Optimism-Driven Exploration for Nonlinear Systems

Teodor Mihai Moldovan, Sergey Levine, Michael I. Jordan, Pieter Abbeel

***Abstract*— Tasks with unknown dynamics and costly system interaction time present a serious challenge for reinforcement learning. If a model of the dynamics can be learned quickly, interaction time can be reduced substantially. We show that combining an optimistic exploration strategy with model-predictive control can achieve very good sample complexity for a range of nonlinear systems. Our method learns a Dirichlet process mixture of linear models using an exploration strategy based on optimism in the face of uncertainty. Trajectory optimization is used to plan paths in the learned model that both minimize the cost and perform exploration. Experimental results show that our approach achieves some of the most sample-efficient learning rates on several benchmark problems, and is able to successfully learn to control a simulated helicopter during hover and autorotation with only seconds of interaction time. The computational requirements are substantial.**

I. Introduction

Learning unknown system dynamics while simultaneously using this information to reach a goal has been a serious challenge for reinforcement learning (RL) in the case of continuous state spaces. Such tasks arise in robotics, where time constraints, safety, and wear-and-tear place stringent limits on the amount of system interaction allowed [1], as well as adaptive control, where the dynamics change unpredictably and the controller must adapt online [2]. Model-based RL reduces the required interaction time by learning a model of the dynamics. Although many methods have been proposed for model-based RL with efficient exploration in discrete MDPs, data-efficient model-based RL in continuous systems remains a challenging problem despite substantial recent advances [3], [4], [5].

We propose a model-based RL algorithm for continuous systems based on the principle of optimism in the face of uncertainty. Our exploration strategy chooses exploratory actions based on the estimated uncertainty in the learned dynamics model. To exploit new information about the model as soon as it becomes available, we replan the actions online using a model-predictive control (MPC) algorithm that integrates with our exploration strategy. This requires a model that can be updated efficiently and provides uncertainty estimates for exploration. We use a Dirichlet process mixture model (DPMM), which scales gracefully with the number of samples and provides Bayesian uncertainty estimates. By updating the model online and using MPC to continually replan optimistic trajectories in light of new observations, our method can learn continuous tasks with very little system interaction.

Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94709

Our main contribution is a data-efficient reinforcement learning method based on model-predictive control. We extend optimism-driven exploration, which has previous been applied mainly to discrete problems [6], [7], to continuous nonlinear deterministic systems, and show that a dynamics representation based on the Dirichlet process mixture models (DPMM) [8] can be effectively combined with this exploration strategy. We perform MPC using a pseudospectral trajectory optimization method [9] that incorporates our exploration strategy into the dynamics equations. In our evaluation, we demonstrate that our approach not only achieves state of the art sample efficiency on standard benchmark problems, but that it is also able to do so without the use of any shaping cost, with only the target state available beforehand. This is in contrast to most continuous methods, which typically employ a smooth shaping cost [3], [10]. Furthermore, since we use online trajectory optimization, we do not rely on an episodic formulation of the task. This allows us to perform adaptive control in settings where restarting the episode is impossible.

The asymptotic complexity of our algorithm in the number of data points collected is $O(n \log n)$ and could be improved to sublinear at the expense of further approximations [11], a significant improvement from the competing Gaussian Process based approach [3], however the afferent constants make our implementation two orders of magnitude slower than real-time. To demonstrate this capability, we learn a hovering policy for a simulated helicopter mid-flight, using only 5 seconds of interaction time and losing on average 3.5 meters of altitude before stabilizing the hover, as well as an autorotation maneuver to recover from an engine failure, using only 8 seconds of interaction time.

II. Overview

Optimism in the face of uncertainty is a general principle for model-based reinforcement learning underlying numerous efficient algorithms backed by theoretical guarantees in multiple settings. Policies based on upper confidence bounds (UCB) illustrate this principle [12] for multi-armed bandit problems. The Bayesian optimistic local transitions algorithm [6] and the optimistic linear programming algorithm [7] apply the same principle to discrete MDPs, and recent results show regret bounds also based on this principle for linear systems with quadratic costs [4].

We apply optimism-driven exploration to nonlinear dynamical systems, using trajectory optimization to act optimistically with respect to the learned dynamics. Our method extends the approaches presented in [4] and [6] to nonlinear dynamical systems. In addition to choosing actions, the

Algorithm 1 Continuous RL with optimistic exploration

Require: dynamical system starting at state x, optional goal state x_{goal}
Require: sampling frequency ν_s, control frequency ν_c
  $\tau \leftarrow$ zero control trajectory
  $z \leftarrow$ empty list of observations
  repeat
    $x \leftarrow$ state after following τ for $1/\nu_c$ seconds starting at state x
    $z \leftarrow z$ appended with states, time derivatives, controls $(\dot{x}, x, u)$ sampled every $1/\nu_s$ seconds.
    $f \leftarrow$ learned dynamics given all observations z as specified in Section V.
    $\tau \leftarrow$ optimal trajectory for dynamics f with starting state x as specified in Section IV
  until task objective achieved

trajectory optimization algorithm is also allowed to pick favorable dynamics that are sufficiently likely under the model based on past observations. Favorable dynamics are chosen relative to the cost function and, consequently, our exploration strategy prioritizes learning those aspects of the dynamics that are most relevant for success. As more data becomes available, the uncertainty in the model decreases, since alternatives that disagree with the true dynamics become less likely. This results in progressively less exploration as the model improves. Asymptotically, exploration reduces to zero with enough data, though in practice, the goal is often reached before this happens.

