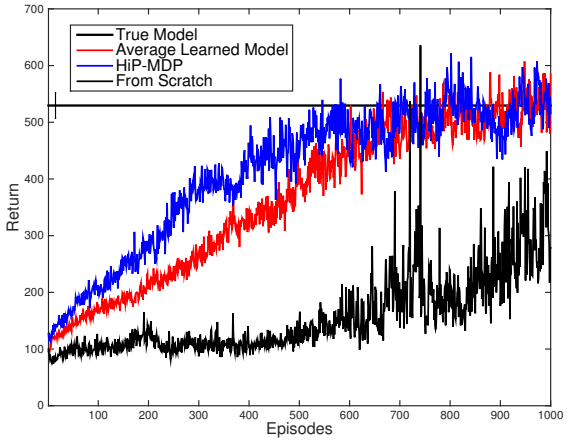

Figure 3: Bicycle performance. The HiP-MDP quickly approaches near-optimal performance.

5.3 Bicycle

The bicycle domain [Randlov and Alstrom, 1998] requires the agent to keep a bicycle traveling at a constant speed upright for as long as possible within a bounded 60x60m area. The agent has three torque actions on the handlebars—$\{-2\text{N}, 0, 2\text{N}\}$—and three mass-displacement actions—$\{-2\text{cm}, 0, 2\text{cm}\}$—for a total of nine possible actions. To simulate imperfect balance, a random displacement between $[-2\text{cm}, 2\text{cm}]$ is added to the agent's displacement action. The dynamics of the bicycle depend on several parameters, and here we vary the bicycle's height and mass, physical parameters to which a real-life bicycle-riding agent should be robust.

Our batch training data consisted of two trials of 10 episodes for the bicycle height ranging from 0.74 to 1.24 meters in increments of 0.15 and weight ranging from 10 to 20 kilograms in increments of 5. These batch data were used to infer the filter parameters z_{kad} and approximate MAP-estimates for the basis functions f_{kad} for the HiP-MDP.

At test time, we ran 25 trials for each setting of the two task parameters. For each trial, we first used the mean model parameter values to compute an initial value function; we then mixed learning and planning—one episode of planning for each 10 episodes of real data, up to 200 episodes of real data—while filtering the weight parameters w_k. Both learning and planning was performed using Sarsa(λ) with $\alpha = 0.05$ and $\lambda = 0.95$, an epsilon-greedy exploration policy with $\epsilon = 0.05$, and reward discount $\gamma = .99$, using tile coding with the boundaries used in the original paper [Randlov and Alstrom, 1998] and then continued with Sarsa. The average model computed a single set of bicycle dynamics from all of the batch data. We applied 1500 episodes of Sarsa on this model to learn a value function to initialize the learner in a new setting. Figure 3 compare the HiP-MDP and averaged approaches to learning each bicycle parameter configuration from scratch, averaged over all height and mass configurations. Both approaches show positive transfer, but the HiP-MDP has a much faster learning trajectory, reaching near-optimal performance nearly 150 episodes before the average-initialized approach. The slow rate of the learner that starts from scratch demonstrates the overall difficulty of the bicycle problem.

6 Discussion and Related Work

Bai *et al.* [2013] use a very similar hidden-parameter setting treated as a POMDP to perform Bayesian planning; they assume the model is given, whereas our task is to learn it. The HiP-MDP is similar to other POMDPs with fixed hidden states, for example, POMDPs used for slot-filling dialogs [Williams and Young, 2005]. The key difference, however, is that the objective of the HiP-MDP is not simply to gather information (e.g., simply learn the transition function T); it is to perform well on the task (e.g., drive a new car). Transfer learning—the goal of the HiP-MDP—has received much attention in reinforcement learning. Most directly related in spirit to our approach are Hidden-Goal MDPs [Fern and Tadepalli, 2010], hidden-type MDPs and bandits [Azar *et al.*, 2013; Brunskill and Li, 2013; Mahmud *et al.*, 2014; Rosman *et al.*, 2016] and hierarchical model-based transfer [Wilson *et al.*, 2012a]. In these settings the agent must determine its current MDP from a discrete set of MDPs. However, note that in the discrete setting the agent switches between distinct MDPs (or, in the case of Wilson *et al.* [2012a], a finite set of distributions over discrete MDPs), rather than learning how a continuous parametrization modifies the dynamics of the entire family. Representation transfer approaches [Ferguson and Mahadevan, 2006; Taylor and Stone, 2007; Ferrante *et al.*, 2008] typically focus on learning a set of basis functions sufficient for representating any value function defined in a specific state space, or on transfer between two different representations of the same task. By contrast, the HiP-MDP focuses on modeling the dimensions of variation of a family of related tasks.

