Learning Policies by Imitating Optimal Control

CS 294-112: Deep Reinforcement Learning

Week 3, Lecture 2

Sergey Levine

Overview

1. Last time: learning models of system dynamics and using optimal control to choose actions
   - Global models and model-based RL
   - Local models and model-based RL with *constraints*
2. What if we want a *policy*?
   - Much quicker to evaluate actions at runtime
   - Potentially better generalization
3. Can we just backpropagate into the policy?
4. How does this relate to imitation learning?

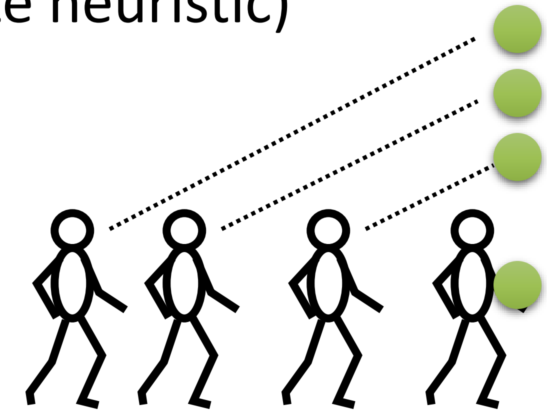
Today's Lecture

1. Backpropagating into a policy with learned models
2. How this becomes equivalent to *imitating* optimal control
3. The guided policy search algorithm
4. Imitating optimal control with DAgger
5. Limitations & considerations

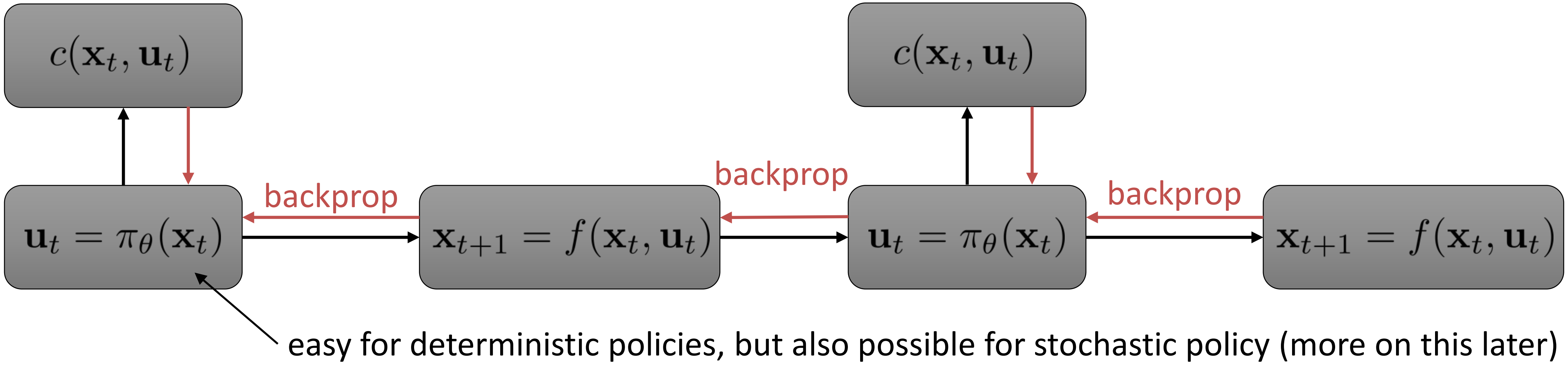
- Goals
  - Understand how to train policies using optimal control
  - Understand tradeoffs between various methods

So how can we train policies?

- So far we saw how we can…
  - Train global models (e.g. GPs)
  - Train local models (e.g. linear models)
  - Combine global and local models (e.g. using Bayesian linear regression)
- But what if we want a policy?
  - Don't need to replan (faster)
  - Potentially better generalization (e.g. gaze heuristic)



