ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Model-Based Reinforcement Learning with Continuous States and Actions

Marc P. Deisenroth[1*], Carl E. Rasmussen[1,2], and Jan Peters[2]

1- University of Cambridge - Department of Engineering
Trumpington Street, Cambridge CB2 1PZ - UK
{mpd37|cer54}@cam.ac.uk

2- Max Planck Institute for Biological Cybernetics
Spemannstraße 38, 72070 Tübingen - Germany
jan.peters@tuebingen.mpg.de

Abstract.
Finding an optimal policy in a reinforcement learning (RL) framework with continuous state and action spaces is challenging. Approximate solutions are often inevitable. GPDP is an approximate dynamic programming algorithm based on Gaussian process (GP) models for the value functions. In this paper, we extend GPDP to the case of unknown transition dynamics. After building a GP model for the transition dynamics, we apply GPDP to this model and determine a continuous-valued policy in the entire state space. We apply the resulting controller to the underpowered pendulum swing up. Moreover, we compare our results on this RL task to a nearly optimal discrete DP solution in a fully known environment.

1 Introduction

In reinforcement learning (RL) an agent must learn good decision policies based on observations or trial-and-error interactions with a dynamic environment. Dynamic programming (DP) is a common methodology for achieving this task by solving the Bellman equation, which characterizes properties of the value function. Usually, standard table-based algorithms for discrete setups do not straightforwardly apply to continuous state and action domains. Function approximators can generalize the value function to continuous-valued state spaces, but usually they are limited to discrete action domains [1]. In case of non-probabilistic, parametric function approximation we are restricted to a fixed class of functions and might run into problems in case of noisy data. A state-of-the-art nonparametric Bayesian regression method is provided by the Gaussian process (GP) framework [2]. Model-based policy iteration in continuous state and action spaces based on value function evaluation using GPs is presented in [3]. In [4], model-free policy iteration is proposed within the GP framework to perform both policy evaluation and policy improvement. Gaussian process dynamic programming (GPDP) is a model-based dynamic programming algorithm for fully known dynamics in which the value functions are modeled by GPs [5].

*M. P. Deisenroth is supported by the German Research Foundation (DFG) through grant RA 1030/1.

In this paper, we extend GPDP to the case of unknown deterministic dynamics by using a learned system model. Moreover, we compare the performance of the resulting policy to the policy of a benchmark controller using exact dynamics.

2 Reinforcement Learning with Unknown Dynamics

Uncertainty is a key property of RL. Modeling uncertain functions properly is an extremely hard problem in general. GP regression is a powerful Bayesian method to model latent functions without being restricted to a specific parametric function class, such as polynomials. Thus, throughout this paper, we model latent functions by means of GPs that generalize latent functions from a small, finite training set to the entire continuous-valued space. Moreover, confidence intervals are provided.

We consider the undiscounted finite-horizon RL problem of finding a policy π^* that minimizes the long-term loss $g_{\text{term}}(\mathbf{x}_N) + \sum_{k=0}^{N-1} g(\mathbf{x}_k, \mathbf{u}_k)$ where k indexes discrete time. Here, g is the immediate loss that depends on the state $\mathbf{x} \in \mathbb{R}^{n_x}$ and a chosen control action $\mathbf{u} = \pi(\mathbf{x}) \in \mathbb{R}^{n_u}$. A state-dependent terminal loss is denoted by g_{term}.

Example: Setup
Throughout this paper, we use the underpowered pendulum swing up as running example. Initially the pendulum is hanging down. The goal is to swing the pendulum up and to balance it in the inverted position. This task has previously been considered a hard problem [6]. We assume pendulum dynamics following the ODE

$$\ddot{\varphi}(t) = \frac{-\mu\dot{\varphi}(t) + mgl\sin(\varphi(t)) + u(t)}{ml^2} \tag{1}$$

where $\mu = 0.05\,\text{kg m}^2/\text{s}$ is the coefficient of friction. The applied torque is restricted to $u \in [-5, 5]\,\text{Nm}$. The characteristic pendulum frequency is approximately $2\,\text{s}$. Angle and angular velocity are denoted by φ and $\dot{\varphi}$ and given in rad and rad/s, respectively. The system can be influenced by applying a force u any $200\,\text{ms}$. The pendulum dynamics (1) are discretized in time with $200\,\text{ms}$ between two samples. The immediate loss g is $g(\mathbf{x}_k, u_k) = \mathbf{x}_k^T \operatorname{diag}([1, 0.2])\mathbf{x}_k + 0.1\,u_k^2$, the optimization horizon is $2\,\text{s}$.

