

Model-Based Reinforcement Learning with Continuous States and Actions

Marc Deisenroth



European Symposium on Artificial Neural Networks
Bruges, Belgium

April 23, 2008

joint work with Carl Rasmussen^{1,2} and Jan Peters²

¹ Department of Engineering, University of Cambridge, Cambridge, UK

² Max Planck Institute for Biological Cybernetics, Tübingen, Germany

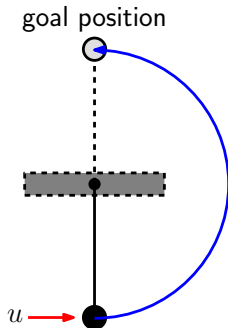
Problem

use reinforcement learning to solve underpowered pendulum swing-up task

- 1 start from downward position
- 2 bring the pendulum to the upright position
- 3 balance it there

challenges:

- **underpowered** system, i.e., maximum torque is insufficient to bring the pole to the upright position
- unknown dynamics
- **continuous** states and actions (no spatial discretization)
- interested in **policy**, not a single trajectory



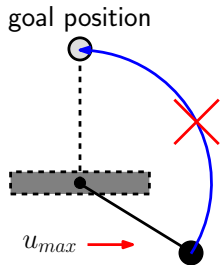
Problem

use reinforcement learning to solve underpowered pendulum swing-up task

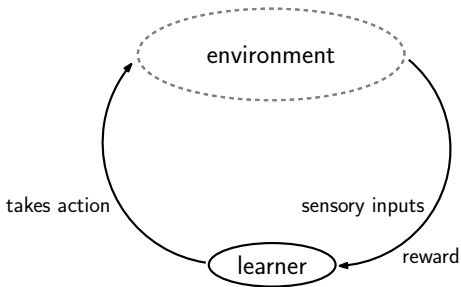
- 1 start from downward position
- 2 bring the pendulum to the upright position
- 3 balance it there

challenges:

- **underpowered** system, i.e., maximum torque is insufficient to bring the pole to the upright position
- unknown dynamics
- **continuous** states and actions (no spatial discretization)
- interested in **policy**, not a single trajectory



Reinforcement Learning



→ maximize expected rewards over a long term

connections to

- optimal control
- (Bayesian) decision theory
- game theory
- animal learning



borrowed from *Science* magazine

Dynamic Programming

efficient method to solve RL problem in discrete time:
dynamic programming (DP)

problems:

- only tractable for finite number of states and actions
 - curse of dimensionality
- approximations to solve the problem for **continuous-valued** states and actions

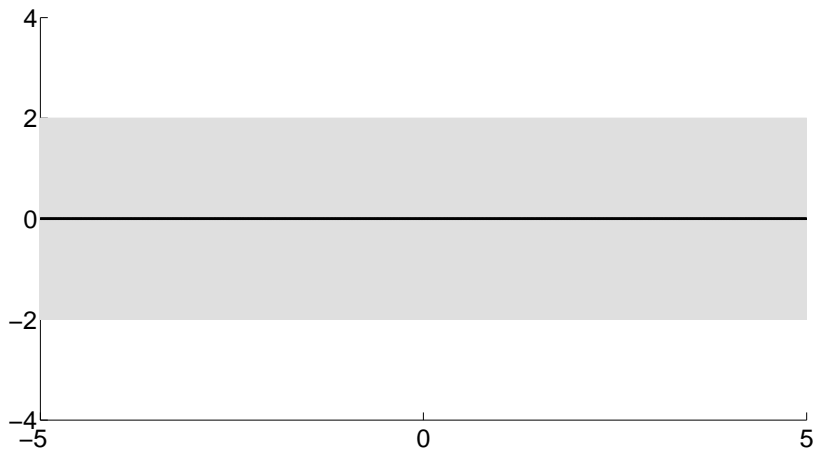
here: use **Gaussian process** framework

Bayesian Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on (noisy) observations