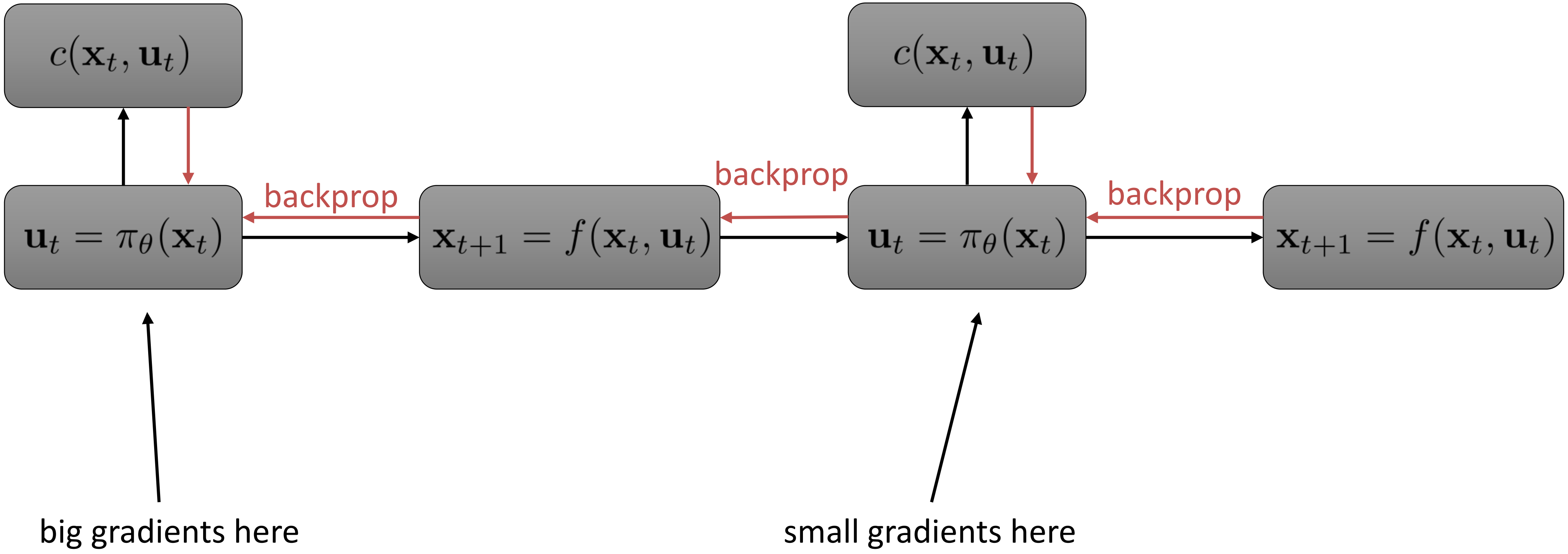
Backpropagate directly into the policy?



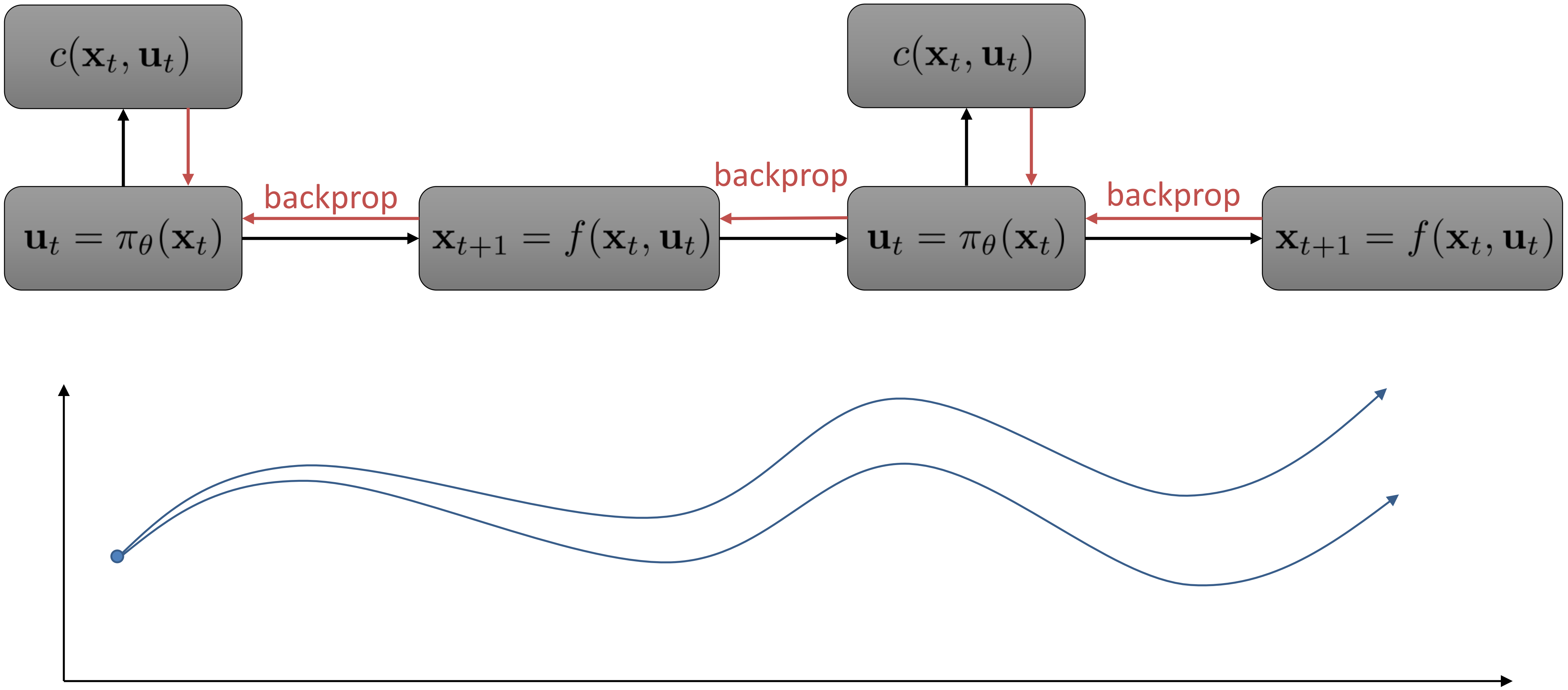
model-based reinforcement learning version 2.0:

