Efficient Reinforcement Learning for Motor Control

Marc Peter Deisenroth* and Carl Edward Rasmussen
Department of Engineering, University of Cambridge
Trumpington Street, Cambridge CB2 1PZ, UK

Abstract— **Artificial learners often require many more trials than humans or animals when learning motor control tasks in the absence of expert knowledge. We implement two key ingredients of biological learning systems, generalization and incorporation of uncertainty into the decision-making process, to speed up artificial learning. We present a coherent and fully Bayesian framework that allows for efficient artificial learning in the absence of expert knowledge. The success of our learning framework is demonstrated on challenging nonlinear control problems in simulation and in hardware.**

I. Introduction

Learning from experience is a fundamental characteristic of intelligence and holds great potential for artificial systems. Computational approaches for artificial learning from experience are studied in reinforcement learning (RL) and adaptive control. Although these fields have been studied for decades, the rate at which artificial systems learn still lags behind biological learners with respect to the amount of experience required to solve a task. Experience can be gathered by direct interaction with the environment.

Traditionally, learning even relatively simple control tasks from scratch has been considered "daunting" [14] in the absence of strong task-specific prior assumptions. In the context of robotics, one popular approach employs knowledge provided by a human "teacher" to restrict the solution space [1], [3], [14]. However, expert knowledge can be difficult to obtain, expensive, or simply not available. In the context of control systems and in the absence of strong task-specific knowledge, artificial learning algorithms often need more trials than physically feasible.

In this paper, we propose a principled way of learning control tasks without any expert knowledge or engineered solutions. Our approach mimics two fundamental properties of human experience-based learning. The first important characteristic of humans is that we can *generalize* experience to unknown situations. Second, humans explicitly model and incorporate *uncertainty* into their decisions [6]. We present a general and fully Bayesian framework for efficient RL by coherently combining generalization and uncertainty representation.

Unlike for discrete domains [10], generalization and incorporation of uncertainty into the decision-making process are not consistently combined in RL although heuristics exist [1]. In the context of motor control, generalization typically requires a model or a simulator, that is, an internal representation of the system dynamics. Since our objective is to reduce the interactions with the real system needed to successfully learn a motor control task, we have to face the problem of sparse data. Thus, we explicitly require a *probabilistic* model to additionally represent and quantify uncertainty. For this purpose, we use flexible non-parametric probabilistic Gaussian process (GP) models to extract valuable information from data and to reduce model bias.

II. General Setup

We consider discrete-time control problems with continuous-valued states $\mathbf{x}$ and external control signals (actions) $\mathbf{u}$. The dynamics of the system are described by a Markov decision process (MDP), a computational framework for decision-making under uncertainty. An MDP is a tuple of four objects: the state space, the action space (also called the control space), the one-step transition function f, and the immediate cost function $c(\mathbf{x})$ that penalizes the distance to a given target $\mathbf{x}_{\text{target}}$. The deterministic transition dynamics

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{1}$$

are not known in advance. However, in the context of a control problem, we assume that the immediate cost function $c(\,\cdot\,)$ can be chosen given the target $\mathbf{x}_{\text{target}}$.

The goal is to find a *policy* π^* that minimizes the expected long-term cost

$$V^\pi(\mathbf{x}_0) = \sum_{t=0}^{T} \mathbb{E}_{\mathbf{x}_t}\left[c(\mathbf{x}_t)\right] \tag{2}$$

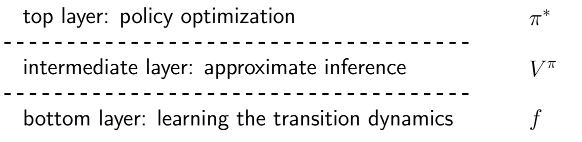
of following a policy π for a finite horizon of T time steps. The function V^π is called the *value function*, and $V^\pi(\mathbf{x}_0)$ is called the value of the state $\mathbf{x}_0$ under policy π.