Bayesian Regression with Gaussian Processes

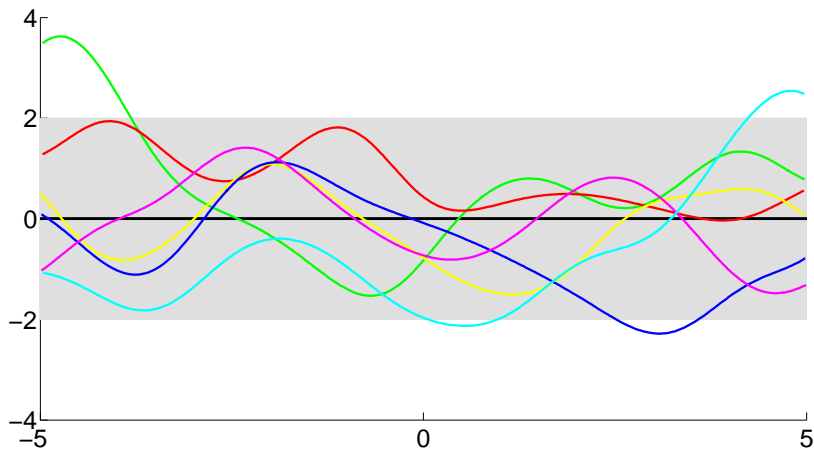
example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on (noisy) observations



shaded area: uncertainty about underlying function

Bayesian Regression with Gaussian Processes

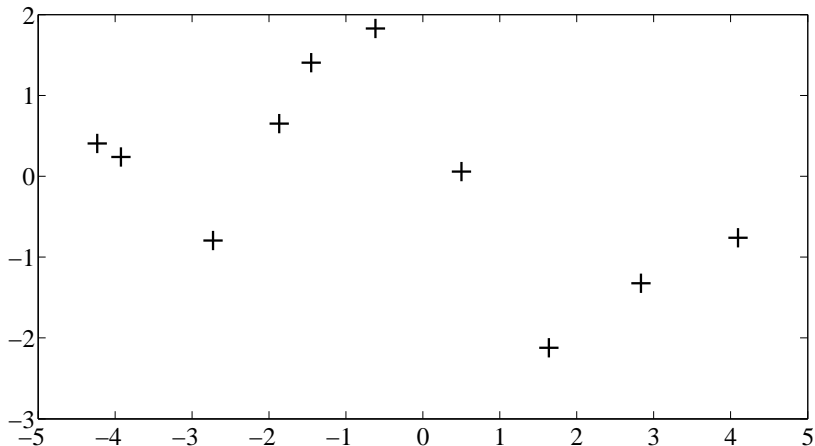
example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on (noisy) observations



shaded area: uncertainty about underlying function

Bayesian Regression with Gaussian Processes

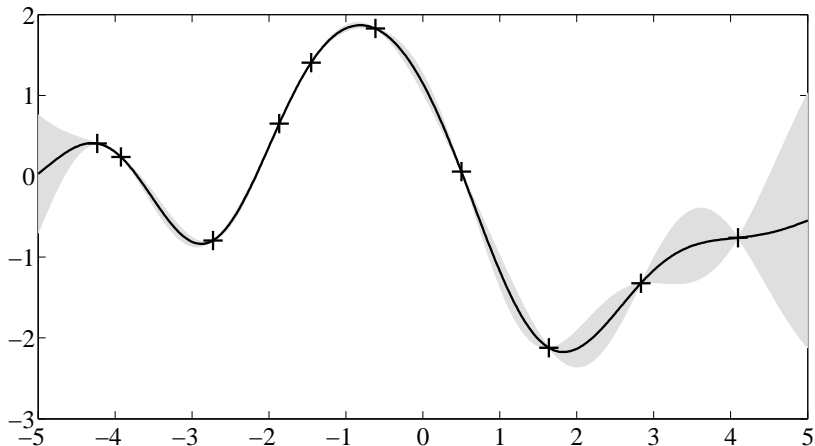
example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on (noisy) observations



shaded area: uncertainty about underlying function

Bayesian Regression with Gaussian Processes

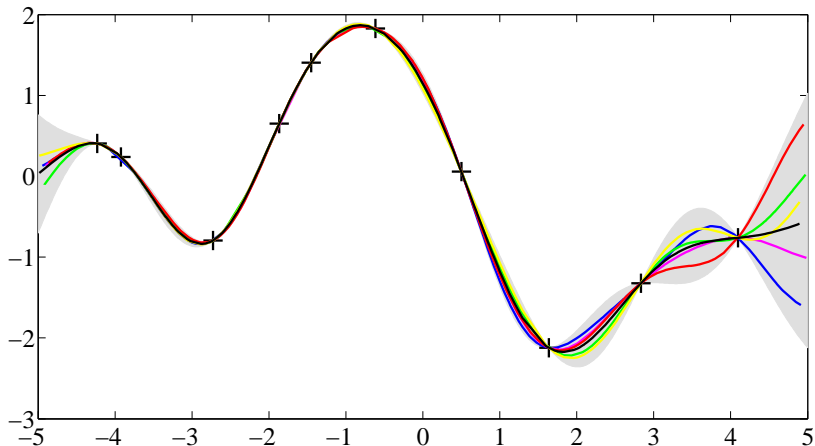
example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on (noisy) observations



shaded area: uncertainty about underlying function

Bayesian Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on (noisy) observations



shaded area: uncertainty about underlying function

Objective and Novel Approach

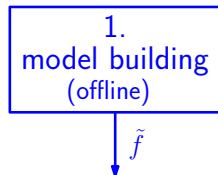
find best actions to control pendulum optimally