Analogously to other exploration methods based on optimism, we take an online approach that avoids the need for restarts or episodes. This allows our algorithm to learn with the fewest number of samples and to perform adaptive control. As summarized in Algorithm 1, our method observes the current state, updates the model as described in Section V, and recomputes the trajectory with optimistic dynamics, as described in Sections III and IV. This is done online at regular intervals. Intuitively, this procedure succeeds because either the algorithm makes progress towards the goal, or the optimistic dynamics are incorrect and the newly collected data ensures that similar mistakes will be avoided in the future. This approach requires a model-predictive control (MPC) algorithm that continually recomputes the trajectory to account for new observations, and a dynamics model that can represent uncertainty. We use a pseudospectral trajectory optimization algorithm for MPC [9], and a Dirichlet process mixture model (DPMM) [13] for the dynamics, which strikes a good balance between expressiveness and computational complexity, as discussed in Section V.

III. Optimism Based Exploration in Nonlinear Dynamical Systems

We assume that the true dynamics are given by $\dot{x} = f^*(x, u)$, where $x, \dot{x}$ and u are the state of the system, its time derivative, and the control. The system is assumed to be deterministic[1] but state observations are allowed to be noisy, as is typically unavoidable in practical systems. The algorithm must minimize the integral of a cost rate $q(x(t), u(t))$ while optionally reaching a goal state x_{goal}, which may be only partially specified.[2] If none of the dimensions of the goal state are specified, the algorithm simply minimizes the total cost. The planning algorithm does not have access to the true dynamics, it only has access to a statistical estimator f based on past observations. As discussed in Section V, we used a DPMM for this estimator, though any consistent estimator may be used. The MPC algorithm is allowed to use optimistic dynamics in the vicinity of the expected dynamics that deviate by at most one standard deviation in each dimension as specified by the covariance matrix. This approach leads the system to be optimistic about the way dynamics will play out in regions where the dynamics model estimate still has high covariance; if indeed the dynamics play out as optimistically hoped for, then this is a good exploitation trajectory, if they don't, then the model estimate will be updated with data from that region, resulting in a reduced covariance and less optimism about the dynamics in that region going forward. Asymptotically, the expected dynamics converge to the true dynamics and the standard deviation converges to zero, so the planning algorithm will asymptotically obey the true dynamics.

Given the estimator f of the dynamics, our optimistic planning algorithm can be formulated as a constrained functional optimization problem:

$$\begin{aligned}
&\text{minimize}_{\ x,u,\xi,h>0}: \quad \int_{t=0}^{h} q(x(t),u(t))dt \qquad (1)\\
&\text{s.t.} \quad \dot{x}(t) = E[f(x(t),u(t))] + \text{Cov}^{\frac{1}{2}}[f(x(t),u(t))]\cdot\xi(t),\\
&\qquad -1 \le u(t), \xi(t) \le 1, \quad x(0) = x_{\text{start}}\\
&\text{optional constraints:} \quad x(h) = x_{\text{goal}},\ h = h_{\text{fixed}}
\end{aligned}$$

where the decision variables $x(t), u(t), \xi(t)$ are vector functions of time representing the state of the system, the controls to be applied and virtual controls encoding exploration serving optimism. The trajectory duration h is also included as a decision variable and the function $q(x(t), u(t))$ is an arbitrary cost rate function.

Without the virtual controls ξ, this optimization is a standard optimal control task. The additional virtual controls ξ, which are scaled by the uncertainty in the model, specify how much the dynamics are allowed to deviate from the mean in units of standard deviation. The formulation can either treat the horizon h as an optimization variable or, optionally, constrain the horizon to a fixed value. Similarly, it is possible but optional to specify a goal state as a constraint, either fully or only partially. Since this problem is defined in terms of states and controls (the physical controls plus

[1]The model we employ (DPMM) supports both stochastic and deterministic systems. The planning algorithm, however, only supports deterministic systems; our method could be adapted to stochastic systems by swapping in a suitable planner in its place.

[2]If the goal state is not specified at all, we can employ a minimum cost formulation by adding cost as an additional term in the state, which can then be minimized under the same optimization framework.

the virtual controls), it can be solved using any optimal control algorithm. We use a pseudospectral sequential linear programming solver discussed in Section IV.

