

# Policy Optimization by Implicit Probabilistic Simulation

Marc Deisenroth and Carl Edward Rasmussen



European Workshop on Reinforcement Learning  
Lille, France

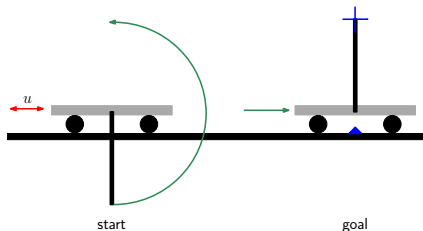
July 03, 2008

# Motivation

- fast (data-efficient) RL algorithm with continuous states and actions
- here: model-based approach
- optimize policy by fictitious simulation based on model instead of real interactions
- need for probabilistic models to express what is known/unknown

# Efficient RL solution

illustrative example



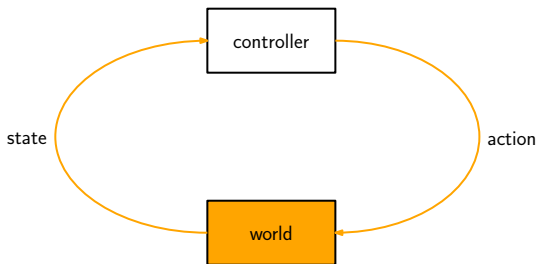
- measure: 4 state variables
- input: deterministic action (horizontal force applied to cart)
- low sampling frequency (5 Hz)
- problem unsolvable for linear controller
- easy standard (nonlinear) control problem, but we **learn from scratch**:  
interleave dynamics learning and policy optimization

→ demo

# Key Idea

loop over

- 1 make observations from **real world** while applying current optimal policy (init: random actions)



# Key Idea

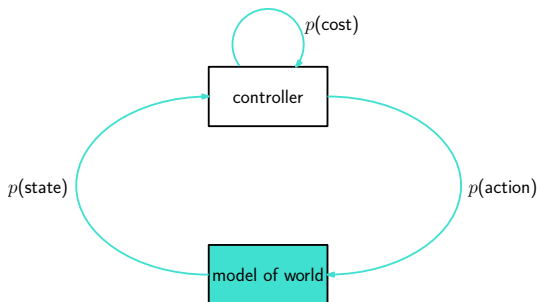
loop over

- 1 make observations from **real world** while applying current optimal policy (init: random actions)
- 2 generate/update (probabilistic) model of the world

# Key Idea

loop over

- 1 make observations from **real world** while applying current optimal policy (init: random actions)
- 2 generate/update (probabilistic) model of the world
- 3 optimize policy based on **fictitious simulation** using new probabilistic model of the world



# Modeling the Dynamics

- state of cart-pole system is given by position, velocity, angle, angular velocity:  $\mathbf{s} = [x, \dot{x}, \varphi, \dot{\varphi}]^T$
- model a discrete-time system with continuous-valued states and actions:  $\mathbf{s}_{t+1} = f(\mathbf{s}_t, u_t)$ , where  $f$  is unknown
- describe dynamics by 4 **Gaussian process** (GP) models (one for each state component):

$$s_{t+\Delta_t}^i \sim \mathcal{GP}(\mathbf{s}_t, u_t), \quad i = 1, \dots, 4$$

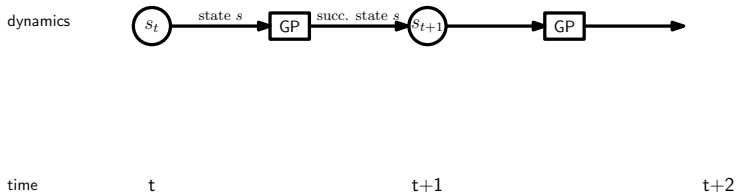
# Modeling the Dynamics

- state of cart-pole system is given by position, velocity, angle, angular velocity:  $\mathbf{s} = [x, \dot{x}, \varphi, \dot{\varphi}]^T$
- model a discrete-time system with continuous-valued states and actions:  $\mathbf{s}_{t+1} = f(\mathbf{s}_t, u_t)$ , where  $f$  is unknown
- describe dynamics by 4 **Gaussian process** (GP) models (one for each state component):

$$s_{t+\Delta_t}^i \sim \mathcal{GP}(\mathbf{s}_t, u_t), \quad i = 1, \dots, 4$$

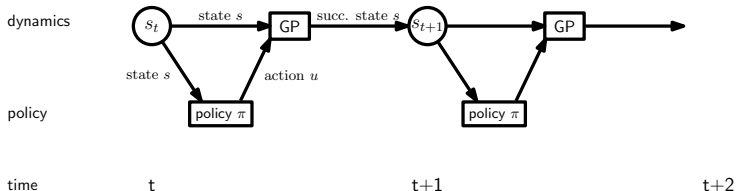
- GPs predict **distribution** of successor state
  - describes what we know and what we don't know
  - we can explicitly account for uncertainties in learned model
- easy to learn for small  $\Delta_t$ , very difficult for large  $\Delta_t$

# Long-Term Predictions



- cascade short-term predictions
- keep track of uncertainty evolution → explicitly **propagate uncertainty**