- 1 initially **unknown** (deterministic) transition dynamics
- 2 **continuous** state and action domains
- 3 model of optimal policy **everywhere**

Objective and Novel Approach

find best actions to control pendulum optimally

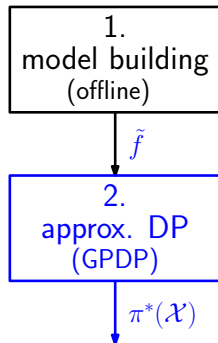
- 1 initially **unknown** (deterministic) transition dynamics
- 2 **continuous** state and action domains
- 3 model of optimal policy **everywhere**



Objective and Novel Approach

find best actions to control pendulum optimally

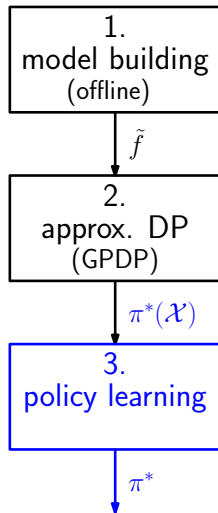
- 1 initially **unknown** (deterministic) transition dynamics
- 2 **continuous** state and action domains
- 3 model of optimal policy **everywhere**



Objective and Novel Approach

find best actions to control pendulum optimally

- 1 initially **unknown** (deterministic) transition dynamics
- 2 **continuous** state and action domains
- 3 model of optimal policy **everywhere**



1. Dynamics Learning with Gaussian Processes

model discrete-time system dynamics $\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k)$

- sample trajectories starting from initial state and goal state
- fit Gaussian process to model dynamics

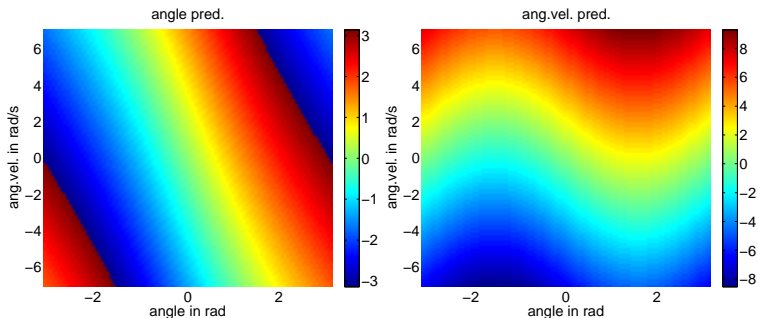


Figure: Dynamics model for fixed inputs

→ use learned model of the system dynamics in [Gaussian Process Dynamic Programming \(GPDP\)](#)

1. Dynamics Learning with Gaussian Processes

model discrete-time system dynamics $\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k)$

- sample trajectories starting from initial state and goal state
- fit Gaussian process to model dynamics

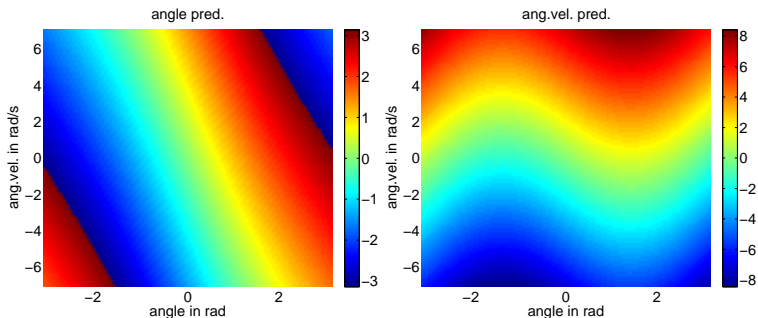


Figure: Dynamics model for fixed inputs

→ use learned model of the system dynamics in [Gaussian Process Dynamic Programming \(GPDP\)](#)

1. Dynamics Learning with Gaussian Processes

model discrete-time system dynamics $\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k)$