Larger or smaller bounds for ξ may be used to increase or reduce the amount of exploration by replacing the constraint $-1 \leq \xi \leq 1$ with $-p \leq \xi \leq p$ for some fixed $p > 0$. Doing so allows p standard deviations from the mean in each dimension. We found that resizing the prediction region was not necessary in our experiments and simply used $p = 1$. Note that the planner is allowed to choose different favorable dynamics (through different ξ) at different time steps; this is consistent with the formulation of previously proposed algorithms based on the same principle [12], [6], [7], [4].

IV. Optimal Control

We control the dynamical system using model-predictive control (MPC) [14], [15], [16], [17]. This section does not contain any novel contributions; it describes our choice of optimal control algorithm and clarifies how optimal control is used in conjunction with the exploration objective.

Our exploration framework requires optimizing a trajectory based on Equation (1), executing the actions along this trajectory in open loop for $1/\nu_c$ seconds, and then reoptimizing the trajectory based on the new state and the updated model. Since the dynamics model is continually changing, constant replanning is likely to be required, as is the case in prior methods [18]. MPC is an approach for optimal control that fits naturally with our exploration formulation and is well suited for sample-efficient online learning. We briefly summarize our approach to optimal control in this section, though in practice any trajectory optimization algorithm that can solve problems of the type shown in Equation (1) would be suitable.

The optimal control objective is to find a sequence of states and controls that minimize the time integral of the cost rate function $q(\dot{x}, x, u)$ while optionally enforcing that a goal state is reached. To solve this continuous control problem, we must first discretize it in time. Instead of enforcing the dynamics at all times, we now only enforce them at a finite number k of appropriately chosen collocation times $0 \leq h\tau_1 \leq h\tau_2 \leq \cdots \leq h\tau_k \leq h$, where h is the time horizon and $\tau_i \in [0, 1]$. To simplify notation, let $\dot{x}_i = \dot{x}(h\tau_i)$. The states $x(t)$ can be expressed as time integrals of the time derivatives of states as follows:

$$x(h\tau_i) - x(0) = h \int_0^{\tau_i} \dot{x}(h\tau) d\tau \approx h \sum_j A_{ij}(k) \dot{x}_j$$

$$x(h) - x(0) = h \int_0^{1} \dot{x}(h\tau) d\tau \approx h \sum_j w_j \dot{x}_j,$$

where the approximate integration operators A_{ij} and w_j are chosen depending on the type of time discretization scheme used.

The optimal control problem can now be expressed as a discrete time non-linear optimization problem:

$$\begin{aligned} &\text{minimize}_{\ \dot{x}_i, x_i, u_i, \xi_i, h>0} : \ h \sum_i w_i q(x_i, u_i) \qquad (2) \\ &\text{s.t.} \quad \dot{x}_i = E[f(x_i, u_i)] + \text{Cov}^{\frac{1}{2}}[f(x_i, u_i)] \cdot \xi_i, \\ &\qquad x_i := x_{\text{start}} + h \sum_j A_{ij} \dot{x}_j, \quad -1 \leq u_i, \xi_i \leq 1 \\ &\text{optional constraints: } x_{\text{goal}} - x_{\text{start}} = h \sum_i w_i \dot{x}_i, \ h = h_{\text{fixed}} \end{aligned}$$

where the decision vector variables $\dot{x}_i, x_i, u_i, \xi_i$ are now time-discretized versions of the corresponding variables from Equation (1). We have thus replaced a constrained functional optimization problem to a constrained finite dimensional optimization problem.

Pseudospectral methods [9] specify efficient time discretization schemes grounded in implicit integration theory. In these schemes, the time steps $\tau_1, \ldots \tau_k$ are not equally spaced, but instead are clustered more densely around the endpoints. The approximate integration operators A_{ij} and w_j are prescribed accordingly. Pseudospectral methods are advantageous because fewer collocation times can be used without compromising accuracy, the dual variables of the time-discretized optimization problem are provably close to the co-states of the continuous time optimization problem, and interpolation of the resulting controls between collocation time steps is well specified. In our experiments we used Gaussian pseudospectral time discretization [9].

The optimization problem (2) is non-convex, so in general we can only guarantee convergence to a (potentially infeasible) local optimum. In our experiments we used a sequential linear programming solver [19], though any other efficient non-linear solver may also be used.

V. Dirichlet Process Mixture of Linear Models

Optimism-driven exploration requires accurate variance estimates (or trust regions) for the partially learned dynamics model. Furthermore, since our algorithm uses online updates, the dynamics model must strike a good balance between representational power and ability to scale with the number of samples. In our experiments, we use the Dirichlet process mixture model (DPMM) for this purpose. Many other probabilistic models are also suitable for use with our method, though the DPMM provides a good balance of flexibility and efficiency, particularly for physical systems that are well approximated by the piecewise linear dynamics that are induced by a Gaussian mixture [20]. The DPMM automatically chooses the number of clusters in this mixture, as described in this section. We first review the DPMM [13] and then propose a simple posterior distribution approximation, equivalent to weighted least squares, that is well suited for dynamics prediction. At a high level, it is sufficient to think of this model as fitting a Gaussian mixture with a Bayesian procedure that automatically adapts the number of clusters and provides estimates of the uncertainty in the model.