1. run base policy $\pi_0(\mathbf{u}_t|\mathbf{x}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_i\}$
2. learn dynamics model $f(\mathbf{x}, \mathbf{u})$ to minimize $\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}'_i\|^2$
3. backpropagate through $f(\mathbf{x}, \mathbf{u})$ into the policy to optimize $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$
4. run $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$, appending the visited tuples $(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ to $\mathcal{D}$

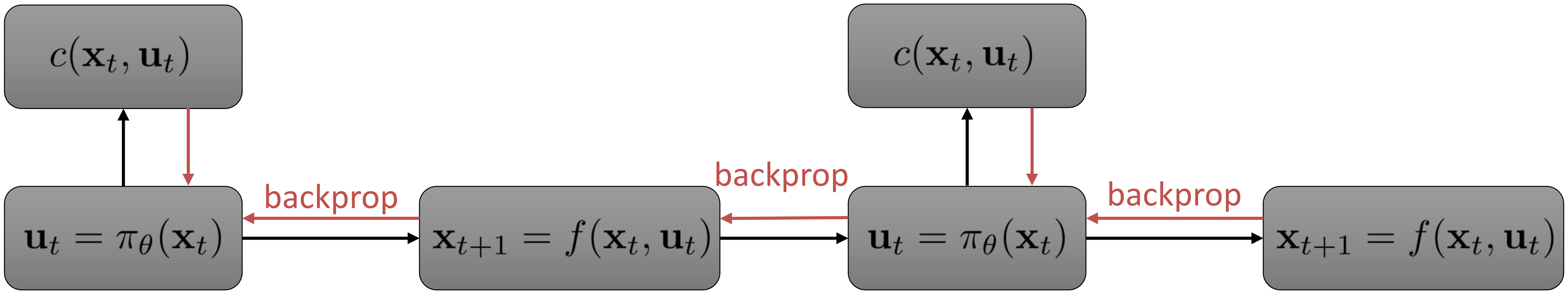
What's the problem with backprop into policy?



What's the problem?



What’s the problem?

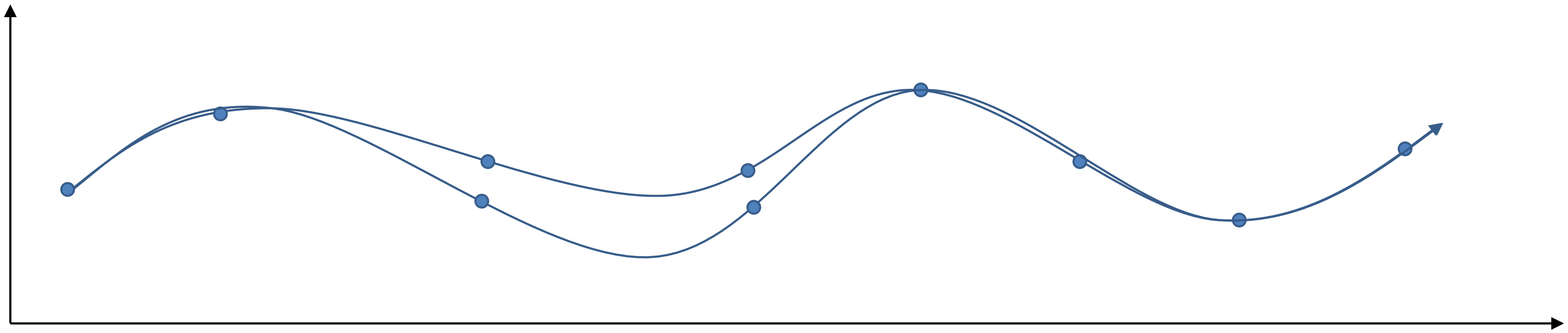


- Similar parameter sensitivity problems as shooting methods
  - But no longer have convenient second order LQR-like method, because policy parameters couple all the time steps, so no dynamic programming
- Similar problems to training long RNNs with BPTT
  - Vanishing and exploding gradients
  - Unlike LSTM, we can’t just “choose” a simple dynamics, dynamics are chosen by nature

What's the problem?

