



Dynamic Programming with Gaussian Process Models

Marc P. Deisenroth¹, Carl E. Rasmussen^{1,2}, Jan Peters²

¹ Department of Engineering, University of Cambridge, UK
² Max Planck Institute for Biological Cybernetics, Tübingen, Germany



Abstract

Reinforcement learning with continuous state and action spaces is a challenging problem. In case of nonlinear stochastic dynamics no general closed-form solutions exist to determine an optimal policy. Here, we attempt to learn a model of **unknown stochastic dynamics** of the environment. Using this model we apply an approximate dynamic programming approach based on Gaussian processes, which we call **GPDP** [1]. We explicitly take sources of uncertainty into account. The optimal state-feedbacks returned by this algorithm are generalized to a **discontinuous policy** on the entire state space by means of a mixture-of-experts-like algorithm where we switch between two Gaussian process models trained on subsets of state-feedbacks.

1 Considered Problem

- reinforcement learning problem
- unknown nonlinear stochastic system dynamics
- treat problems with continuous states and actions
- determine closed-loop policy for underpowered systems
- model-based approach

2 Solution Approach

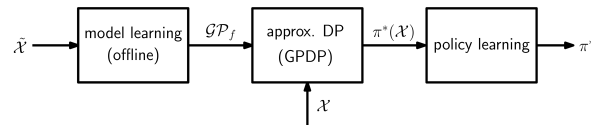


FIGURE 1: Three steps toward a closed-loop policy.

2.1 Model Building

consider stochastic systems of the form

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k$$

with

k : discrete time index

\mathbf{x} : system state

\mathbf{u} : control action

f : unknown nonlinear system function

$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ additive process noise with unknown diagonal covariance matrix Σ

- Gaussian process model for system dynamics based on inputs and deterministic observations of an intrinsically stochastic system: $\mathbf{x}_{k+1} \sim \mathcal{GP}_f(m_f, k_f)$

- get model of dynamics and diagonal covariance matrix Σ

2.2 GPDP

Algorithm 1 GPDP for unknown stochastic systems (batch version)

```

function GPDP(GP_f, X, U)
  V_N(X) = g_term(X)                                ▷ terminal loss
  V_N ~ GP_v(m_v, k_v)                               ▷ GP model for V_N
  for k = N - 1 to 0 do
    for all x_k in X do
      for all u_k in U do
        Q(x_k, u_k) = g(x_k, u_k) + gamma E[V_{k+1}(x_{k+1}) | x_k, u_k, GP_f]
      end for
      Q(x_k, ·) ~ GP_q(m_q, k_q)                       ▷ GP model for Q
      pi*_k(x_k) in arg min_u Q(x_k, u)                ▷ minimization
      V_k(x_k) = Q(x_k, pi*_k(x_k))
    end for
    V_k ~ GP_v(m_v, k_v)                             ▷ GP model for V_k
  end for
  return pi*_0(X)
end function
  
```

- important step

$$\begin{aligned}
 E[V_{k+1}(\mathbf{x}_{k+1}) | \mathbf{x}_k, \mathbf{u}_k] &= E_{V_{k+1}} [E[V_{k+1}(\mathbf{x}_{k+1}) | \mathbf{x}_k, \mathbf{u}_k]] \\
 &= \iint V_{k+1}(\mathbf{x}_{k+1}) p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) p(V_{k+1}) dV_{k+1} d\mathbf{x}_{k+1} \\
 &= \int m_v(\mathbf{x}_{k+1}) p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) d\mathbf{x}_{k+1}, \quad (1)
 \end{aligned}$$

where $\mathbf{x}_{k+1} \sim \mathcal{GP}_f(m_f, k_f)$

- explicitly consider uncertainty about
 - model of system dynamics
 - stochasticity of system
 - model of value function V
- Gaussian approximation of distribution of uncertain integral value (1) analytically computable with Bayesian Monte Carlo [2]

2.3 Learning a Discontinuous Policy

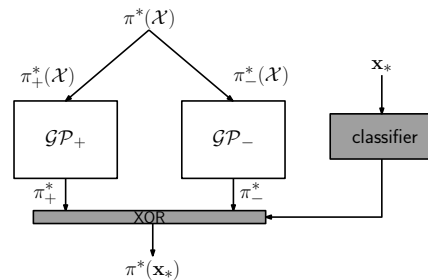
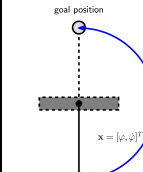


FIGURE 2: Learning a discontinuous policy by means of switching locally smooth GP models.

- train two GP models on sub-datasets (positive and negative actions)
- known labels (+/-) for $\pi*_0(X)$ from GPDP
- classifier to select the right GP for a test point \mathbf{x}_*
- similar to the mixture-of-experts setting [3]

3 Underpowered Pendulum Swing Up



- continuous-time system with ODE

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

- control change every $\Delta_t = 0.2s$ (zero-order hold)
- (deterministic) measurement of state every 0.2s by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k) + \mathbf{w}_k = \int_{t=k\Delta_t}^{(k+1)\Delta_t} F(\mathbf{x}(t), u_k, v(t)) dt$$

with $F = [\dot{\varphi}, \varphi]^T$, $\mathbf{x} = [\varphi, \dot{\varphi}]^T$, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma)$

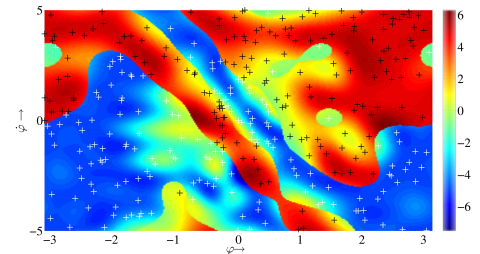


FIGURE 3: Mean of learned discontinuous policy with training sets for local models.

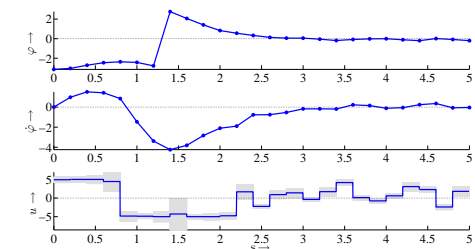


FIGURE 4: Sample state and control trajectories over time. 2σ confidence interval of chosen control (zero-order hold) given in lower panel.

References

- [1] Marc P. Deisenroth, Jan Peters, and Carl E. Rasmussen. Approximate Dynamic Programming with Gaussian Processes. In *ACC 2008*, Seattle, WA, USA, June 2008.
- [2] Carl E. Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. In *NIPS 15*, pages 489–496. The MIT Press, Cambridge, MA, U.S.A., 2003.
- [3] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87, 1991.