A single Gaussian distribution (or, equivalently, a single linear model) is not sufficient to capture the dynamics of

complex nonlinear systems, but a mixture of Gaussians is often descriptive enough if we assume that, for each sufficiently small region in state space, the underlying (unknown) dynamical system is locally linear. Unfortunately, it is generally not possible to determine the optimal number of clusters a priori and this parameter has a large impact—too many clusters leads to over-fitting and too few clusters collapses modes that could be discriminated between with enough data. Dirichlet process mixture models (DPMMs) tackle this issue by allowing a countably infinite number of clusters, with the actual number of clusters determined via posterior inference. Under the Dirichlet process prior the expected number of clusters grows logarithmically in the number of observations [13] and posterior inference sharpens this growth rate so as to control model complexity as we observe more data. Our model is based on the following standard DPMM [8]:

$$\begin{aligned} G \,|\, \alpha, \lambda &\sim \mathrm{DP}(\alpha, \mathcal{NW}^{-1}(\lambda)) \\ \eta_n \,|\, G &\sim G \\ z_n \,|\, \eta_n &\sim \mathcal{N}(\eta_n), \end{aligned}$$

where α is the concentration parameter of the Dirichlet process prior and the base measure is the normal-inverse-Wishart distribution.

The DPMM represents a distribution over clusterings of the stacked vector $z = [\dot{x}, x, u]$. As discussed later in this section, we use this model to predict $\dot{x}$ by conditioning on the covariate x, u. For convenience of notation we only show the DPMM for learning full state dynamics. Learning partial models is also possible (as shown in our experiments) by defining z in terms of the relevant partial state components. It is convenient to rewrite this model using an exponential family representation. The natural parameter vector for the normal-inverse-Wishart prior is obtained by stacking together the usual parameters: $\lambda = [nm, \Psi + nmm^T, n, \nu + 2 + d_z]$, where d_z is the dimension of z. The vector of sufficient statistics is also obtained by concatenation: $T(z) = [z, zz^T, 1, 1]$. What makes this representation convenient is that posterior updates are now extremely simple: $\tau = \lambda + T(z)$. Given the importance of α in DPMMs, we treat it not as a fixed hyperparameter but endow it with a gamma prior distribution, $\mathrm{Ga}(0, 0)$, which we integrate over via the posterior inference algorithm.

a) Variational Inference: While Dirichlet process mixture models are often fit with one of a number of different Markov chain Monte Carlo or sequential Monte Carlo algorithms, given our stringent requirements for computational efficiency we have chosen instead to make use of variational inference methods. In particular, we make use of the mean-field variational inference procedure for DPMMs developed by Blei and Jordan [8]. This procedure, based on a stick-breaking representation of the Dirichlet process, requires us to set an upper bound on the number of clusters that can be represented. This is a parameter of the inference procedure, not of the model, and it is generally set to be as high as computational resources permit. Many of the represented clusters will not be populated.

The mean-field variational inference algorithm of Blei and Jordan [8] can be viewed as an approximate version of expectation-maximization that iterates between the following two steps:

E step: (3)

$$\varphi_{k,n} \propto \exp\left(\nabla A(\tau_k)^T T(z_n) + \psi(a_k) - \psi(b_k) + \sum_{i=1}^{k} (\psi(b_i) - \psi(a_i + b_i))\right)$$

M step: (4)

$$\begin{aligned} \tau_k &= \lambda + \sum_n \varphi_{k,n} T(z_n), \\ a_k &= 1 + \sum_n \varphi_{k,n}, \\ b_k &= \alpha + \sum_n \sum_{j=k+1}^{K} \varphi_{j,n}, \end{aligned}$$

where ψ is the digamma function, A is the partition function of the normal-inverse-Wishart prior, and $\varphi_{k,n}$ and τ_k are variational parameters that are the degrees of freedom of the algorithm.