A recent series of papers [Ammar *et al.*, 2014; Bou Am-

mar *et al.*, 2015b; 2015a] transfer learned policy parameters across families of related tasks using policy gradient methods. These papers synthesize new policies as linear combinations of policy components from previously solved instances. They are complementary to our approach, which focuses on parametrizing the environmental model rather than the policy. However, the supervised nature of transition models means that our approach is not limited to policy gradients; we can also infer the number of parameters in the environmental model from data.

The IBP-GP prior itself relates to a body of work on multiple-output Gaussian processes (e.g. [Álvarez and Lawrence, 2011; Teh *et al.*, 2005; Wilson *et al.*, 2012b]), though most of these are focused on learning a convolution kernel to model several related outputs on a single task, rather than parameterizing several related tasks. Gaussian process latent variable models [Lawrence, 2004] have been used for dimensionality reduction in dynamical systems. As with other multi-output GP models, however, GP-LVMs find a time-varying, low-dimensional representation for a single system, while we characterize each instance in a set of systems by a stationary, low-dimensional vector. The focus of these efforts has also been on modeling rather than control.

The extensive work on scaling inference in these Gaussian process models [Álvarez and Lawrence, 2011; Titsias, 2009; Damianou *et al.*, 2011] provides avenues for relaxing the approximations made in this work (while adding new ones). In settings where one may not have an initial batch of data from several instances, a fully Bayesian treatment of the filter parameters z_{kad} and the basis functions f_{kad} might allow the agent to more accurately navigate its exploration-exploitation trade-offs. Exploring which uncertainties are important to model—and which are not—is an important question for future work. Other extensions within this particular model include applying clustering or more sophisticated hierarchical methods to group together basis functions f_{kad} and thus share statistical strength. For example, one might expect that "opposing actions," such as in cartpole, could share similar basis functions.

Another interesting direction is allowing for some time-varying dynamics within the problem, but in structured ways that still make inference relatively easy. For example, we may place priors on the weights w_k that allow them to shift—but slowly. Or we may believe that abrupt changes are possible but unlikely—such as an agent moving from one kind of surface to another. Finally, it would be interesting to explore the limit of what kind of transfer is possible: given data from two differing domains, will we just learn two sets of basis functions with non-overlapping filter parameters z_{kad}? What kind of modeling hierarchies might allow for more sharing between only somewhat-related tasks?

7 Conclusion

Machine learning approaches for control generally expect repeated experiences in the same domain. However, in practice agents are more likely to experience repeated domains that vary in limited and specific ways. In this setting, traditional planning approaches that rely on exact models may fail due to the large inter-instance variation. By contrast, reinforcement learning approaches, which often assume that the dynamics of a new instance are unknown, may fail to leverage information across related instances.

The HiP-MDP model explicitly captures this inter-instance variation, providing a compromise between these two standard paradigms for control. It should prove useful when the family of domains has a parametrization that is small relative to its model and where objectives remain similar across domains. Many applications—such as handling similar objects, driving similar cars, even managing similar network flows—fit this scenario, where batch observational data from related tasks are easy to obtain in advance. In such cases, being able to generalize dynamics from only a few interactions with a new operating regime (using data from many prior interactions with similar systems) is a key step in building controllers that exhibit robust and reliable decision-making while gracefully adapting to new situations.

Acknowledgments

This research was supported in part by DARPA under agreement number D15AP00104, and by the National Institutes of Health under award number R01MH109177. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or DARPA.