- What about collocation methods?

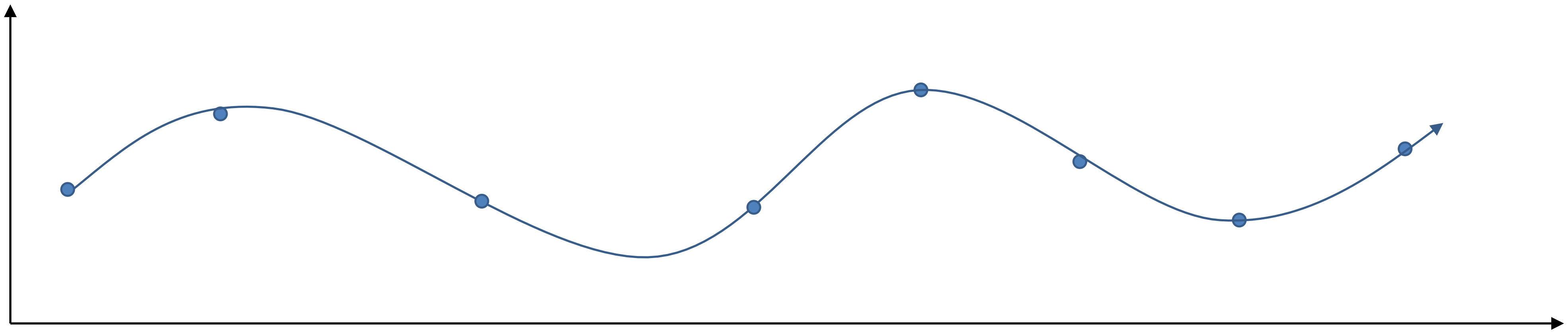
$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T,\mathbf{x}_1,\ldots,\mathbf{x}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$



What's the problem?

- What about collocation methods?

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T,\mathbf{x}_1,\ldots,\mathbf{x}_T,\theta} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}), \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$



Even simpler…

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T,\mathbf{x}_1,\ldots,\mathbf{x}_T,\theta} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

generic trajectory optimization, solve however you want

$$\text{s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

- How can we impose constraints on trajectory optimization?

Review: dual gradient descent

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } C(\mathbf{x}) = 0$$

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda C(\mathbf{x})$$

$$g(\lambda) = \mathcal{L}(\mathbf{x}^\star(\lambda), \lambda)$$

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda)$$

$$\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\mathbf{x}^\star, \lambda)$$

1. Find $\mathbf{x}^\star \leftarrow \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda)$
2. Compute $\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\mathbf{x}^\star, \lambda)$
3. $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$

A small tweak to DGD: augmented Lagrangian

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } C(\mathbf{x}) = 0$$

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda C(\mathbf{x})$$

$$\bar{\mathcal{L}}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda C(\mathbf{x}) + \rho \|C(\mathbf{x})\|^2$$

- Still converges to correct solution
- When far from solution, quadratic term tends to improve stability
- Closely related to alternating direction method of multipliers (ADMM)

1. Find $\mathbf{x}^\star \leftarrow \arg\min_{\mathbf{x}} \bar{\mathcal{L}}(\mathbf{x}, \lambda)$
2. Compute $\frac{dg}{d\lambda} = \frac{d\bar{\mathcal{L}}}{d\lambda}(\mathbf{x}^\star, \lambda)$
3. $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$

Constraining trajectory optimization with dual gradient descent

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

$$\mathcal{L}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t)$$

$$\bar{\mathcal{L}}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t) + \sum_{t=1}^{T} \rho_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t)^2$$

Constraining trajectory optimization with dual gradient descent

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

$$\bar{\mathcal{L}}(\tau,\theta,\lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t) + \sum_{t=1}^{T} \rho_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t)^2$$

1. Find $\tau \leftarrow \arg\min_\tau \bar{\mathcal{L}}(\tau,\theta,\lambda)$ (e.g. via iLQR)
2. Find $\theta \leftarrow \arg\min_\theta \bar{\mathcal{L}}(\tau,\theta,\lambda)$ (e.g. via SGD)
3. $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$

Guided policy search discussion

1. Find $\tau \leftarrow \arg\min_\tau \bar{\mathcal{L}}(\tau, \theta, \lambda)$ (e.g. via iLQR)
2. Find $\theta \leftarrow \arg\min_\theta \bar{\mathcal{L}}(\tau, \theta, \lambda)$ (e.g. via SGD)
3. $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$