2.1 Learning System Dynamics

In the considered RL setup we assume a priori unknown deterministic system dynamics. If possible, it seems worth estimating a dynamics model since, intuitively, model-based methods make better use of available information [1]. In the first step, we attempt to model the system based on observations of sampled trajectories. We consider discrete-time systems of the form $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$. We use a Gaussian process model, the *dynamics GP*, to model the dynamics and write $f \sim \mathcal{GP}_f$. For each output dimension of f the GP model is fully specified by its mean and covariance functions revealing prior beliefs about the latent function [2]. For any new input $(\mathbf{x}_*, \mathbf{u}_*)$ the predictive distribution of $f(\mathbf{x}_*, \mathbf{u}_*)$ conditioned on the training data is Gaussian with mean vector $\boldsymbol{\mu}_*$ and covariance matrix $\boldsymbol{\Sigma}_*$. The posterior GP reveals the remaining uncertainty

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Algorithm 1 GPDP using system model $\mathcal{GP}_f$

1: **input:** $\mathcal{GP}_f, \mathcal{X}, \mathcal{U}$
2: $V_N^*(\mathcal{X}) = g_{\text{term}}(\mathcal{X})$ ▷ terminal loss
3: $V_N^*(\,\cdot\,) \sim \mathcal{GP}_v$ ▷ GP model for V_N^*
4: **for** $k = N-1$ to 0 **do** ▷ DP recursion (in time)
5: **for** all $\mathbf{x}_i \in \mathcal{X}$ **do** ▷ for all states
6: $Q_k(\mathbf{x}_i, \mathcal{U}) = g(\mathbf{x}_i, \mathcal{U}) + \mathrm{E}[V_{k+1}^*(\mathbf{x}_{k+1}) | \mathbf{x}_i, \mathcal{U}, \mathcal{GP}_f]$
7: $Q_k(\mathbf{x}_i, \,\cdot\,) \sim \mathcal{GP}_q$ ▷ GP model for Q
8: $\pi_k^*(\mathbf{x}_i) \in \arg\min_{\mathbf{u}} Q_k(\mathbf{x}_i, \mathbf{u})$
9: $V_k^*(\mathbf{x}_i) = Q_k\big(\mathbf{x}_i, \pi_k^*(\mathbf{x}_i)\big)$
10: **end for**
11: $V_k^*(\,\cdot\,) \sim \mathcal{GP}_v$ ▷ GP model for V_k^*
12: **end for**
13: **return** $\pi_0^*(\mathcal{X})$ ▷ return set of optimal actions

about the underlying latent function f. In the limit of infinite data this uncertainty tends to zero and the GP model converges to the deterministic system, such that $\mathcal{GP}_f = f$.

Example: Learning the pendulum dynamics
In case of the underpowered pendulum swing up, the standard deviation of the model is smaller than 0.03 for 400 training examples. More training points increase the confidence. The absolute error between the underlying dynamics f and the mean of the GP model $\mathcal{GP}_f$ is smaller than 0.04.

2.2 Application of Gaussian Process Dynamic Programming

Using the dynamics GP to describe the latent system dynamics f, we apply GPDP to derive optimal actions based on finite training sets $\mathcal{X}$ of states and $\mathcal{U}$ of actions. The elements of the training sets are randomly distributed within their domains. GPDP generalizes dynamic programming to continuous state and action domains. A sketch of GPDP is given in Algorithm 1. We model both latent functions V^* and Q by means of Gaussian processes. For each $\mathbf{x} \in \mathcal{X}$ we use independent GP models for $Q(\mathbf{x}, \,\cdot\,)$ rather than modeling $Q(\,\cdot\,, \,\cdot\,)$ in joint state-action space. This idea is largely based on two observations. First, a good model of Q in joint state-action space requires substantially more training points and makes standard GP models computationally very expensive. Second, the Q-function can be discontinuous in $\mathbf{x}$, as well as in $\mathbf{u}$ direction. We eliminate one possible source of discontinuity by treating $Q(\mathbf{x}_i, \,\cdot\,)$ and $Q(\mathbf{x}_j, \,\cdot\,)$ as independent.