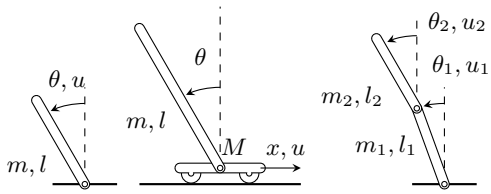
To initialize the base measure we start with $\lambda_0 = [0, 0, 0, 2d_z + 1]$ and we perform a weighted posterior update imagining that the entire training set belongs to a single cluster:

$$\lambda = \lambda_0 + w \cdot \sum_n T(z_i) / \sum_n T(z_i)_{-1},$$

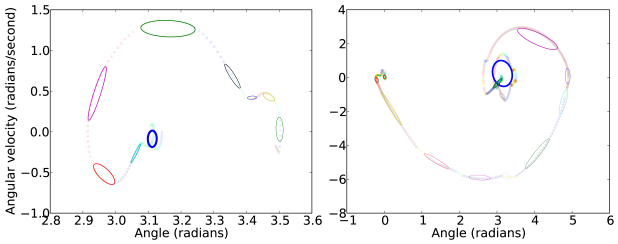
where $T(z_i)_{-1}$ is the last component of the vector of sufficient statistics, and w specifies the relative importance of the prior with respect to a single observation. With this choice of prior parameters the clustering method is invariant to affine transformations of the dataset.

Online variational inference methods (e.g. [11]) could be used for a further improvement in computational complexity and running time. We leave this investigation for future work since the computational cost of planning dominates the cost of inference in our implementation and the additional approximations required for online inference might adversely affect learning performance.

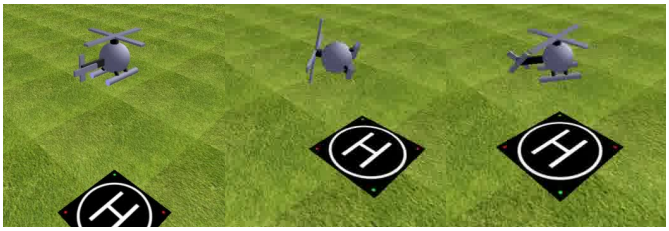
b) Dynamics Prediction: Once the mixture model has been trained, we use it to predict $\dot{x}$ given x, u. Unlike previous approaches that make predictions based only on conditional expectations with the DPMM [21], our exploration method critically depends on establishing good estimates of variance. A difficulty in the DPMM setting is that it is possible for the conditional distribution under the model to be multi-modal, which can complicate planning by introducing local optima. We have run experiments using the complete conditional distribution and confirmed that this issue prevents viable trajectory planning during learning even for systems as simple as an under-actuated pendulum. To



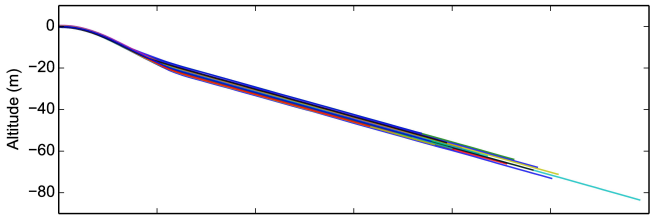
(a) Benchmark tasks

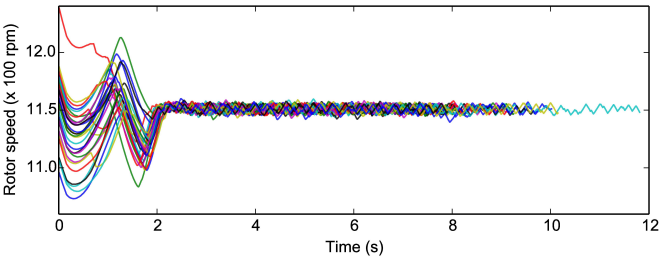


(b) Clusters during pendulum learning

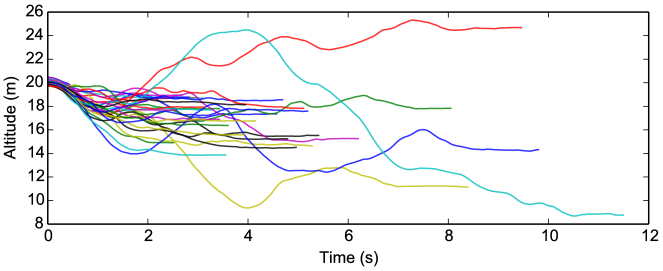


(c) Helicopter hover task diagram





(d) Helicopter learning autorotation



(e) Helicopter learning hover

Fig. 1: Illustrations of the pendulum, cartpole, and double pendulum benchmarks (top left), the DPMM 2 and 4 seconds into learning the pendulum task (middle left), visualization of the simulated helicopter in the beginning, middle, and end of learning to hover (bottom left), the helicopter altitude over multiple trials of the hovering task (top right) and the helicopter altitude and rotor speed over multiple trials of the autorotation task (middle and bottom right).

address this difficulty, we use an approximation scheme to produce a unimodal conditional distribution. Let $p(k \mid x, u)$ be the posterior cluster assignments computed from the model using Bayes' theorem:

$$p(k \mid x, u) = \frac{p(x, u \mid \tau_k)p(k)}{\sum_l p(x, u \mid \tau_l)p(l)}$$

$$p(k) := -\log(1 + b_k/a_k) - \sum_{l=k+1}^{K} \log(a_l/b_l + 1).$$

According to this model, the true conditional density is given by:

$$p(\dot{x} \mid x, u) = \sum_k p(k \mid x, u)p(\dot{x} \mid x, u, \tau_k).$$

We approximate this conditional by $p(\dot{x} \mid x, u, \tau(x, u))$ where $\tau(x, u) = \sum_k p(k \mid x, u)\tau_k$. This approximation is justified by its connection to weighted least squares which is made explicit by expressing $\tau(x, u)$ in terms of the sufficient statistics of the training data:

$$\tau(x, u) = \lambda + \sum_{k,n} p(k \mid x, u)\varphi_{k,n}T(z_n), \tag{5}$$

where $\varphi_{k,n}$ is prescribed by Equation (3).