- Can be interpreted as *constrained* trajectory optimization method
- Can be interpreted as imitation of an optimal control expert, since step 2 is just supervised learning
- The optimal control "teacher" adapts to the learner, and avoids actions that the learner can't mimic

General guided policy search scheme

1. Optimize $p(\tau)$ with respect to some surrogate $\tilde{c}(\mathbf{x}_t, \mathbf{u}_t)$
2. Optimize θ with respect to some supervised objective
3. Increment or modify dual variables λ

Need to choose:

form of $p(\tau)$

optimization method for $p(\tau)$

surrogate $\tilde{c}(\mathbf{x}_t, \mathbf{u}_t)$

supervised objective for $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$

Stochastic (Gaussian) GPS

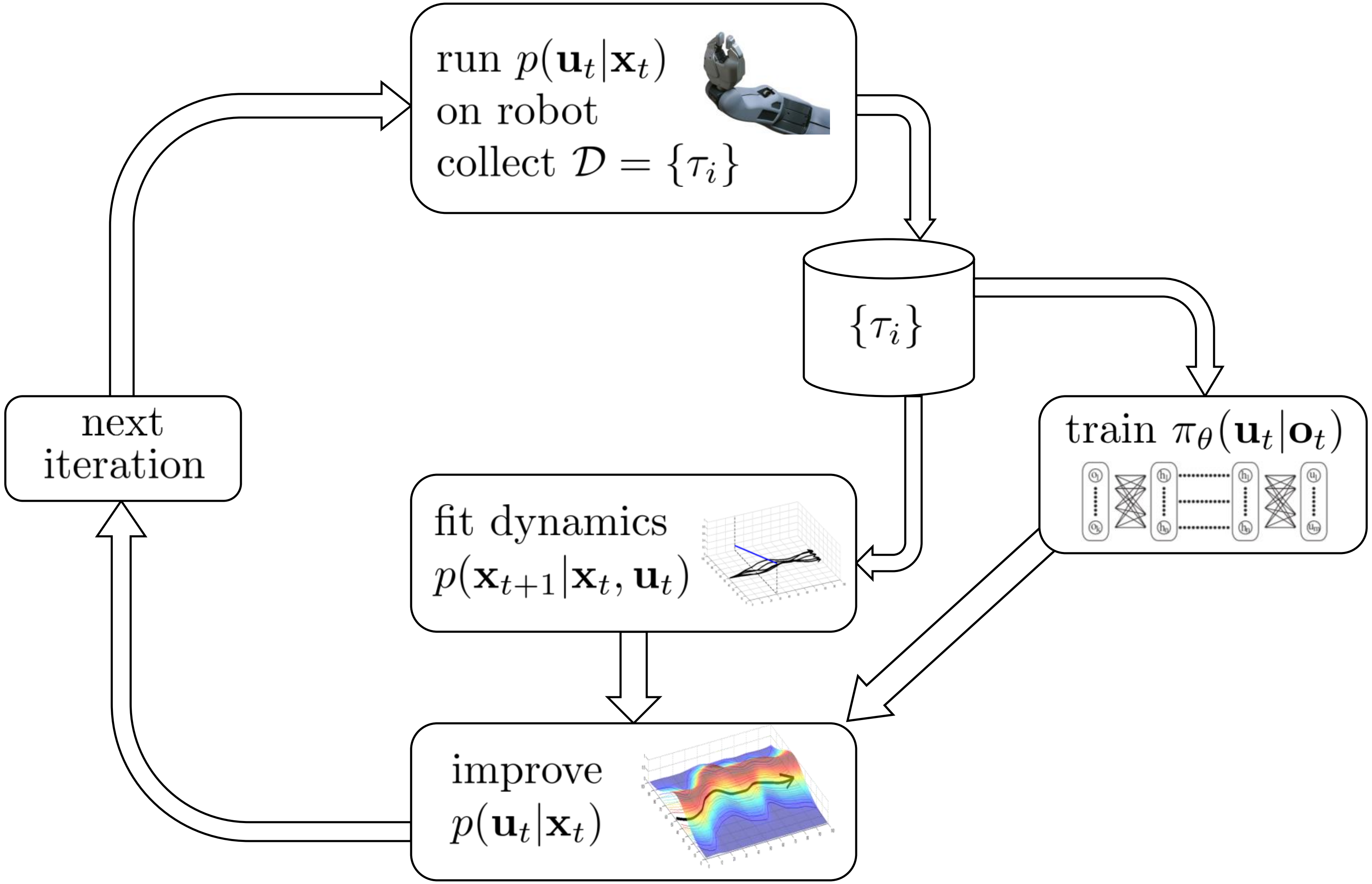
$$\min_{p,\theta} E_{\tau \sim p(\tau)}[c(\tau)] \text{ s.t. } p(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$$

$$p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \mathbf{k}_t + \hat{\mathbf{u}}_t, \Sigma_t)$$

