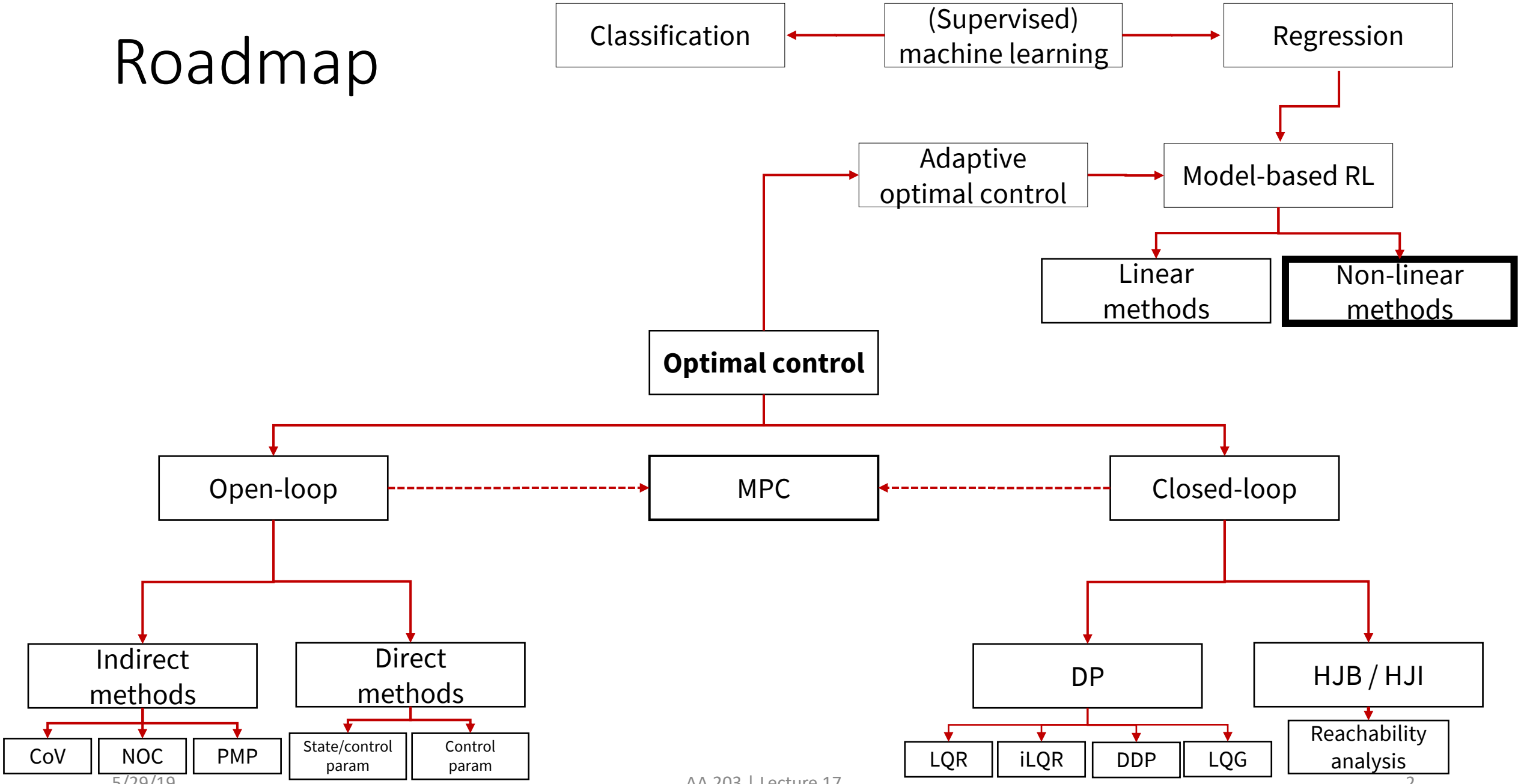
AA203
Optimal and Learning-based Control

Model-based Reinforcement Learning (non-linear methods)

Stanford University

ASL
Autonomous Systems Lab

Roadmap


Classification
(Supervised) machine learning
Regression
Adaptive optimal control
Model-based RL
Linear methods
Non-linear methods
Optimal control
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI
CoV
NOC
PMP
State/control param
Control param
LQR
iLQR
DDP
LQG
Reachability analysis