A policy π is defined as a function that maps states to actions. We consider stationary deterministic policies that are parameterized by a vector $\boldsymbol{\psi}$. Therefore, $\mathbf{u}_{t-1} = \pi(\mathbf{x}_{t-1}, \boldsymbol{\psi})$ and $\mathbf{x}_t = f(\mathbf{x}_{t-1}, \pi(\mathbf{x}_{t-1}, \boldsymbol{\psi}))$. Thus, a state $\mathbf{x}_t$ at time t depends implicitly on the policy parameters $\boldsymbol{\psi}$.

More precisely: In the context of motor control problems, we aim to find a good policy π that leads to a low expected long-term cost $V^\pi(\mathbf{x}_0)$ given an initial state distribution $p(\mathbf{x}_0)$. We assume that no task-specific expert knowledge is available. Furthermore, we desire to minimize interactions with the real system. The setup we consider therefore corresponds to an RL problem with very limited interaction resources.

We decompose the learning problem into a hierarchy of three sub-problems described in Figure 1. At the bottom level, a probabilistic model of the transition function f is

*MP Deisenroth acknowledges support by the German Research Foundation (DFG) through grant RA 1030/1-3 to CE Rasmussen.



top layer: policy optimization π^*

intermediate layer: approximate inference V^π

bottom layer: learning the transition dynamics f

Fig. 1. The learning problem can be divided into three hierarchical problems. At the bottom layer, the transition dynamics f are learned. Based on the transition dynamics, the value function V^π can be evaluated using approximate inference techniques. At the top layer, an optimal control problem has to be solved to determine a model-optimal policy π^*.

Algorithm 1 Fast learning for control

```
1:  set policy to random                          ▷ policy initialization
2:  loop
3:      execute policy                            ▷ interaction
4:      record collected experience
5:      learn probabilistic dynamics model        ▷ bottom layer
6:      loop                                      ▷ policy search
7:          simulate system with policy $\pi$     ▷ intermed. layer
8:          compute expected long-term cost for $\pi$, eq. (2)
9:          improve policy                        ▷ top layer
10:     end loop
11: end loop
```

learned (Section II-B). Given the model of the transition dynamics and a policy π, the expected long-term cost in equation (2) is evaluated. This *policy evaluation* requires the computation of the predictive state distributions $p(\mathbf{x}_t)$ for $t = 1, \dots, T$ (intermediate layer in Figure 1, Section II-C). At the top layer (Section II-D), the policy parameters $\boldsymbol{\psi}$ are optimized based on the result of the policy evaluation. This parameter optimization is called an *indirect policy search*. The search is typically non-convex and requires iterative optimization techniques. The policy evaluation and policy improvement steps alternate until the policy search converges to a local optimum. If the transition dynamics are given, the two top layers correspond to an optimal control problem.

A. High-Level Summary of the Learning Approach

A high-level description of the proposed framework is given in Algorithm 1. Initially, we set the policy to random (line 1). The framework involves learning in two stages: First, when interacting with the system (line 3) experience is collected (line 4) and the internal probabilistic dynamics model is updated based on both historical and novel observations (line 5). Second, the policy is refined in the light of the updated dynamics model (loop over lines 7–9) using approximate inference and gradient-based optimization techniques for policy evaluation and policy improvement, respectively. The model-optimized policy is applied to the real system (line 3) to gather novel experience (line 4).

The subsequent model update (line 5) accounts for possible discrepancies between the predicted and the actually encountered state trajectory. With increasing experience, the probabilistic model describes the dynamics well in regions of the state space that are promising, that is, regions along trajectories with low expected cost.

B. Bottom Layer: Learning the Transition Dynamics

We learn the short-term transition dynamics f in equation (1) with Gaussian process models [13]. A GP can be considered a distribution over functions and is utilized for state-of-the-art Bayesian non-parametric regression [13]. GP regression combines both flexible non-parametric modeling and tractable Bayesian inference.

The GP dynamics model takes as input a representation of state-action pairs $(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. The GP targets are a representation of the consecutive states $\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. The dynamics models are learned using evidence maximization [13]. A key advantage of GPs is that a parametric structure of the underlying function does not need to be known. Instead, an adaptive, probabilistic model for the latent transition dynamics is inferred directly from observed data. The GP also consistently describes the uncertainty about the model itself.