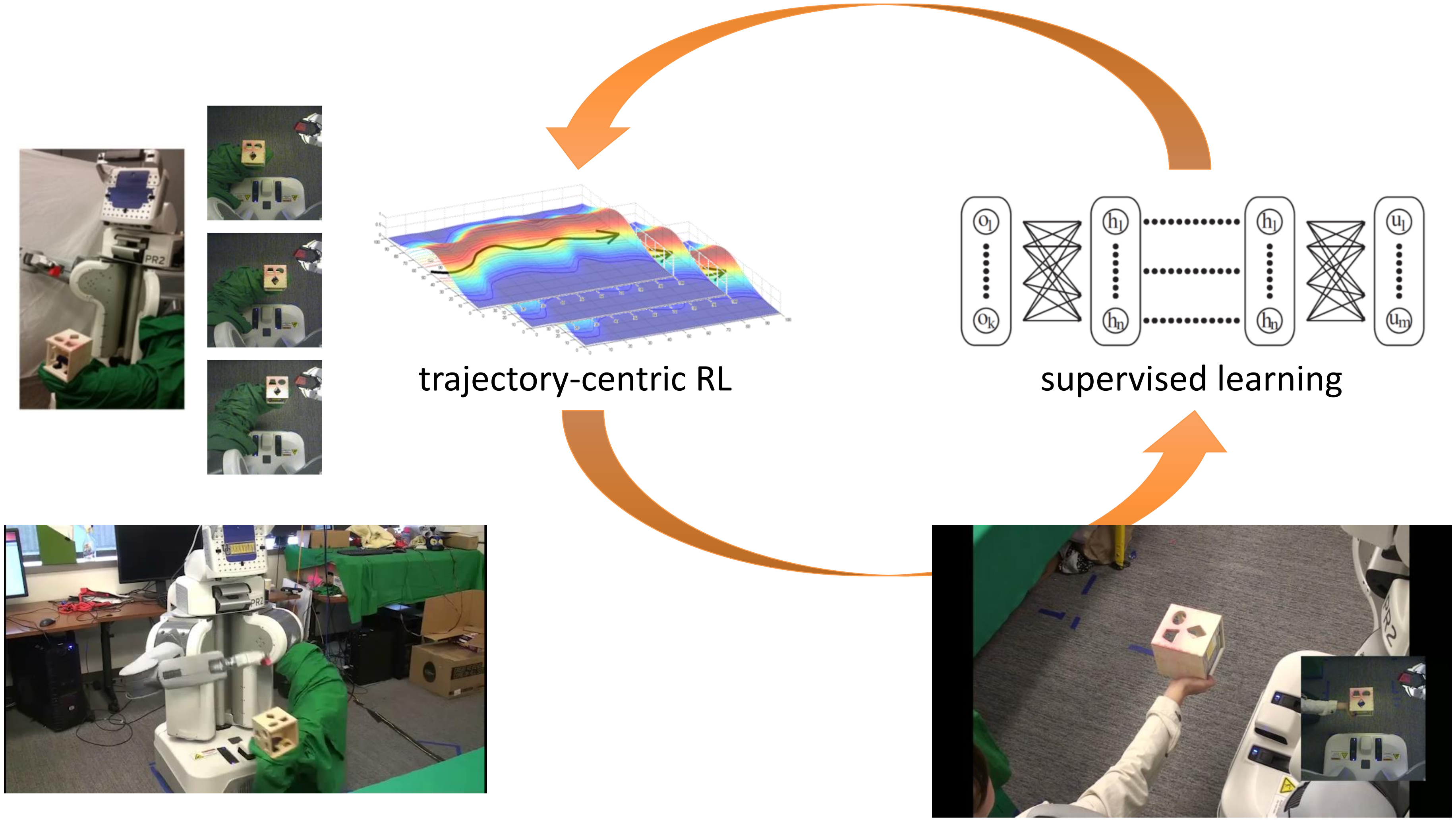
$$\min_p \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\tilde{c}(\mathbf{x}_t,\mathbf{u}_t)] \text{ s.t. } D_{\text{KL}}(p(\tau)\|\bar{p}(\tau)) \leq \epsilon$$

1. Optimize $p(\tau)$ with respect to some surrogate $\tilde{c}(\mathbf{x}_t, \mathbf{u}_t)$
2. Optimize θ with respect to some supervised objective
3. Increment or modify dual variables λ