A Visualizing Latent Features for Cartpole

In all 5 of the MCMC runs, a total of 4 latent parameters were inferred. The output dimensions x and θ consistently only used the first (baseline) feature—that is, our IBP-GP's predictions were in fact the same as using a single GP. This observation is consistent with the cartpole dynamics and the observed prediction errors in figure 4. The second feature was used by both $\dot{x}$ and $\dot{\theta}$ and was positively correlated with both the pole mass m and the pole length l (figures 6, 5, and 5). In the cartpole dynamics equations, both $\dot{x}$ and $\dot{\theta}$ have many ml terms. The third consistently discovered feature was used only by $\dot{x}$ and was positively correlated with the pole mass m and not correlated with the pole length l; the equations for $\dot{x}$ have several terms that depend only on m. The fourth feature (used only to predict $\dot{x}$) had the highest variability; it generally has higher values for more extreme length settings, suggesting that it might be a correction for more complex nonlinear effects.

References

[Álvarez and Lawrence, 2011] M.A. Álvarez and N.D. Lawrence. Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, 12:1459–1500, 2011.

[Ammar *et al.*, 2014] H.B. Ammar, E. Eaton, P. Ruvolo, and M.E. Taylor. Online multi-task learning for policy gradient methods. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1206–1214, 2014.

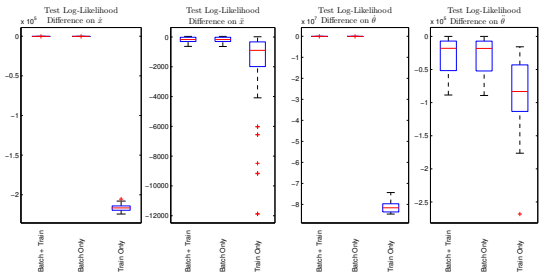

Figure 4: Difference in Test Log-Likelihood on Cartpole. Negative values indicate performance worse than the HiP-MDP

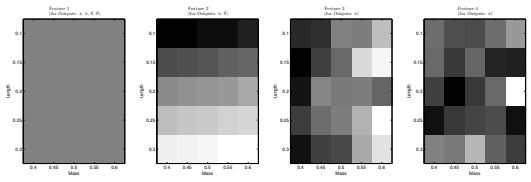

Figure 5: Cartpole Weights vs. Mass and Length

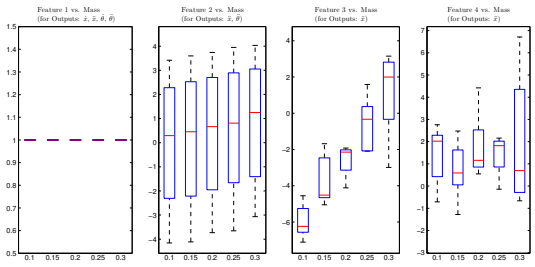

Figure 6: Cartpole Weights vs. Mass

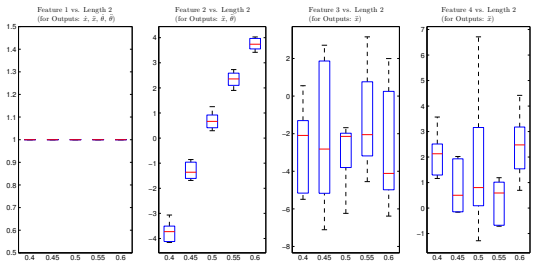

Figure 7: Cartpole Weights vs. Length

[Azar *et al.*, 2013] M. Azar, A. Lazaric, and E. Brunskill. Sequential transfer in multi-armed bandit with finite set of models. In *Advances in Neural Information Processing Systems 26*, pages 2220–2228, 2013.

[Bai *et al.*, 2013] H. Y. Bai, D. Hsu, and W. S. Lee. Planning how to learn. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2853–2859, 2013.

[Bou Ammar *et al.*, 2015a] H. Bou Ammar, E. Eaton, J.M. Luna, and P. Ruvolo. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3345–3351, July 2015.

[Bou Ammar *et al.*, 2015b] H. Bou Ammar, E. Eaton, P. Ruvolo, and M.E. Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2504–2510, January 2015.

[Brunskill and Li, 2013] E. Brunskill and L. Li. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, July 2013.