The resulting approximate prediction is the same as the one we would obtain by weighted Bayesian linear regression with factored weights $v_n = \sum_k p(k \mid x, u)\varphi_{k,n}$. Average predictions under this model correspond to predictions made by weighted least squares with these weights. Weighted least squares is generally considered robust and accurate but computationally expensive [22], since it normally must access the entire training set for each prediction. The approximation is correct when the state-control pair (x, u) for which we are predicting dynamics lies on a single cluster, that is there exists one k^* such that $p(k^* \mid x, u) = 1$ but in general the approximation is not consistent. In our approach, the Dirichlet process mixture model can be regarded as a form of data compression that makes weighted least squares computationally tractable.

Our approximate predictive density models the previously introduced stacked vector $z = [\dot{x}, x, u]$ as being drawn from a Gaussian distribution with a normal-inverse-Wishart prior on the parameters μ, Σ. We end up with hyperparameters Ψ, m, ν, n summarized by $\tau(x, u)$ which depend on x, u

	pendulum	cartpole	double pendulum	helicopter hover	autorotation
our method	3.9 ± 1s	10 ± 3s	17 ± 7s	5 ± 2s	8.0 ± 1s
previous best	12s [23]	17.5s [23]	50s [23]	$30 - 60$ min [24], 32s with PILCO	-

TABLE I: Summary of the total interaction times required to learn each task using our method, as well as the best previously reported interaction times. PILCO results for the helicopter were obtained using the author-provided source code. We simulated the benchmark systems (pendulum, cartpole, double pendulum) using the same dynamics and parameters as the competing approach [23]. For the helicopter tasks we used the dynamics and parameters described in prior work [25], [26]. Success rate for each task was 100% based on at least 20 different random starting conditions for each system. The full set of parameters is available with the code used to run our experiments at: `http://rll.berkeley.edu/odens`.

through the predictive cluster responsibilities $p(k \mid x, u)$:

$$\begin{aligned} \Sigma &\sim \mathcal{W}^{-1}(\Psi, \nu) \\ \mu \mid \Sigma &\sim \mathcal{N}(m, \Sigma/n) \\ z \mid \mu, \Sigma &\sim \mathcal{N}(\mu, \Sigma) \\ \tau(x,u) &= [nm, \Psi + nmm^T, n, \nu + 2 + d_z] \\ &= \lambda + \sum_{k,n} p(k \mid x, u) \varphi_{k,n} T(z_n). \end{aligned} \tag{6}$$

The parameter μ can be partitioned conformably with z as $\mu = [\mu_{\dot{x}}, \mu_x, \mu_u]$. Our dynamics estimator consists of the conditional random variable $\mu_{\dot{x}} \mid \tau(x,u), x$. Explicitly, $E[f(x,u)] = E[\mu_{\dot{x}} \mid \tau(x,u), x, u]$ and $\mathrm{Cov}[f(x,u)] = \mathrm{Cov}[\mu_{\dot{x}} \mid \tau(x,u), x, u]$. This estimator captures only model *ambiguity* resulting from incomplete knowledge of the system's dynamics. Asymptotically, as more data becomes available, this ambiguity vanishes, ensuring that our algorithm transitions smoothly between exploration and exploitation. Note that we use the conditional distribution of the mean $\mu_{\dot{x}} \mid \tau(x,u), x, u$, rather than the conditional distribution of the observed state time derivative $\dot{x} \mid \tau(x,u), x, u$, since the latter includes observation noise that will persist asymptotically while the former excludes observation noise.

VI. Experiments

We first evaluate our method on a set of benchmark problems: the pendulum, cartpole, and double pendulum, shown in Figure 1. These benchmarks provide a comparison with previous state-of-the-art methods, and are all designed to test the ability to plan into the future, since each task initially requires moving away from the goal before the goal can be reached. For example, the pendulum must swing back and forth several times before it is able to reach an upright position. The metric for evaluating performance on these benchmarks is the amount of interaction time required to stabilize the system at a goal state. Previous methods typically used a smooth shaping cost function to smooth out the task and guide the algorithm towards the goal [23]. However, our MPC formulation allows us to directly use trajectory duration as the cost, without shaping, by using the cost rate $q(\dot{x}, x, u) = 1$. Consequently, the tasks presented in this section were formulated as shortest time to goal planning problems.