Stochastic (Gaussian) GPS with local models



Robotics Example

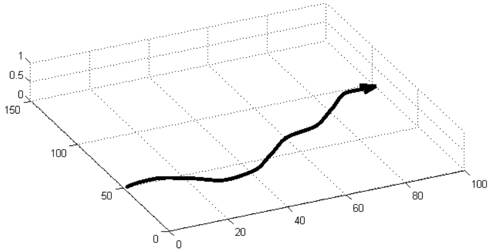


Input Remapping Trick

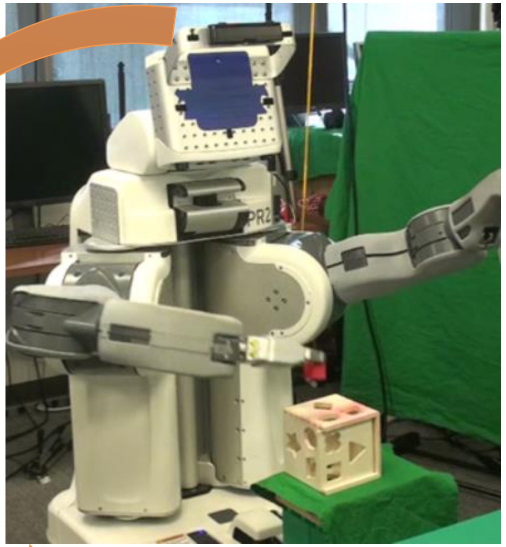
$$\min_{p,\theta} E_{\tau\sim p(\tau)}[c(\tau)] \text{ s.t. } p(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$$