C. Intermediate Layer: Approximate Inference for Long-Term Predictions

For an arbitrary pair $(\mathbf{x}_*, \mathbf{u}_*)$, the GP returns a Gaussian predictive distribution $p(f(\mathbf{x}_*, \mathbf{u}_*))$. Thus, when we simulate the probabilistic GP model forward in time, the predicted states are uncertain. We therefore need to be able to predict successor states when the current state is given by a probability distribution. Generally, a Gaussian distributed state followed by nonlinear dynamics (as modeled by a GP) results in a non-Gaussian successor state. We adopt the results from [4], [11] and approximate the true predictive distribution by a Gaussian with the same mean and the same covariance matrix (exact moment matching). Throughout all computations, we explicitly take the model uncertainty into account by averaging over all plausible dynamics models captured by the GP. To predict a successor state, we average over both the uncertainty in the current state *and* the uncertainty about the possibly imprecise dynamics model itself. Thus, we reduce model bias, which is one of the strongest arguments against model-based learning algorithms [2], [3], [16].

Although the policy is deterministic, we need to consider *distributions* over actions: For a single deterministic state $\mathbf{x}_t$, the policy will deterministically return a single action. However, during the forward simulation, the states are given by a probability distribution $p(\mathbf{x}_t)$, $t = 0, \dots, T$. Therefore, we require the predictive distribution $p(\pi(\mathbf{x}_t))$ over actions to determine the distribution $p(\mathbf{x}_{t+1})$ of the consecutive state. We focus on nonlinear policies π represented by a radial basis function (RBF) network. Therefore,

$$\pi(\mathbf{x}_*) = \sum_{i=1}^{N} \beta_i \phi_i(\mathbf{x}_*) \,, \tag{3}$$

where the basis functions ϕ are axis-aligned Gaussians centered at $\boldsymbol{\mu}_i$, $i = 1, \dots, N$. An RBF network is equivalent to the mean function of a GP or a Gaussian mixture model. In short, the policy parameters $\boldsymbol{\psi}$ of the RBF policy are the locations $\boldsymbol{\mu}_i$ and the weights β_i as well as the length-scales of the Gaussian basis functions ϕ and the amplitude

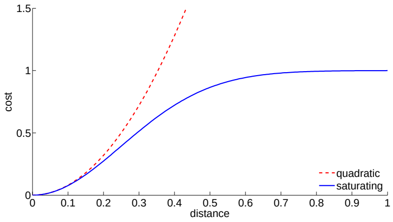


Fig. 2. Quadratic (red, dashed) and saturating (blue, solid) cost functions. The x-axis shows the distance of the state to the target, the y-axis shows the corresponding immediate cost. In contrast to the quadratic cost function, the saturating cost function can encode that a state is simply "far away" from the target. The quadratic cost function pays much attention to how "far away" the state really is.

of the latent policy π. The RBF policy in equation (3) allows for an analytic computation of a distribution over actions as required for consistent predictions.

The GP model for the transition dynamics and the RBF policy allow for the computation of the joint Gaussian probability distribution $p(\mathbf{x}_t, \mathbf{u}_t)$, which is required to compute the distribution $p(\mathbf{x}_{t+1})$ of the consecutive state via moment matching. By iteration, we can thus compute an approximate distribution of the state sequence $\mathbf{x}_0, \ldots, \mathbf{x}_T$, which is required to evaluate the expected long-term cost in equation (2).

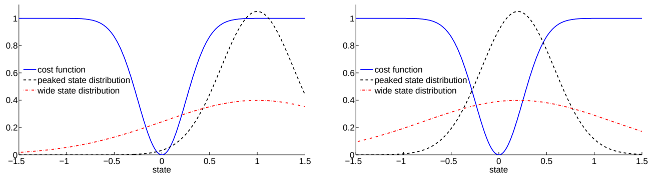
D. Top Layer: Policy Optimization

The optimization problem at the top layer in Figure 1 corresponds to finding policy parameters $\boldsymbol{\psi}^*$ that minimize the expected long term finite-horizon cost V^{π_ψ} in equation (2).