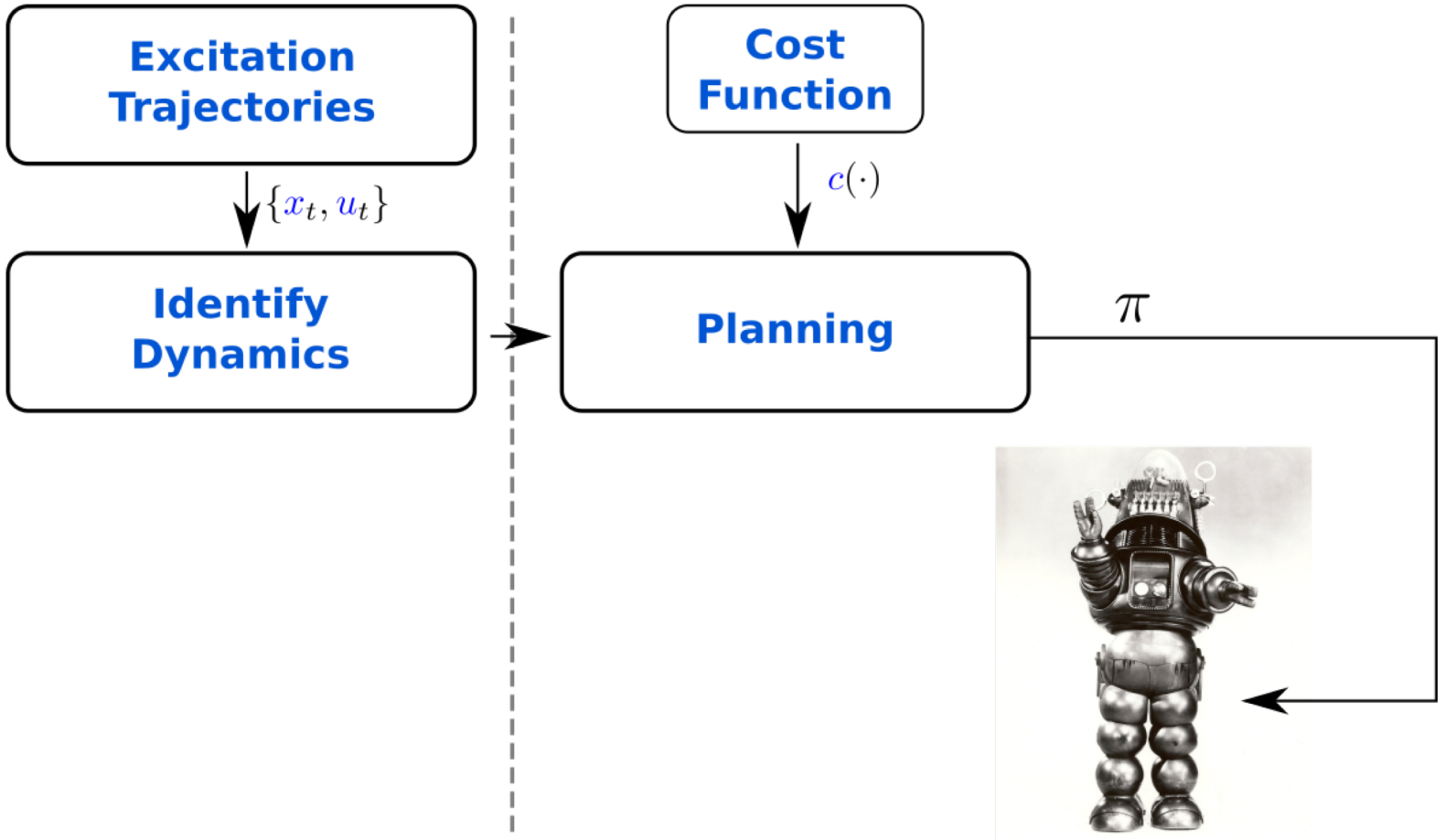
Today Lecture

- In previous lectures you studied MBRL with linear systems, and non-linear modeling (GPs and NNs).
- In this lecture, we will look at the challenges of MBRL when using non-linear dynamics