The previous state-of-the-art results on these problems were achieved by PILCO, which uses a Gaussian process with an episodic formulation to learn the dynamics [3], resetting the state between episodes. As shown in Table I, our method is able to learn each of the benchmark tasks with 2 to 3 times less interaction time and without any resets.

In addition to the benchmarks, we evaluated our algorithm on several helicopter control tasks, based on a simulator described in previous work [25]. In the first task, shown in Figure 1, the algorithm takes control of a helicopter mid-flight, at an altitude of 20m, and must simultaneously learn a model and stabilize the helicopter into a hover. Efficient learning is crucial for this task, since inability to stabilize quickly can result in loss of altitude and a crash. Successfully completing the task requires learning to control the helicopter in a single flight, as no resets are available. The best algorithm submitted to the 2008 Reinforcement Learning Competition for a similar, but not identical, helicopter hovering task required about 30 - 60 minutes of interaction and the use of resets, though other variants required as little as 10 minutes when provided with a good prior policy [24]. Using PILCO, we were able to obtain a hovering behavior after 32 seconds of interaction and 8 resets. Our algorithm was able to stabilize the helicopter in about 5 seconds without resets, losing 3.5m of altitude on average, and never more than 12m.

When comparing our results to those obtained with PILCO it is important to account for the differences between the two approaches. The objective of PILCO is to *learn a policy* for minimizing a given cost function so restarts are perforce required to evaluate the policy from different starting states and improve it. The objective of our approach is less stringent – we aim to *reach a goal state* quickly – which can be achieved with less system interaction, no restarts and no need to learn a policy.

While learning level flight is already a challenging task, a major strength of sample-efficient reinforcement learning is its ability to perform adaptive control when the dynamics change unexpectedly. To evaluate this condition, we simulated an engine failure and used our algorithm to learn autorotation, which is a technique for stabilizing unpowered flight by using the flow of air over the blades to simultaneously stabilize rotor speed and descent velocity. Details on this maneuver, along with a model of its dynamics, can be found in prior work [25]. Our algorithm was provided with the speed of the rotor, the dynamics of the helicopter during normal flight, and the target state. Model learning consisted of learning the relationship between the rotor speed and the

other state variables and controls. Our method was able to achieve a steady controlled descent after 8 seconds of system interaction.

Besides sample efficiency, the asymptotic computational efficiency of our algorithm also improves on the Gaussian process used by PILCO, since with a suitable online learning method, the complexity of training the DPMM grows as $O(n \log n)$ in the number of data points [8], and could be improved to be sublinear with an online algorithm [11]. However, the use of MPC for online replanning carries an additional computational cost. This cost does not increase with the number of data points, but is generally quite high. Our implementation required around 100s of computation per second of interaction time, of which 15s were spent updating the DPMM, and the remainder is spent on replanning the trajectory. As discussed in the next section, fast MPC is an active research area that could lead to large improvements in the running time of our method in the future.

VII. Discussion

We presented a data-efficient reinforcement learning algorithm based around online trajectory optimization with an optimistic exploration strategy that follows the principle of optimism in the face of uncertainty. Since this exploration strategy is driven by the uncertainty in the dynamics model, we chose a Bayesian dynamics model based on the Dirichlet process mixture model, which allows us to represent this uncertainty efficiently and accurately. By using model-predictive control to replan the trajectory online, our method is able to quickly respond to changes in this model as new transitions are observed. Together, these components allow our approach to achieve state-of-the-art sample efficiency on a set of benchmark problems, and to learn two challenging helicopter tasks mid-flight.

Our exploration method is based on previous work on discrete MDPs [6] and linear systems [4] where the strategy we use has provable sample complexity bounds. Although these theoretical guarantees do not immediately extend to arbitrary nonlinear systems, we empirically demonstrate that our strategy achieves very good results in an online setting and without any shaping of the cost function. In contrast, most model-based reinforcement learning algorithms for continuous systems use a smooth cost function to shape the task and encourage exploration [3], [10]. While such cost functions can be easily constructed in practice with basic knowledge about the task, it is typically not clear if they serve an essential role in the exploration component or they are merely helping the planner escape local optima.

Very little system interaction is required to successfully execute each task. However, its computational requirements are considerable. The computational cost stems primarily from the use of model-predictive control, which requires repeatedly solving a nonlinear optimal control problem. MPC allows our method to rapidly adapt to changes in the model and deploy the limited exploration budget more efficiently, but at a high cost in computation time. Recent progress in model-predictive control suggests that real-time MPC methods even for complex, high-dimensional problems, may be feasible in the near future, either with algorithmic improvements, parallelism, or specialized hardware [27], [28], [29]. Integrating these more advanced MPC methods with our exploration approach is a promising direction for future work that can make fast, sample-efficient RL a viable option for real time adaptive control applications.