We employ a conjugate gradients minimizer, which requires the partial derivatives of the value function with respect to the policy parameters. These derivatives are computed analytically by repeated application of the chain rule [12]. Hence, our approach is a gradient-based policy search method.

III. Cost Function

We assume that the immediate cost function c in equation (2) does not incorporate any solution-specific knowledge such as penalties on the control signal or speed variables (in regulator problems). Only the target state $\mathbf{x}_{\text{target}}$ is given. An autonomous learner must be able to learn the remainder of the task by itself: If the system reaches the target state but overshoots due to too high velocities, the learning algorithm should account for this kind of failing in a next trial. We employ a cost function that solely uses a geometric distance d of the current state to the target state. Thus, overshooting causes higher long-term cost than staying close to the target.



(a) Initially, when the mean of the state is far away from the target, uncertain states (red, dashed-dotted) are preferred to more certain states with a more peaked distribution (black, dashed). This leads to initial exploration.

(b) Finally, when the mean of the state is close to the target, certain states with peaked distributions cause less expected cost and are therefore preferred to more uncertain states (red, dashed-dotted). This leads to exploitation once close to the target.

Fig. 3. Automatic exploration and exploitation due to the saturating cost function (blue, solid). The x-axes describe the state space. The target state is the origin.

A cost function commonly used in optimal control (particularly in combination with linear systems) is the quadratic cost (red, dashed in Figure 2). One problem with the quadratic cost function is that the long-term cost in equation (2) is highly dependent on the worst state along a predicted state trajectory. A second problem with the quadratic cost is that the expected cumulative cost in equation (2) is highly sensitive to details of a distribution that essentially encode that the model has lost track of the state. In particular in the early stages of learning, the predictive state uncertainty may grow rapidly with the time horizon. To avoid an extreme dependence on these arbitrary details, we instead use the cost function

$$c(\mathbf{x}) = 1 - \exp\big(-\tfrac{a^2}{2}\, d(\mathbf{x}, \mathbf{x}_{\text{target}})^2\big) \tag{4}$$

that is locally quadratic but which saturates at unity for large deviations d from the desired target $\mathbf{x}_{\text{target}}$ (blue function, solid, in Figure 2). In equation (4), the Euclidean distance from the state $\mathbf{x}$ to the target state is denoted by d, and the parameter a controls the width of the cost function. The saturating cost function in equation (4) resembles the cost function in human reasoning [5].

A. Exploration and Exploitation

The saturating cost function in equation (4) allows for natural exploration even if the policy is greedy, that is, it minimizes the expected long-term cost in equation (2). This property is illustrated in Figure 3. If the mean of a state distribution $p(\mathbf{x}_t)$ is far away from the target $\mathbf{x}_{\text{target}}$, a wide state distribution is more likely to have substantial tails in some low-cost region than a fairly peaked distribution (Figure 3(a)). If we initially start from a state distribution in a high-cost region, the saturating cost therefore leads to automatic exploration by favoring uncertain states.

If the mean of the state distribution is close to the target as in Figure 3(b), wide distributions are likely to have substantial tails in high-cost regions. By contrast, the mass of a peaked distribution is more concentrated in low-cost regions. In this case, a greedy policy prefers peaked distributions close to the target, which leads to exploitation.

Hence, even for a greedy policy, the combination of a probabilistic dynamics model and a saturating cost function leads to exploration as long as the states are far away from the target. Once close to the target, a greedy policy does not veer from a trajectory that lead the system to certain states close to the target.

One way to encourage further exploration is to modify the objective function in equation (2). Incorporation of the state uncertainty itself would be an option, but it would lead to extreme designs [8]. However, we are particularly interested in exploring promising regions of the state space, where "promising" is directly defined by the saturating cost function c in equation (4). Therefore, we consider the *variance of the predicted cost*, which can be computed analytically. To encourage goal-directed exploration, we therefore minimize the objective function

$$\tilde{V}^{\pi}(\mathbf{x}_0) = \sum_{t=0}^{T} \mathbb{E}_{\mathbf{x}}[c(\mathbf{x}_t)] + b\,\sigma_{\mathbf{x}}[c(\mathbf{x}_t)]\,. \tag{5}$$

Here, σ denotes the standard deviation of the predicted cost. For $b < 0$ uncertainty in the cost is encouraged, for $b > 0$ uncertainty in the cost is penalized.