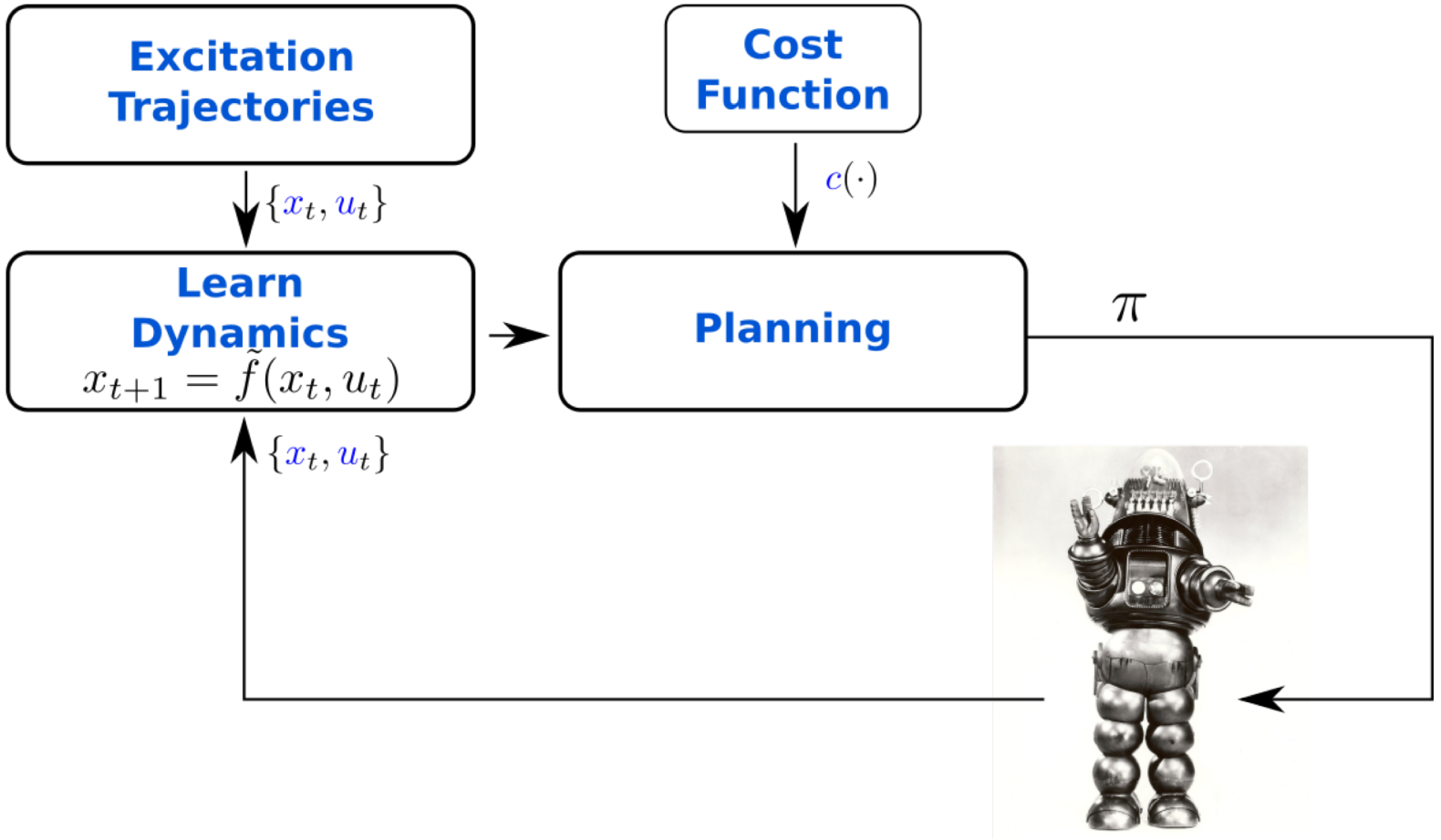
Overview

- Brief Refresher
- Introduction to MBRL with non-linear models
- Case study 1: Probabilistic Inference for Learning Control (PILCO)
- Case study 2: Probabilistic Ensembles with Trajectory Sampling (PETS)
- Open Research and Challenges

System Identification



Model-based Reinforcement Learning (MBRL)



Model-based RL for non-linear systems

- In the real world, dynamics is never perfectly linear
- In previous lectures you studied MBRL with linear systems
- What are the challenges when using non-linear systems?
  - No known closed-form solutions
    (Solutions usually rely on numerical approximation)
  - Non-linear models are only locally accurate

Linear vs Non-linear models

- Linear models are not accurate for non-linear dynamics
- “In nonlinear systems, local model knowledge is inaccurate elsewhere”
- In practice, non-linear models should be considered accurate only in the training distribution.
- Moreover, with non-linear models no close-form solutions to planning

PILCO [Deisenroth et al. 2014]

Main idea:

–Train dynamics model $\tilde{f}$ from previous data (using a GP)

–For a given policy $\pi(\boldsymbol{\theta})$, compute the state distribution $p(\boldsymbol{x}_0), \ldots, p(\boldsymbol{x}_T)$ and the corresponding cost $J^\pi = \sum_{t=0}^{T} \mathbb{E}[c(\boldsymbol{x}_t)]$

–Numerically optimize the policy parameters w.r.t. the cost $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J^\pi|_\theta$

– Evaluate the resulting policy $\pi(\boldsymbol{\theta}^*)$ in the real world, and collect new data

Forward Dynamics Model

- Consider dynamical system $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + w\,, \quad w \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}_w)$
- This system can be approximated by a Gaussian Process $\tilde{f}$ trained from previous data $\mathbb{D} = \{(\boldsymbol{x}_t, \boldsymbol{u}_t), \boldsymbol{x}_{t+1}\}$
- Prediction of the GP $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = N(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{y}]\,, \quad \boldsymbol{\Sigma}_{t+1} = k_{**} - \boldsymbol{k}_*^T(\boldsymbol{K} + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{k}_*$$

- Two important practical tricks:

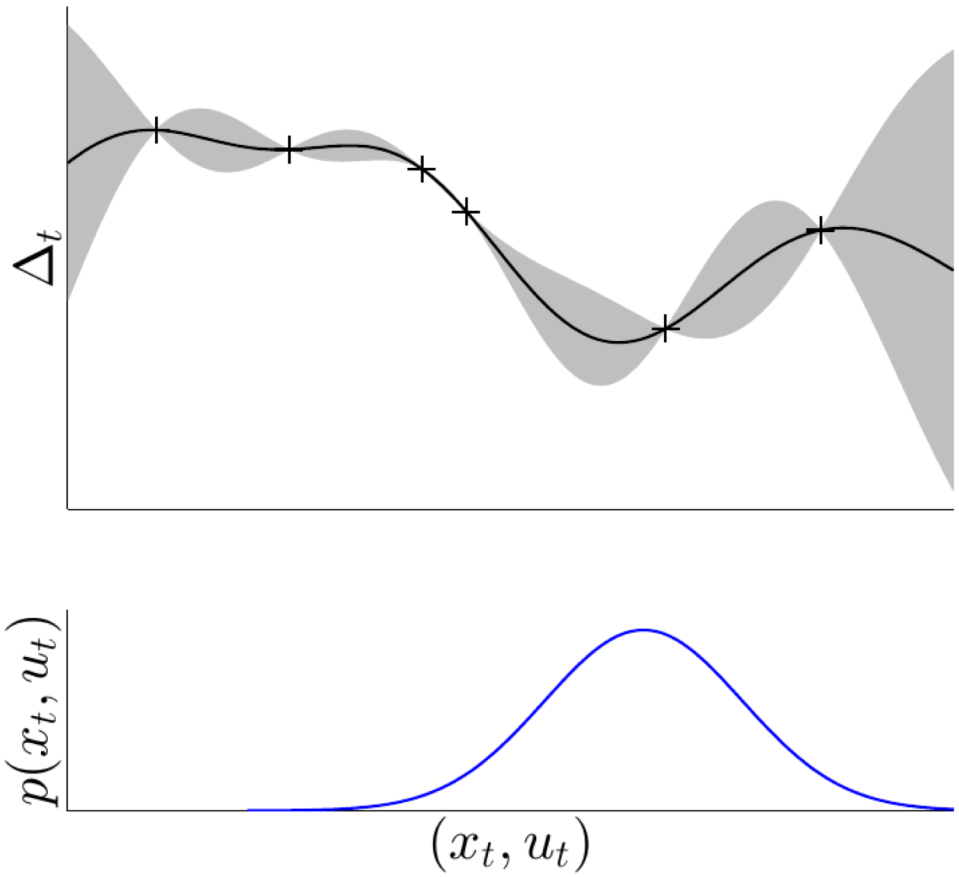
–Replace absolute prediction with delta prediction $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \Delta\boldsymbol{x}_t$
(Equivalent to specify a prior mean function $m(x) = x$)

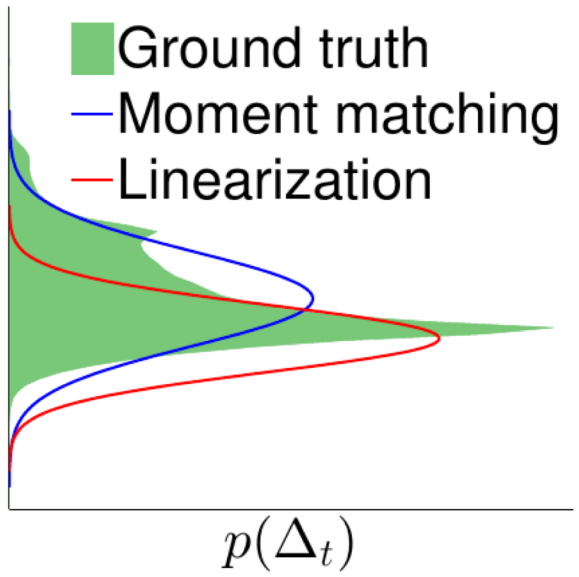
–Augment angles $x^i = [\sin(x^i)\,, \cos(x^i)]$

State Propagation

- Need to compute state distribution $p(\boldsymbol{x}_0), \ldots, p(\boldsymbol{x}_T)$
- We know that $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = N(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) \quad \text{s.t.} \quad \boldsymbol{u}_t = \pi(\boldsymbol{x}_t, \boldsymbol{\theta})$
- We assume an approximate Gaussian joint probability distribution
- $p(\boldsymbol{x}_t, \boldsymbol{u}_t) = p(\boldsymbol{x}_t, \pi(\boldsymbol{x}_t)) = p(\tilde{\boldsymbol{x}}_t)$
- From the resulting form $p(\boldsymbol{\Delta}_t) = \iint p(\tilde{f}(\tilde{\boldsymbol{x}}_t)|\tilde{\boldsymbol{x}}_t) p(\tilde{\boldsymbol{x}}_t) d\tilde{f} d\tilde{\boldsymbol{x}}_t$ we need to integrate out both the random variable $\tilde{\boldsymbol{x}}_t$, and the random function $\tilde{f}$ (which is the posterior GP distribution)