Although we chose specific methods for exploration, modeling, and control, our approach is modular and extendible. For instance, the previously mentioned fast MPCs techniques could be readily incorporated into our method with additional action dimensions for exploration, and any model that can provide estimates of the uncertainty in the dynamics could be combined with our approach. Further exploring the choice of each of these components is also an interesting direction for future work.

Our helicopter control results suggest that our method can be successful at adaptive control tasks, where the algorithm must quickly "take command" of an unfamiliar dynamical system. Such methods are especially important for handling unexpected situations in safety-critical systems such as aircraft, where a change in the model dynamics due to failure might require a rapid correction [30]. Our autorotation example is one such situation, and we show that our algorithm can recover gracefully and reliably. Explicitly combining our approach with a principled handling of safety constraints could lead to very robust adaptive control methods for safety-critical applications in the future.

Acknowledgements

This material is based upon work supported in part by the Army Research Office under the MAST program and by the Office of Naval Research under contract/grant number N00014-11-1-0688.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotic Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] K. Astrom and B. Wittenmark, *Adaptive control*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[3] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning*, L. Getoor and T. Scheffer, Eds. Bellevue, Washington, USA: Omnipress, 2011, pp. 465–472.

[4] Y. Abbasi-Yadkori, C. Szepesvári, S. Kakade, and U. V. Luxburg, "Regret bounds for the adaptive control of linear quadratic systems," in *Proceedings of the 24th Annual Conference on Learning Theory*, 2011.

[5] S. Kuindersma, R. Grupen, and A. Barto, "Variational Bayesian optimization for runtime risk-sensitive control," in *Robotics: Science and Systems VIII (RSS)*, Sydney, Australia, July 2012.

[6] M. Araya, O. Buffet, and V. Thomas, "Near-optimal BRL using optimistic local transitions," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ser. ICML '12, J. Langford and J. Pineau, Eds. New York, NY, USA: Omnipress, Jul. 2012, pp. 97–104.

[7] P. L. B. Ambuj Tewari, "Optimistic linear programming gives logarithmic regret for irreducible MDPs," in *Proceedings of Neural Information Processing Systems Conference*, 2007.

[8] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Analysis*, vol. 1, no. 1, pp. 121–143, Mar. 2006.

[9] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao, "Direct trajectory optimization and costate estimation via an orthogonal collocation method," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1435–1440, Nov. 2006.

[10] M. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and trends in robotics*, vol. 2, no. 1-2, pp. 1–142, 2013.

[11] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan, "Streaming variational Bayes," in *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013.

[12] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration-exploitation tradeoff using variance estimates in multi-armed bandits," *Theoretical Computer Science*, vol. 410, no. 19, pp. 1876–1902, Apr. 2009.

[13] Y. W. Teh, "Dirichlet process," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer, 2010, pp. 280–287.

[14] M. Morari and J. H Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[15] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Society for Industrial & Applied Mathematics, 2010, vol. 19.

[16] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practicea survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[17] E. F. Camacho and C. Bordons, *Model predictive control*. Springer Berlin, 1999, vol. 303.

[18] A. Aswani and P. Bouffard, "Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter," in *Proc. American Control Conference (ACC)*, 2012.

[19] F. Palacios-Gomez, L. Lasdon, and M. Engquist, "Nonlinear optimization by successive linear programming," *Management Science*, vol. 28, no. 10, pp. 1106–1120, 1982.

[20] S. M. Khansari-Zadeh and A. Billard, "BM: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models," in *International Conference on Robotics and Automation (ICRA)*, 2010.

[21] S. P. Chatzis, D. Korkinof, and Y. Demiris, "A nonparametric Bayesian approach toward robot learning by demonstration," *Robotics and Autonomous Systems*, vol. 60, no. 6, pp. 789–802, Jun. 2012.

[22] T. Strutz, *Data Fitting and Uncertainty: A practical introduction to weighted least squares and beyond*. Vieweg+Teubner Verlag, 2010.

[23] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, 2013.

[24] R. Koppejan and S. Whiteson, "Neuroevolutionary reinforcement learning for generalized helicopter control," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 145–152.

[25] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, Jun. 2010.

[26] P. Abbeel, A. Coates, T. Hunter, and A. Ng, "Autonomous autorotation of an RC helicopter," *Experimental Robotics*, pp. 385–394, 2009.

[27] I. Manchester, U. Mettin, F. Iida, and R. Tedrake, "Stable dynamic walking over uneven terrain," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 265–279, 2011.

[28] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model-predictive control of humanoid robots," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[29] S. Vichik and F. Borrelli, "Solving linear and quadratic programs with an analog circuit," *Computers and Chemical Engineering*, 2014.

[30] M. Bodson and J. E. Groszkiewicz, "Multivariable adaptive algorithms for reconfigurable flight control," *IEEE Transactions on Control Systems Technology*, vol. 5, no. 2, pp. 217–229, 1997.