To determine the Q-value in line 6 of the GPDP algorithm, we have to solve an integral of the form

$$\mathrm{E}[V^*(f(\mathbf{x}, \mathbf{u}))|\mathbf{x}, \mathbf{u}] = \iint V^*(f(\mathbf{x}, \mathbf{u})) p(f(\mathbf{x}, \mathbf{u})|\mathbf{x}, \mathbf{u}) p(V^*) \mathrm{d}f \mathrm{d}V^* \qquad (2)$$

where both the system function f and the value function V^* are modeled by $\mathcal{GP}_f$ and $\mathcal{GP}_v$, respectively. Therefore, the value of the integral is a random

variable. The GP model of V^* with squared exponential (SE) covariance function and the Gaussian predictive distribution $p(f(\mathbf{x},\mathbf{u})|\mathbf{x},\mathbf{u})$, provided by the system model, allow us to determine a distribution of the uncertain integral value (2) by applying the Bayesian Monte Carlo method [7]. We have to integrate over both sources of uncertainty the model uncertainty of $\mathcal{GP}_f$ and the uncertainty about V^*. However, if the model uncertainty of $\mathcal{GP}_f$ tends to zero (many data), the integrand of (2) tends to $V^*(f(\mathbf{x},\mathbf{u}))p(V^*)$, and the distribution of (2) equals the distribution of V^*. In this case, $\mathrm{E}[V^*(f(\mathbf{x},\mathbf{u}))] = m_v(m_f(\mathbf{x},\mathbf{u}))$.

2.3 Determination of a Closed-Loop Policy

We extend the policy from a finite training set to the entire state space as follows. The actions $\pi_0^*(\mathcal{X})$ of the training set $\mathcal{X}$ returned by GPDP are regarded as (uncertain) measurements of an underlying optimal policy. We attempt to model the latent optimal policy by means of a GP. Depending on the loss function and the system dynamics, the choice of an appropriate covariance function is extremely difficult. We approach the problem from a different direction: In applications of control algorithms to real underpowered robots, smoothness of actions is desired to protect the actuators. Thus, we assume that a close-to-optimal policy for an underpowered system is at least piecewise smooth. Possible discontinuities appear at boundaries of a manifold where the sign of the control signal changes.

With this assumption we attempt to model the latent policy π^* with *two* Gaussian processes. One GP is trained only on the subset $\pi_+^*(\mathcal{X}) \subset \pi_0^*(\mathcal{X})$ of positive actions, the other GP on the remaining set denoted by $\pi_-^*(\mathcal{X})$. Please note that we know the values $\pi_0^*(\mathcal{X})$ from the GPDP algorithm. Both GP models play the role of local experts in the region of their training sets. We assume that the latent policy is locally smooth. Thus, we use the SE covariance function in $\mathcal{GP}_{\pi_+}$ and $\mathcal{GP}_{\pi_-}$, respectively. It remains to predict the class for new query points. This problem is solved by a binary classifier. We greedily choose the GP with higher posterior class probability to predict the optimal action for any state. The classifier plays a similar role as the gating network in a mixture-of-experts setting [8]. Finally, we obtain a combined GP model $\mathcal{GP}_\pi$ of an optimal policy in a continuous-valued part of the state space that models discontinuities along the boundaries of a manifold.

Example: Policy learning
The model of an optimal policy for the underpowered pendulum swing up is given in Figure 1. The black crosses and white circles mark the training sets $\pi_+(\mathcal{X})$ and $\pi_-(\mathcal{X})$, respectively. Discontinuities at the boundaries of the diagonal band (upper left to lower right corner) represent states where maximum applicable torque is just not strong enough to bring the pendulum to the inverted position. The decision of the controller is to use torque in opposite direction, to exploit the dynamics of the pendulum, and to bring it to the goal state from the other side.

Especially in real systems where actions have to be smooth to protect actuators, the suggested method seems reasonable and applicable. Moreover, it seems more general compared to directly learning the policy from training data with a

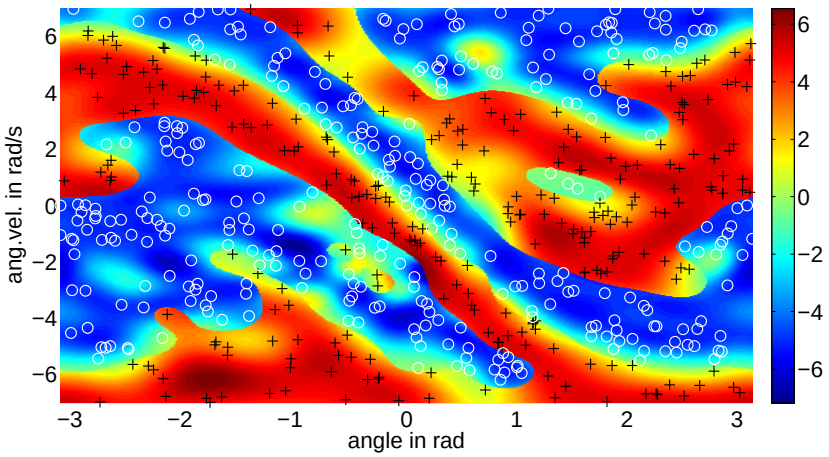

Fig. 1: Optimal policy modeled by switching between two GPs, any of which is trained on different subsets (white circles and black crosses) of optimal actions $\pi_0^*(\mathcal{X})$ returned by GPDP.

single GP and a problem-specific covariance function. Correct problem-specific covariance functions may perform better. However, a lot of expert knowledge is needed in case of possibly nonsmooth predicted policies.