State Propagation

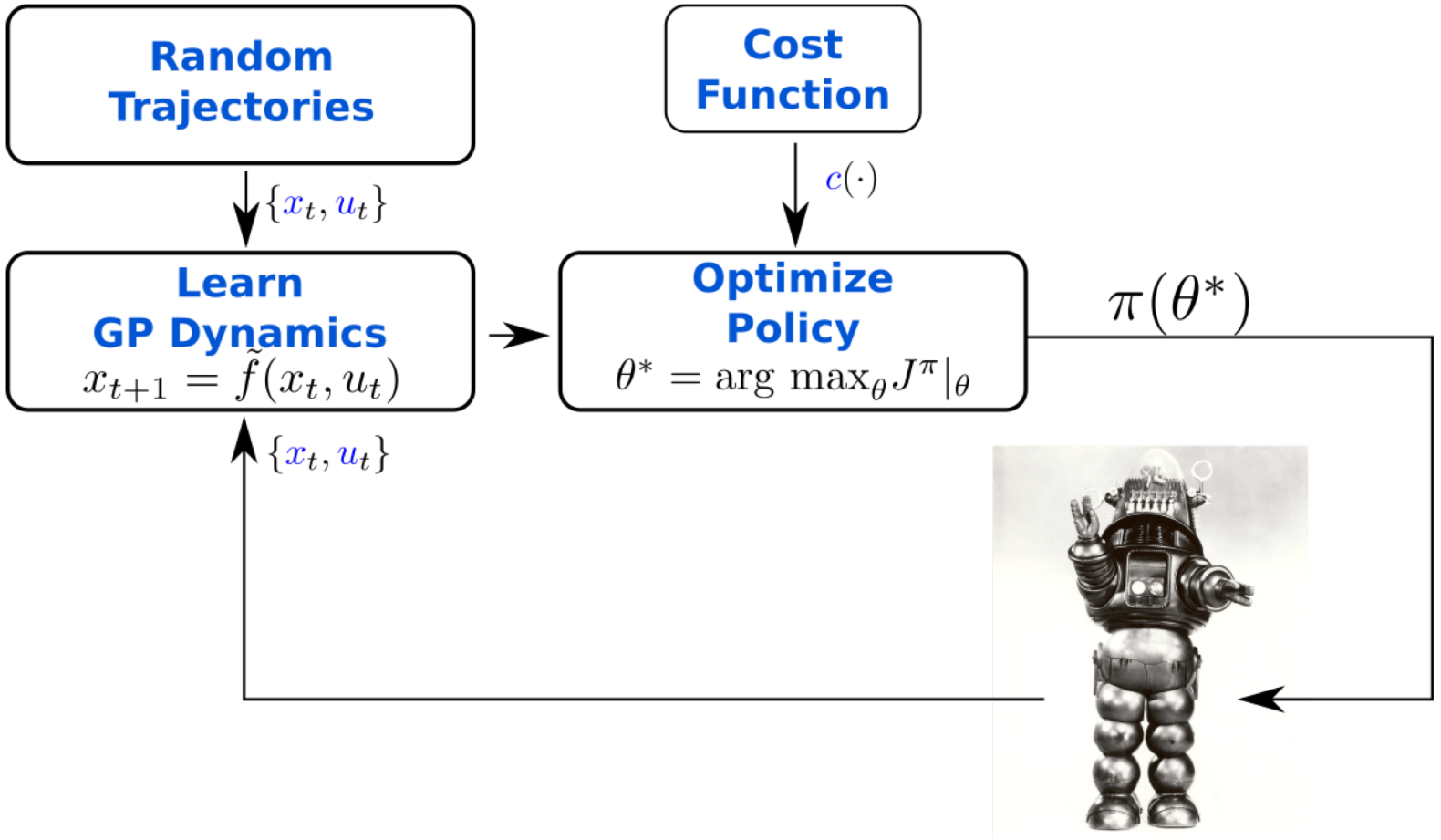




Policy optimization

- Cost-to-go can be computed as $\mathbb{E}[c(\boldsymbol{x}_t)] = \int c(\boldsymbol{x}_t) N(\boldsymbol{x}_t | \boldsymbol{\mu}_t, \boldsymbol{\Sigma_t}) d\boldsymbol{x}_t$
- The policy $\pi(\boldsymbol{\theta})$ can be linear or non-linear (e.g., RBF network)
- The policy optimization $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J^\pi|_\theta$ must be solved numerically
- PILCO allows to compute analytical gradients $\frac{dJ^\pi(\theta)}{d\theta}$
  (*More details in the paper*)
- We can thus use first-order optimization methods (e.g., L-BFGS)

Recap


Random Trajectories
$\{x_t, u_t\}$
Learn GP Dynamics
$x_{t+1} = \tilde{f}(x_t, u_t)$
$\{x_t, u_t\}$
Cost Function
$c(\cdot)$
Optimize Policy
$\theta^* = \arg\max_\theta J^\pi|_\theta$
$\pi(\theta^*)$