- sample trajectories starting from initial state and goal state
- fit Gaussian process to model dynamics

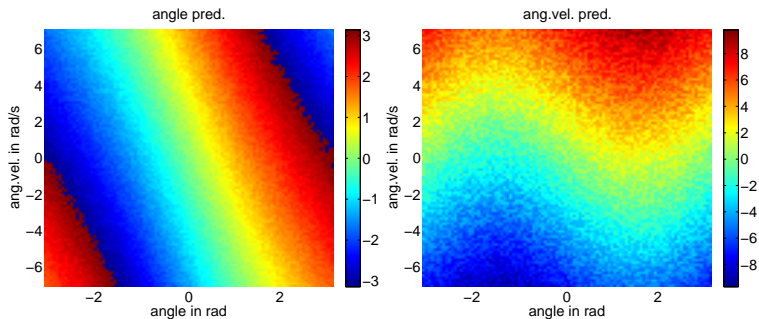


Figure: Dynamics model for fixed inputs

→ use learned model of the system dynamics in [Gaussian Process Dynamic Programming \(GPDP\)](#)

2. Gaussian Process Dynamic Programming

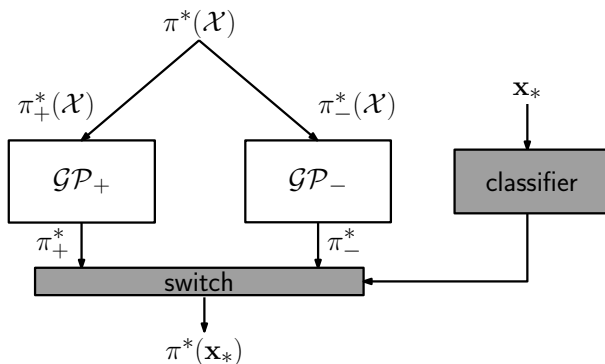
key idea of GPDP (Deisenroth et al., ACC 2008):

- model value functions in DP recursion with GPs
 - ➔ generalization of DP to continuous states and actions
- return optimal actions $\pi^*(\mathcal{X})$ for finite set of states \mathcal{X}

here: only GP model of dynamics available

- GPDP can be adapted to derive $\pi^*(\mathcal{X})$ based on **learned** model dynamics
- explicitly consider **uncertainty** in model by averaging over it

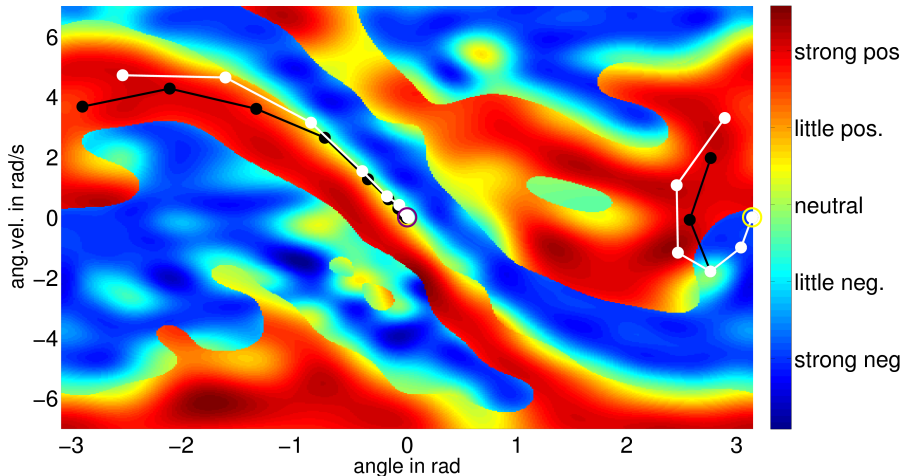
3. Policy Learning



- extend finite set of optimal actions to entire state space \rightarrow get policy π^*
- switch between locally trained policy GPs to model discontinuous policy
- similar to mixture-of-experts setting (Jacobs et al., Neur. Comp. 1991)

Learned Policy with Trajectories

learned policy with state trajectories



black: near-optimal DP controller using **correct** dynamics
white: GPDP controller based on **learned** dynamics

- ▶ 3 steps to model-based RL
 - ① learn deterministic system dynamics (pre-processing step)
 - ② use (probabilistic) dynamics model in Gaussian Process Dynamic Programming
 - ③ policy learning by switching between local models
- ▶ solution to RL problem in continuous-valued state and action domains based on a learned model

<http://mlg.eng.cam.ac.uk/marc/>