GPDP scales in $\mathcal{O}(|\mathcal{X}||\mathcal{U}|^3 + |\mathcal{X}|^3)$ per time step: A GP model for $Q(\mathbf{x}, \cdot)$ for all $\mathbf{x} \in \mathcal{X}$ and one GP model for the V-function are used. Standard DP scales in $\mathcal{O}(|\mathcal{X}_{\mathrm{DP}}|^2|\mathcal{U}_{\mathrm{DP}}|)$ with substantially bigger sets $\mathcal{U}_{\mathrm{DP}}, \mathcal{X}_{\mathrm{DP}}$. A strength of GPDP is that in case of stochastic dynamics the corresponding RL problem can be solved with no additional computational and memory requirements. In general, the benchmark controller is no longer applicable because of enormous memory demand if a full transition matrix has to be stored. Another point to be mentioned is that the sets $\mathcal{X}$, $\mathcal{U}$ in GPDP can be time-variant. They just serve as training points for the GP models generalizing the value functions to continuous-valued domains.

Example: Performance

We simulate the time-discretized pendulum for $5\,\mathrm{s}$. We consider a benchmark controller based on exact DP in discretized state and action spaces with 7.5×10^7 states in joint state-action space as almost optimal. Figure 2 shows the results of applying both controllers. The dashed blue lines represent the optimal solution of the DP controller with fully known dynamics. The solid green lines are the solution of the GPDP controller based on learned dynamics. The upper panels describe trajectories of angle φ and angular velocity $\dot{\varphi}$, respectively. Applied control actions are given in the lower panel where the error bars in the GPDP solution describe the confidence when applying the corresponding control action (twice standard deviation). The state trajectories (upper panels) almost coincide, and the chosen control actions differ slightly. In the considered example, GPDP based on learned dynamics causes 1.66% more cumulative loss over $5\,\mathrm{s}$.

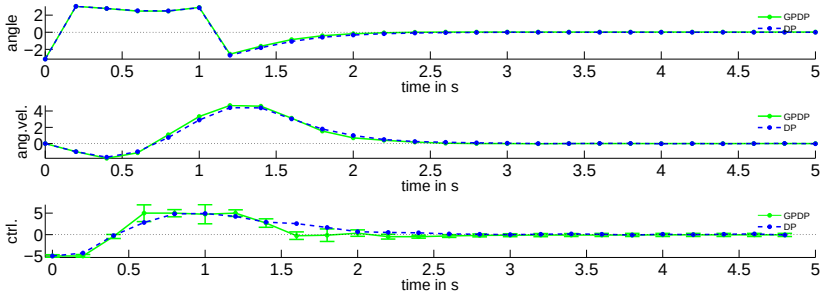

Fig. 2: Trajectories of angle, angular velocity and applied actions for discretized DP and continuous GPDP. The trajectories almost coincide although GPDP uses learned dynamics.

3 Summary

In this paper, we extended a Gaussian process dynamic programming algorithm to the case of unknown deterministic dynamics. We assumed that the system can be modeled by means of a Gaussian process. For this setting, we obtained a closed-loop policy for continuous-valued state and action domains. We showed that in the case of the underpowered pendulum swing up, the policy based on learned system model performs almost as well as a computationally expensive optimal controller.

References

[1] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, USA, 1996.

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

[3] Carl E. Rasmussen and Malte Kuss. Gaussian Processes in Reinforcement Learning. In *Advances in Neural Information Processing Systems 16*, pp. 751–759, June 2004.

[4] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement Learning with Gaussian Processes. In *22nd International Conference on Machine Learning*, pp. 201–208, August 2005.

[5] Marc P. Deisenroth, Jan Peters, and Carl E. Rasmussen. Approximate Dynamic Programming with Gaussian Processes. In *27th American Control Conference*, June 2008.

[6] Christopher G. Atkeson and Stefan Schaal. Robot Learning from Demonstration. In *14th International Conference on Machine Learning*, pp. 12–20, July 1997.

[7] Carl E. Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems 15*, pp. 489–496, 2003.

[8] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87, 1991.