[da Silva *et al.*, 2012] B.C. da Silva, G.D. Konidaris, and A.G. Barto. Learning parameterized skills. In *Proceedings of the Twenty Ninth International Conference on Machine Learning*, pages 1679–1686, June 2012.

[Damianou *et al.*, 2011] A.C. Damianou, M.K. Titsias, and N.D. Lawrence. Variational Gaussian process dynamical systems. In *Advances in Neural Information Processing Systems 24*, 2011.

[Ferguson and Mahadevan, 2006] K. Ferguson and S. Mahadevan. Proto-transfer learning in Markov decision processes using spectral methods. In *The ICML Workshop on Structural Knowledge Transfer for Machine Learning*, June 2006.

[Fern and Tadepalli, 2010] Alan Fern and Prasad Tadepalli. A computational decision theory for interactive assistants. In *Advances in Neural Information Processing Systems 23*, pages 577–585, 2010.

[Ferrante *et al.*, 2008] E. Ferrante, A. Lazaric, and M. Restelli. Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *Autonomous Agents and Multiagent Systems*, pages 1329–1332, 2008.

[Griffiths and Ghahramani, 2011] T.L. Griffiths and Z. Ghahramani. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

[Guez *et al.*, 2012] A. Guez, D. Silver, and P. Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems 25*, 2012.

[Kaelbling *et al.*, 1998] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[Kober *et al.*, 2012] J. Kober, A. Wilhelm, E. Oztop, and J. Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):316–379, 2012.

[Konidaris *et al.*, 2011] G.D. Konidaris, S. Osentoski, and P.S. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, pages 380–385, 2011.

[Kurniawati *et al.*, 2008] H. Kurniawati, D. Hsu, and W.S. Lee. Sarsop: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems*, 2008.

[Lawrence, 2004] N.D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems 16*, 2004.

[Mahmud *et al.*, 2014] M.M.H Mahmud, B. Rosman, S. Ramamoorthy, and P. Kohli. Adapting interaction environments to diverse users through online action set selection. In *Proceedings of the AAAI 2014 Workshop on Machine Learning for Interactive Systems*, 2014.

[Poupart *et al.*, 2006] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Twenty-Third International Conference in Machine Learning*, pages 697–704, 2006.

[Randlov and Alstrom, 1998] J. Randlov and P. Alstrom. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, 1998.

[Rasmussen and Williams, 2005] C.E. Rasmussen and C.K.I Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[Rosman *et al.*, 2016] B. Rosman, M. Hawasly, and S. Ramamoorthy. Bayesian policy reuse. *Machine Learning*, 2016. To appear.

[Ross *et al.*, 2008] S. Ross, B. Chaib-draa, and J. Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *The IEEE International Conference on Robotics and Automation*, pages 2845 – 2851, 2008.

[Silver and Veness, 2010] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems 23*, pages 2164–2172, 2010.

[Snelson and Ghahramani, 2005] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, 2005.

[Sutton and Barto, 1998] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Taylor and Stone, 2007] M.E. Taylor and P. Stone. Representation transfer for reinforcement learning. In *AAAI 2007 Fall Symposium on Computational Approaches to Representation Change during Learning and Development*, November 2007.

[Teh *et al.*, 2005] Y. W. Teh, M. Seeger, and M. I. Jordan. Semiparametric latent factor models. In *International Workshop on Artificial Intelligence and Statistics*, 2005.

[Titsias, 2009] M. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of The 12th International Conference on Artificial Intelligence and Statistics*, 2009.

[Williams and Young, 2005] J. Williams and S. Young. Scaling up POMDPs for dialogue management: The "summary POMDP" method. In *Proceedings of the IEEE ASRU Workshop*, 2005.

[Wilson *et al.*, 2012a] A. Wilson, A. Fern, and P. Tadepalli. Transfer learning in sequential decision problems: A hierarchical Bayesian approach. *Journal of Machine Learning Research - Proceedings Track*, 27:217–227, 2012.

[Wilson *et al.*, 2012b] A.G. Wilson, D.A. Knowles, and Z. Ghahramani. Gaussian process regression networks. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.