What is the difference between the variance of the state and the variance of the cost? The variance of the predicted cost depends on the variance of the state: If the state distribution is fairly peaked, the variance of the corresponding cost is always small. However, an uncertain state does not necessarily cause a wide cost distribution: If the mean of the state distribution is in a high-cost region and the tails of the distribution do not substantially cover low-cost regions, the uncertainty of the predicted cost is very low. The only case the cost distribution can be uncertain is if a) the state is uncertain and b) a non-negligible part of the mass of the state distribution is in a low-cost region. Hence, using the uncertainty of the cost for exploration avoids extreme designs by exploring regions that might be close to the target.

Exploration-favoring policies ($b < 0$) do not greedily follow the most promising path (in terms of the expected cost (2)), but they aim to gather more information to find a better strategy in the long-term. By contrast, uncertainty-averse policies ($b > 0$) follow the most promising path and after finding a solution, they never veer from it. *If* uncertainty-averse policies find a solution, they presumably find it quicker (in terms of number of trials required) than an exploration-favoring strategy. However, exploration-favoring strategies find solutions more reliably. Furthermore, at the end of the day they often provide better solutions than uncertainty-averse policies.

IV. Results

Our proposed learning framework is applied to two under-actuated nonlinear control problems: the Pendubot [15] and the inverted pendulum. The under-actuation of both systems makes myopic policies fail. In the following, we exactly follow the steps in Algorithm 1.

A. Pendubot

The Pendubot depicted in Figure 4 is a two-link, under-actuated robot [15]. The first joint exerts torque, but the second joint cannot. The system has four continuous-valued state variables: two joint angles and two joint angular velocities. The angles of the joints, θ_1 and θ_2, are measured anti-clockwise from upright. An applied external torque $u \in [-3.5, 3.5]\,\mathrm{Nm}$ controls the first joint. In our simulation, the values for the masses and the lengths of the pendulums are $m_1 = 0.5\,\mathrm{kg} = m_2$ and $\ell_1 = 0.6\,\mathrm{m} = \ell_2$. The sampling frequency is set to $13.\bar{3}\,\mathrm{Hz}$, which is fairly low for this kind of problem: A sampling frequency of $2,000\,\mathrm{Hz}$ is used in [9].

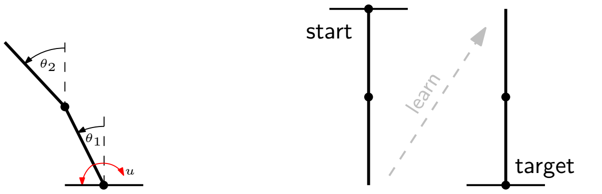


Fig. 4. Pendubot. The control task is to swing both links up and to balance them in the inverted position by exerting a torque to the first joint only.

Starting from the position, where both joints hang down, the objective is to swing the Pendubot up and to balance it in the inverted position. Note that the dynamics of the system can be chaotic.

Our cost function penalizes the Euclidean distance d from the tip of the outer pendulum to the target state.

The width $1/a = 0.5\,\mathrm{m}$ of in the cost function in equation (4) is chosen, such that the immediate cost is about unity as long as the distance between the pendulum tip and the target state is greater than the length ℓ_2 of the outer pendulum. Thus, the tip of the outer pendulum has to cross horizontal to significantly reduce the immediate cost from unity. Initially, we set the exploration parameter in equation (5) to $b = -0.2$ to favor more exploration in predicted high-reward regions of the state space. We increase the exploration parameter linearly, such that it reaches 0 in the last trial. The learning algorithm is fairly robust to the selection of the exploration parameter b in equation (5): In most cases, we could learn the tasks with $b \in [-0.5, 0.2]$.

Figure 5 sketches a solution to the Pendubot problem after an experience of approximately $90\,\mathrm{s}$. The learned controller attempts to keep the pendulums aligned, which, from a mechanical point of view, leads to a faster swing-up motion.