# Interaction between Dynamics Model and Policy



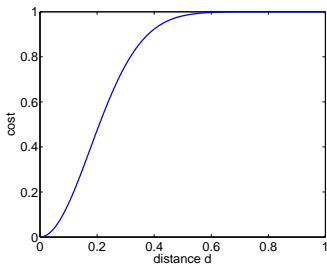
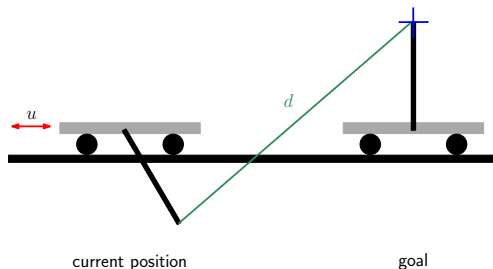
- policy implements deterministic (nonlinear) function
- policy evaluation: determine expected finite-horizon cost

$$V^\pi(\mathbf{s}_0) = \sum_{t=0}^T \mathbb{E}[\ell(\mathbf{s}_t) | \pi, p(\mathbf{s}_0)] \quad \text{here: } T = 25$$

by fictitious simulation:

state is uncertain (probabilistic dynamics model)  $\rightarrow$  distribution over actions

# Cost Function



$$\ell(s) = 1 - \exp(-c d^2) \in [0, 1]$$

- general geometric cost
  - depends on position variables  $(x, \varphi)$  only (no velocity information)
  - learning algorithm has to detect that velocities are crucial
- algorithm-specific, but not hand-crafted problem-specific cost:
  - very uncertain about the state  $\rightarrow$  full expected cost

# Gradient-Based Policy Optimization

minimize expected finite-horizon cost

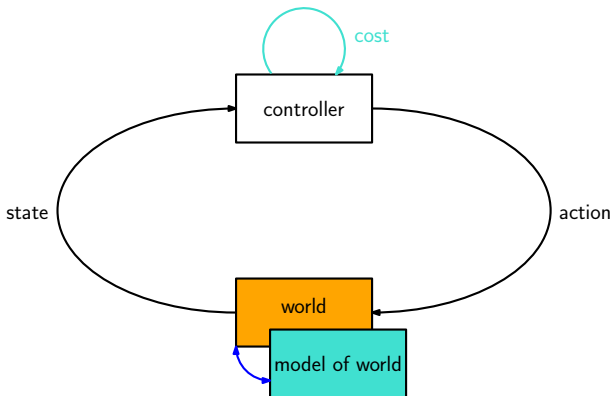
$$V^\pi(\mathbf{s}_0) = \sum_{t=0}^T \mathbb{E}[\ell(\mathbf{s}_t) | \pi, p(\mathbf{s}_0)] \quad \text{here: } T = 25$$

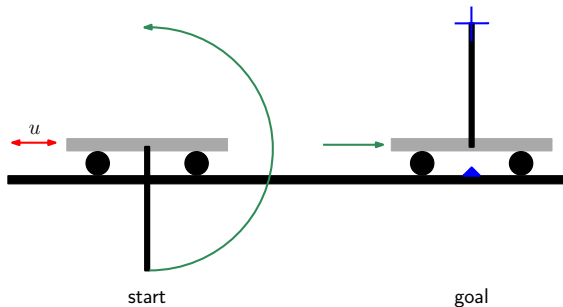
- function of the policy parameters  $\theta$   
→ gradient of  $V^\pi$  wrt  $\theta$  (analytic solution)
- conjugate gradients to optimize policy

# Reminder: Key Idea of Algorithm

loop over

- 1 make observations from system while applying optimal policy (init: random actions)
- 2 generate/update probabilistic dynamics model
- 3 gradient-based policy optimization: fictitious simulation using new probabilistic dynamics model





total of 25–35 seconds of interaction with real system to solve task near optimally

# Wrap-up

assumptions employed

▶ time discretization  $\Delta_t$ : 200 ms

▶ optimization horizon  $T$ : 5 s

▶ constant  $c$  in the cost function  $\ell$

➔ need some information about dynamics, but still fairly general assumptions

# Wrap-up

assumptions employed

- ▶ time discretization  $\Delta_t$ : 200 ms
- ▶ optimization horizon  $T$ : 5 s
- ▶ constant  $c$  in the cost function  $\ell$

→ need some information about dynamics, but still fairly general assumptions

results

- ▶ learn nonlinear dynamics and nonlinear controller
- ▶ data efficient RL algorithm
- ▶ no sampling/oracle required

<http://mlg.eng.cam.ac.uk/car1/ewrl08/>

not yet addressed

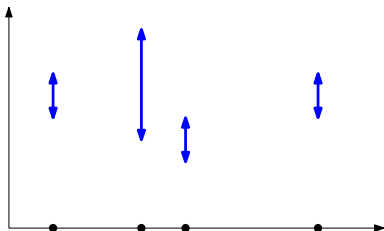
- ▶ active learning
- ▶ exploration-exploitation tradeoff
- ▶ “pruning” observation data set **without** losing information

# Optimizing a GP Controller

# Optimizing a GP Controller

→ **learn** controller by finding parameters that minimize expected finite-horizon cost (gradient-based)

- **hyperparameters** of the covariance function
- **pseudo training set**: find best function values for given inputs



# Optimizing a GP Controller

→ learn controller by finding parameters that minimize expected finite-horizon cost (gradient-based)

- hyperparameters of the covariance function
- pseudo training set: find best function values for given inputs

how can we know where the (pseudo) inputs of the GP controller are?

→ optimize these ones as well!

→ training the GP controller relies on internal simulation (no direct interaction with system required)