Inverted Pendulum

Limitations of PILCO

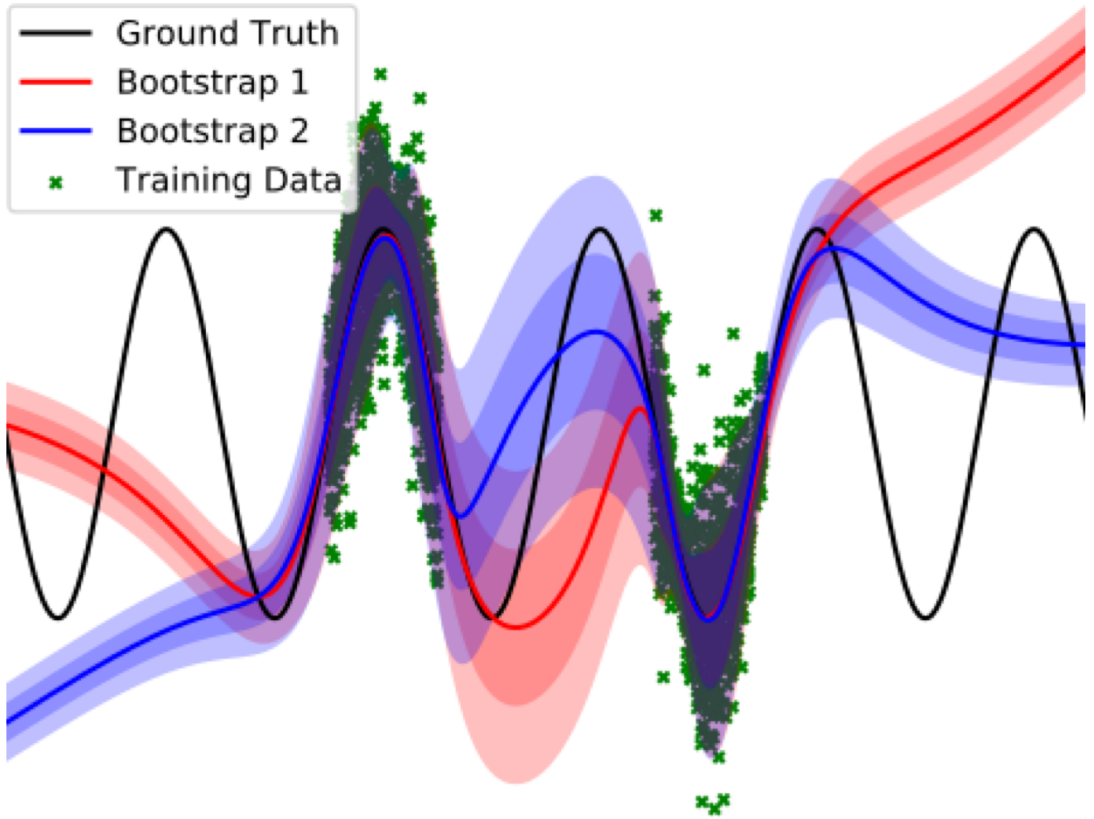
- Gaussian Processes
  - Do not scale well to high-dimensional spaces
  - Can not model discontinuities
  - Require expert knowledge to select an appropriate kernel
- Moment Matching propagation
  - Approximation can be inaccurate
  - Many dynamical systems are multi-modal
- Numerical optimization can be challenging
  - *Computationally slow*
  - *Non-convex optimization problem*

PE-TS [Chua et al. 2018]

- Strongly inspired by PILCO
- Aim to scale to tasks where PILCO is impractical
- 3 main modifications:
  - Dynamics Modeling:
    NN to scale to higher-dimensional spaces
  - State-action propagation:
    Compute state-action distribution using sampling
  - Policy:
    Instead of closed-form policy, uses MPC

Forward Dynamics model

- Neural Networks can scale to hundreds/thousands of inputs
- However, it is not easy to incorporate uncertainty in Neural Networks
- PETS uses a Probabilistic Ensemble of Neural Networks
  - Each NN output a Gaussian distribution *(Aleatoric uncertainty)*
  - Multiple models are trained in an Ensemble *(Epistemic uncertainty)*



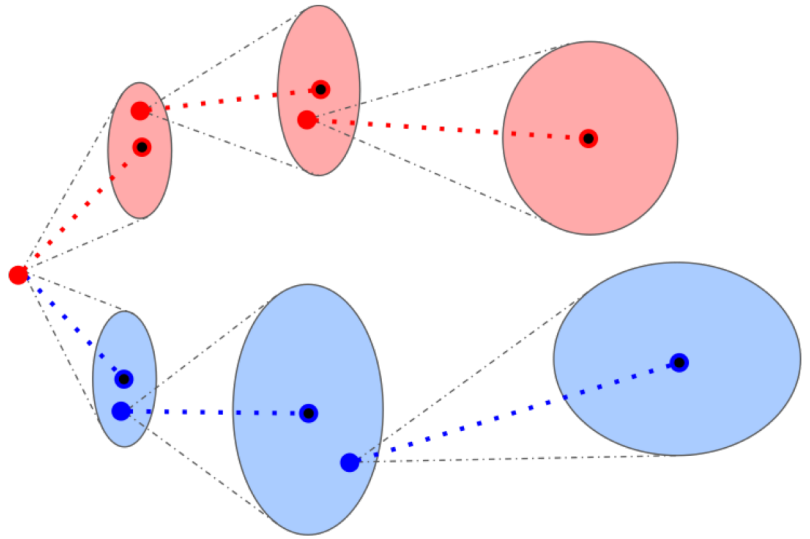
Propagation

- Trajectory sampling approximates

$$p(\boldsymbol{\Delta}_t) = \iint p(\tilde{f}(\tilde{\boldsymbol{x}}_t)|\tilde{\boldsymbol{x}}_t)p(\tilde{\boldsymbol{x}}_t)d\tilde{f}d\tilde{\boldsymbol{x}}_t$$