B. Inverted Pendulum (Cart-Pole)

The inverted pendulum shown in Figure 6 consists of a cart with mass m_1 and an attached pendulum with mass m_2 and length ℓ, which swings freely in the plane. The pendulum angle θ is measured anti-clockwise from hanging down. The cart moves horizontally on a track with an applied external force u. The state of the system is given by the position x and the velocity $\dot{x}$ of the cart and the angle θ and angular velocity $\dot{\theta}$ of the pendulum.

The objective is to swing the pendulum up and to balance it in the inverted position in the middle of the track by simply pushing the cart to the left and to the right.

We reported simulation results of this system in [12]. In this paper, we demonstrate our learning algorithm in real hardware. Unlike classical control methods, our algorithm learns a model of the system dynamics in equation (1) from data only. It is therefore not necessary to provide a probably inaccurate idealized mathematical description of the transition dynamics that includes parameters, such as friction, motor constants, or delays. Since $\ell \approx 125\,\mathrm{mm}$, we set

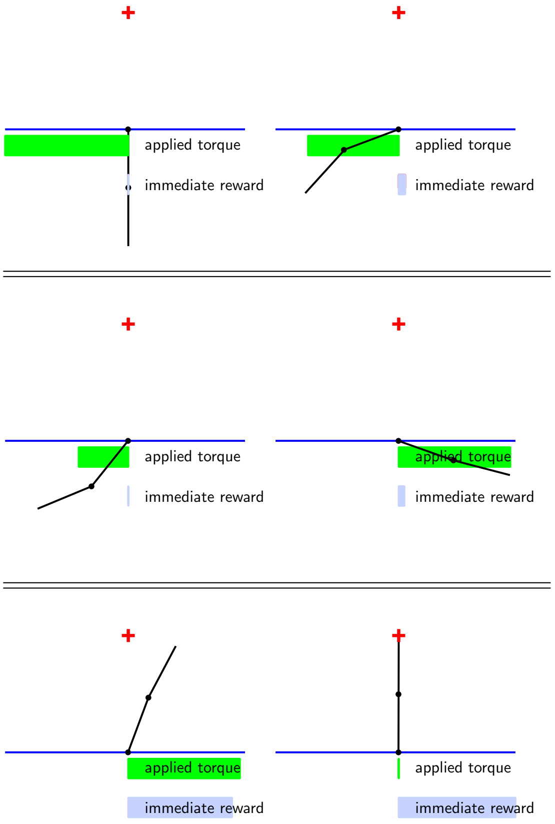


Fig. 5. Illustration of the learned Pendubot swing up. Six snapshots of the swing up (top left to bottom right) are shown. The cross marks the target state of the tip of the outer pendulum. The green bar shows the applied torque. The gray bar shows the immediate reward (negative cost plus 1). In order to swing the Pendubot up, energy is induced first, and the Pendubot swings left and then right up. Close to the target in the inverted position (red cross), the controller does no longer apply significant torques and keeps the Pendubot close to the target.

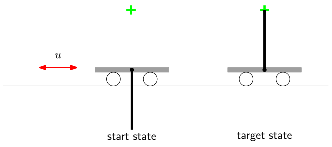


Fig. 6. Inverted pendulum. The task is to swing the pendulum up and to balance it in the inverted position in the middle of the track by applying horizontal forces to the cart only.

the sampling frequency to 10 Hz, which is about five times faster than the characteristic frequency of the pendulum. Furthermore, we choose the cost function in equation (4) with $1/a \approx 0.07\,\text{m}$, such that the cost incurred does not substantially differ from unity if the distance between the pendulum tip and the target state is greater than ℓ. The force is constrained to $u \in [-10, 10]\,\text{N}$.