- from samples (particle)
- Each particle is independently propagated as

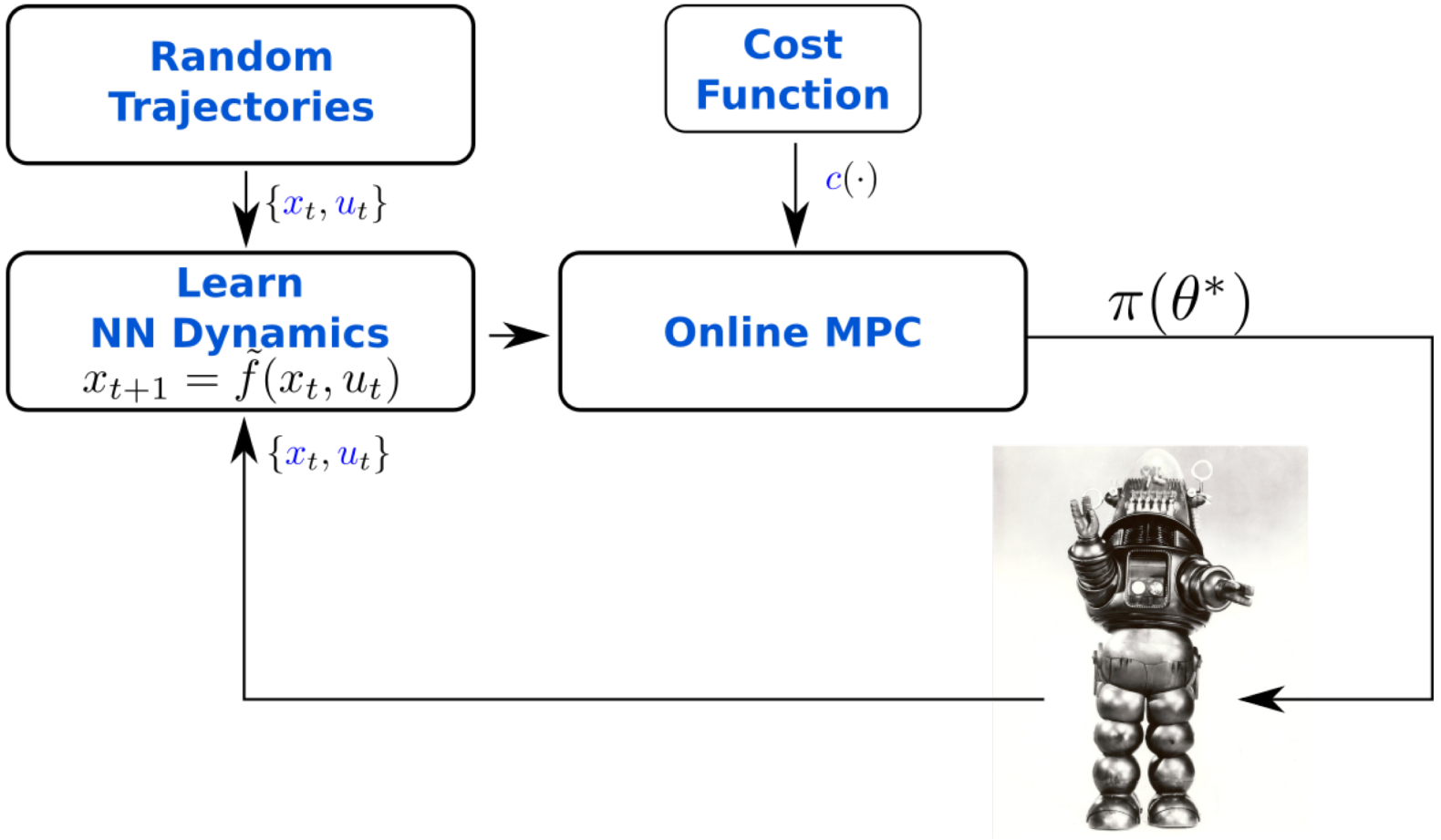
$$\boldsymbol{x}_{t+1} \sim p(\boldsymbol{x}_{t+1}) = \tilde{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$$



Policy

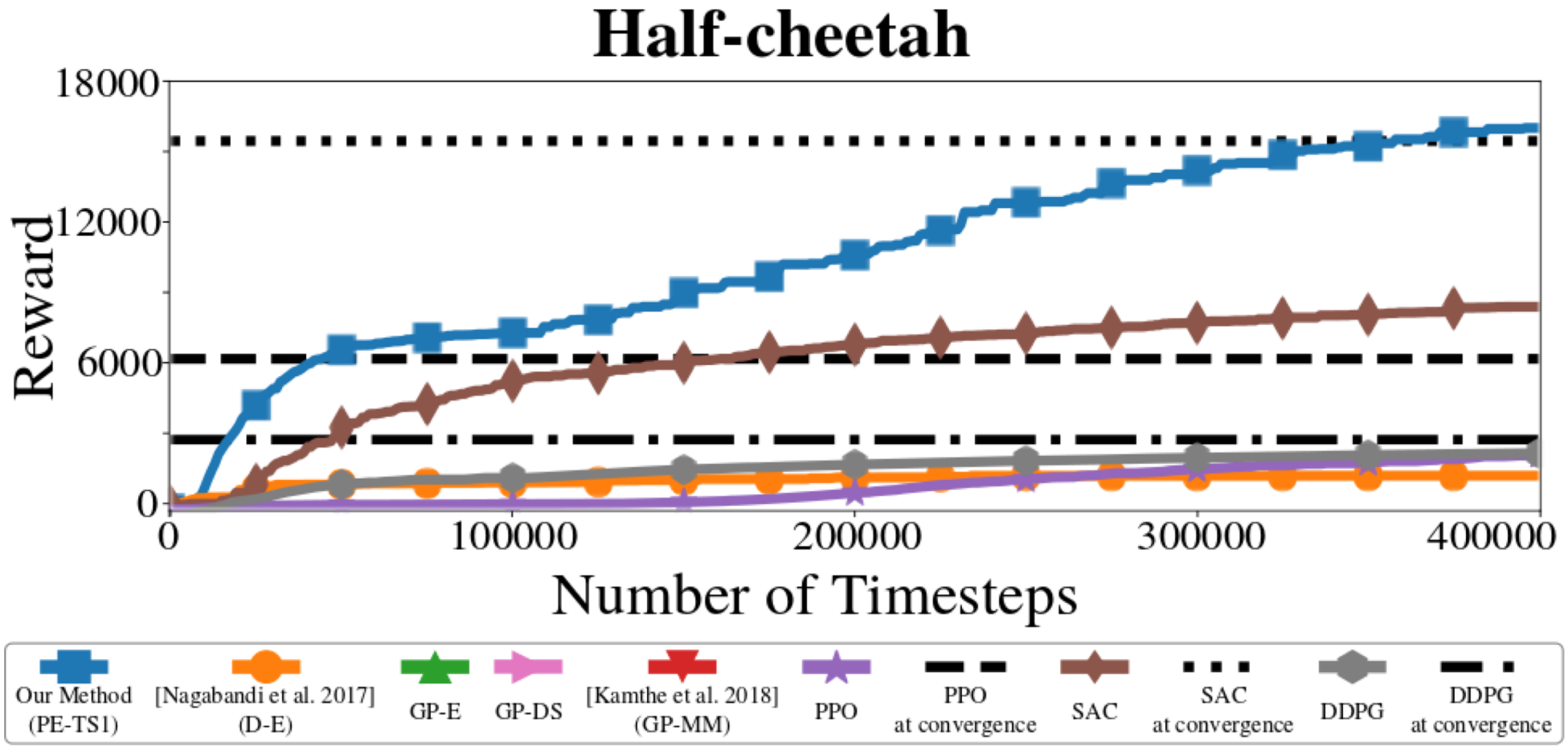
- Instead of computing a closed-form policy, PETS uses MPC
- At each timestep, MPC computes the action
- Advantage: more robust
- Drawback: computationally intensive