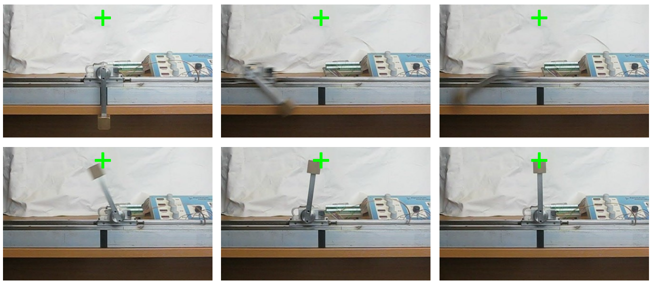


Fig. 7. Inverted pendulum in hardware; snapshots of a controlled trajectory. The pendulum is swung up and balanced in the inverted position close to the target state (green cross). To solve this task, our algorithm required only 17.5 s of interaction with the physical system.

Following Algorithm 1, we initialized the learning system with two trials of length $T = 2.5\,\text{s}$, where random actions (horizontal forces to the cart) were applied. The five seconds of data collected in these trials were used to train a first probabilistic dynamics model. Using this model to internally simulate the dynamics, the parameters of the RBF controller were optimized. In the third trial, this controller is applied to the real system. The controller manages to keep the cart in the middle of the track, but the pendulum does not go beyond horizontal—the system never experienced states where the pendulum is above horizontal. However, it takes the new observations into account and re-train the probabilistic dynamics model. With the uncertainty in the predictions decreases and a good policy for to the updated model is found. Applying this new policy for another 2.5 s leads to the fourth trial where the controller swings the pendulum up, but drastically overshoots. However, for the first time states close to the target state ware encountered. Taking these observations into account, the dynamics model is updated, and the corresponding controller is learned. In the fifth trial, the controller learned to reduce the angular velocity substantially since falling over leads to high expected cost. After two more trials, the learned controller can solve the cart-pole task based on a total of 17.5 s experience only. Figure 7 shows snapshots of a test trajectory of 20 s length. A video showing the entire learning process can be found at http://mlg.eng.cam.ac.uk/marc/.

Our learning algorithm is very general and worked immediately when we applied it to real hardware. Since we could derive all required parameters (width of the cost function and time sampling frequency) from the length of the pendulum, no parameter tuning was necessary.

V. Discussion

Our approach learns very fast in terms of the amount of experience required to solve a task. However, the current implementation requires about ten minutes of CPU time on a standard PC per policy search. The most demanding computations are the approximate inference based on moment matching and the computation of the derivatives, which require $\mathcal{O}(Tn^2D^3)$ operations. Here T is the prediction horizon, n the size of the dynamics training set, and D is the dimension of the state. Once the policy has been learned, the policy can be implemented and applied in real time (Section IV-B).

The model of the transition dynamics f in equation (1) is probabilistic, but the internal simulation is fully deterministic: For a given policy parameterization and an initial state distribution $p(\mathbf{x}_0)$ the approximate inference is deterministic and does not require any sampling. This property is still valid if the transition dynamics f and/or the policy π are stochastic. Due to the deterministic simulative model, any optimization method for deterministic functions can be employed for the policy search.

The algorithms directly generalizes to multiple actuators by using a policy in equation (3) with multivariate outputs. With this approach, we successfully applied our learning framework to the Pendubot task with two actuators (not discussed in this paper.)

We have demonstrated learning in the special case where we assume that the state is fully observable. In principle, there is nothing to hinder the use of the algorithm when observations are noisy. After learning a generative model for the latent transitions, the hidden state can be tracked using the GP-ADF filter proposed in [4]. The learning algorithm and the involved computations generalize directly to this setting.

Our experience is that the probabilistic GP dynamics model leads to fairly robust controllers: First, since the model can be considered a distribution over all models that plausibly explain the experience, incorporation of new experience does not usually make previously plausible models implausible. Second, the moment-matching approximations in the approximate inference is a conservative approximation of a distribution: Let q be the approximate Gaussian distribution that is computed by moment matching and p be the true predictive distribution, then we minimize $\mathrm{KL}(p||q)$. Minimizing $\mathrm{KL}(p||q)$ ensures that q is non-zero where the true distribution p is non-zero. This is an important issue in the context of coherent predictions and, therefore, robust control: The approximate distribution q is not overconfident, but can be too cautious since it captures all modes of the true distribution as shown by [7]. If we still can learn a controller using the admittedly conservative moment-matching approximation, the controller is expected to be robust.

If a deterministic dynamics model is used, incorporation of new experience can drastically change the model. We observed that this model change can have strong influence to the optimization procedure [12]. In case of the Pendubot, we have experimental evidence that the deterministic learning algorithm cannot explore the state space sufficiently well, that is, it never came close to the target state at all.

The general form of the saturating cost function in equation (4) can be chosen for arbitrary control problems. Therefore, it is not problem specific. However, it clearly favors the incorporation of uncertainty into dynamics models. Hence, it can be considered algorithm-specific.

Our learning algorithm does not require an explicit global model of the value function V^{π}, but instead evaluates the value function for an initial state distribution $p(\mathbf{x}_0)$. Although global value function models are often used to derive an optimal policy they are an additional source of errors. It is often unclear how an error in the value function affects to policy if the value function model is not exact.

VI. Conclusion

We proposed a general framework for efficient reinforcement learning in the context of motor control problems. The key ingredient of this framework is a probabilistic model for the transition dynamics, which mimics two important features of biological learners: the ability to generalize and the explicit incorporation of uncertainty into the decision-making process. We successfully applied our algorithm to the simulated Pendubot and the cart-pole problem in real hardware demonstrating the flexibility and the success of our approach. To our best knowledge, we report an unprecedented speed of learning for both tasks.

References

[1] Pieter Abbeel, Morgan Quigley, and Andrew Y. Ng. Using Inaccurate Models in Reinforcement Learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 1–8, Pittsburgh, PA, USA, June 2006.

[2] Christopher G. Atkeson and Juan C. Santamaría. A Comparison of Direct and Model-Based Reinforcement Learning. In *Proceedings of the International Conference on Robotics and Automation*, 1997.

[3] Christopher G. Atkeson and Stefan Schaal. Robot Learning from Demonstration. In *Proceedings of the 14th International Conference on Machine Learning*, pages 12–20, Nashville, TN, USA, July 1997. Morgan Kaufmann.

[4] Marc P. Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *Proceedings of the 26th International Conference on Machine Learning*, pages 225–232, Montreal, Canada, June 2009. Omnipress.

[5] Konrad P. Körding and Daniel M. Wolpert. The Loss Function of Sensorimotor Learning. In *Proceedings of the National Academy of Sciences*, volume 101, pages 9839–9842, 2004.

[6] Konrad P. Körding and Daniel M. Wolpert. Bayesian Decision Theory in Sensorimotor Control. *Trends in Cognitive Sciences*, 10(7):319–326, June 2006.

[7] Malte Kuss and Carl E. Rasmussen. Assessing Approximations for Gaussian Process Classification. In *Advances in Neural Information Processing Systems 18*, pages 699–706. The MIT Press, Cambridge, MA, USA, 2006.

[8] David J. C. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4:590–604, 1992.

[9] Rowland O'Flaherty, Ricardo G. Sanfelice, and Andrew R. Teel. Robust Global Swing-Up of the Pendubot Via Hybrid Control. In *Proceedings of the 2008 American Control Conference*, pages 1424–1429, Seattle, WA, USA, June 2008.

[10] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An Analytic Solution to Discrete Bayesian Reinforcement Learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 697–704, Pittsburgh, PA, USA, 2006. ACM.

[11] Joaquin Quiñonero-Candela, Agathe Girard, Jan Larsen, and Carl E. Rasmussen. Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 701–704, April 2003.

[12] Carl E. Rasmussen and Marc P. Deisenroth. *Recent Advances in Reinforcement Learning*, volume 5323 of *Lecture Notes in Computer Science*, chapter Probabilistic Inference for Fast Learning in Control, pages 229–242. Springer-Verlag, November 2008.

[13] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2006.

[14] Stefan Schaal. Learning From Demonstration. In *Advances in Neural Information Processing Systems 9*, pages 1040–1046. The MIT Press, Cambridge, MA, USA, 1997.

[15] Mark W. Spong and Daniel J. Block. The Pendubot: A Mechatronic System for Control Research and Education. In *Proceedings of the Conference on Decision and Control*, pages 555–557, 1995.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 1998.