Recap


Random Trajectories
Cost Function
$c(\cdot)$
$\{x_t, u_t\}$
Learn NN Dynamics
$x_{t+1} = \tilde{f}(x_t, u_t)$
Online MPC
$\pi(\theta^*)$
$\{x_t, u_t\}$

Recap

Half-cheetah



Half-cheetah

Current Challenges and Open Questions

- MBRL is a hot research topic
- Many unsolved questions:
  - High-dimensional state spaces
  - POMDP
  - Priors for model
  - State-temporal abstraction
  - Exploration
  - Mixing model-based and model-free RL
  - ...

MBRL from images

- Is it possible to learn models of extremely high-dimensional spaces (e.g., images) that can be used for control?
- Recent literature suggest that the answer is YES!
(Very active field of research in the last 3-4 years)
- 3 key insights:
  - Use (and adapt) deep models from computer vision
  - Very important to collect lot of data from large distributions
(Often done via self-supervised learning)
  - Use of latent variable models
-

Model Capacity

- Non-linear models have poor generalization outside of training distribution.
- How can we improve their generalization capabilities?
- Multiple possible paths:
  - Impose more structure/prior in the model
    (e.g., by incorporating analytical knowledge or via Meta-learning)
  - Risk sensitive planning

Exploration

- Behavior of the policy can be too greedy
- No encouragement to explore new area of the space
- How can we explicitly encourage exploration?
- Multiple proposed approaches
  - Information gain
  - Novelty
  - Curiosity
  - ...

Long-term planning

- During state propagation, error compounds
- Are there alternatives to recurrent one-step-ahead predictions?
- Recently a lot of work on multi-step ahead models
  - What exact form should the model take?
  - How can we efficiently plan in this space?

Summary

- Presented formulation and issues of non-linear MBRL.
- Studied two algorithms in detail:
  - Probabilistic Inference for Learning Control (PILCO)
  - Probabilistic Ensembles with Trajectory Sampling (PETS)
- Discussed current research topics and ongoing challenges

Next Time

- Model-free RL

Further Reading

•Deisenroth, M.; Fox, D. & Rasmussen, C. Gaussian Processes for Data-Efficient Learning in Robotics and Control IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2014, 37, 408-423

•Chua, K.; Calandra, R.; McAllister, R. & Levine, S. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models Advances in Neural Information Processing Systems (NIPS), 2018, 4754-4765

•Williams, G.; Wagener, N.; Goldfain, B.; Drews, P.; Rehg, J. M.; Boots, B. & Theodorou, E. A. Information theoretic MPC for model-based reinforcement learning International Conference on Robotics and Automation (ICRA), 2017

•Watter, M.; Springenberg, J.; Boedecker, J. & Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images Advances in neural information processing systems (NIPS), 2015, 2746-2754

•Finn, C. & Levine, S. Deep visual foresight for planning robot motion IEEE International Conference on Robotics and Automation (ICRA), 2017, 2786-2793

•Nagabandi, A.; Kahn, G.; Fearing, R. S. & Levine, S. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning ArXiv e-prints, 2017

•Bansal, S.; Calandra, R.; Xiao, T.; Levine, S. & Tomlin, C. J. Goal-Driven Dynamics Learning via Bayesian Optimization IEEE Conference on Decision and Control (CDC), 2017, 5168-5173