training time

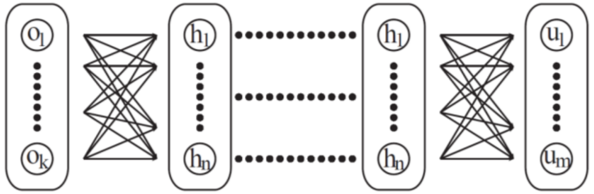
$\mathbf{x}_t \rightarrow \mathbf{u}_t$



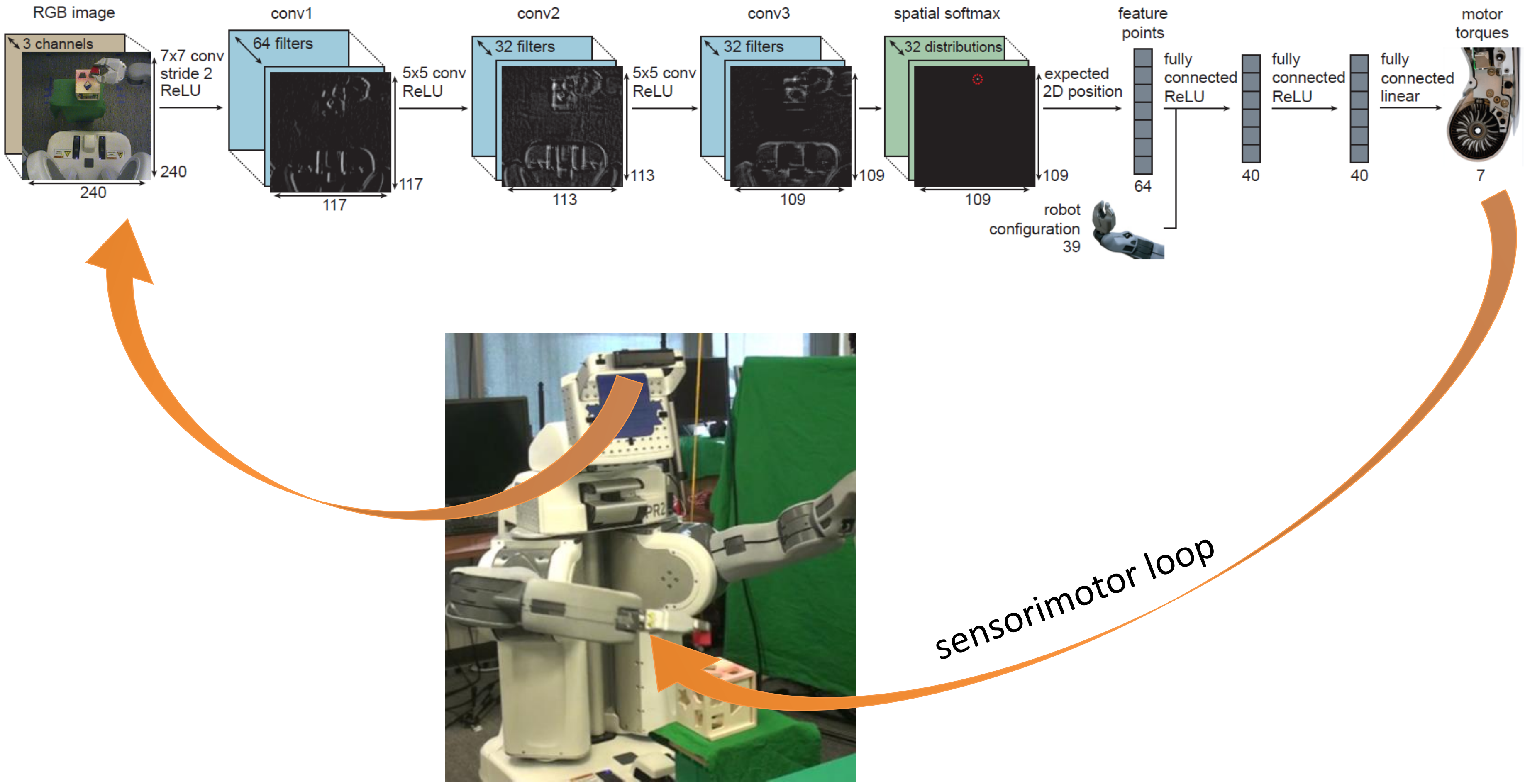
test time



$\mathbf{o}_t \rightarrow \mathbf{u}_t$



CNN Vision-Based Policy



Case study: vision-based control with GPS

End-to-End Training of Deep Visuomotor Policies

Sergey Levine† SVLEVINE@EECS.BERKELEY.EDU
Chelsea Finn† CBFINN@EECS.BERKELEY.EDU
Trevor Darrell TREVOR@EECS.BERKELEY.EDU
Pieter Abbeel PABBEEL@EECS.BERKELEY.EDU
Division of Computer Science
University of California
Berkeley, CA 94720-1776, USA
†These authors contributed equally.

Case study: vision-based control with GPS

Imitating optimal control with DAgger

Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning

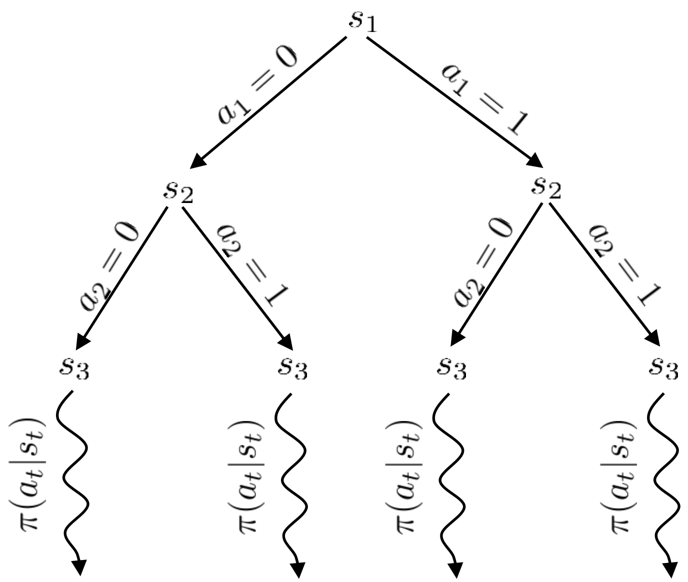
Xiaoxiao Guo
Computer Science and Eng.
University of Michigan
guoxiao@umich.edu

Satinder Singh
Computer Science and Eng.
University of Michigan
baveja@umich.edu

Honglak Lee
Computer Science and Eng.
University of Michigan
honglak@umich.edu

Richard Lewis
Department of Psychology
University of Michigan
rickl@umich.edu

Xiaoshi Wang
Computer Science and Eng.
University of Michigan
xiaoshiw@umich.edu



A problem with DAgger

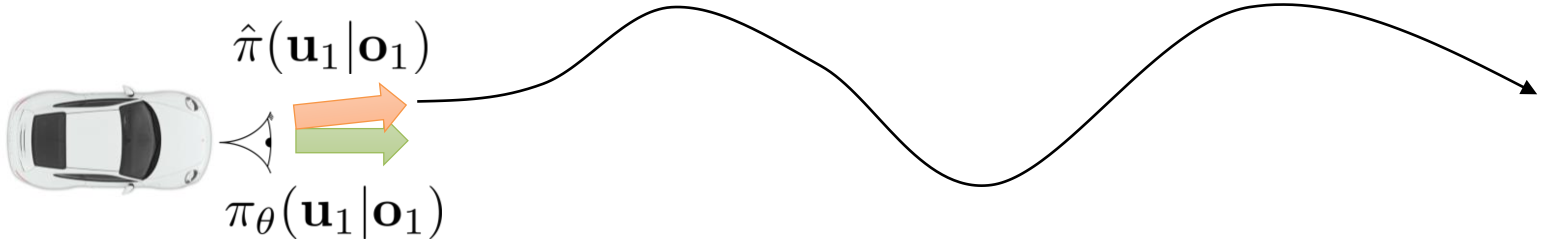
1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



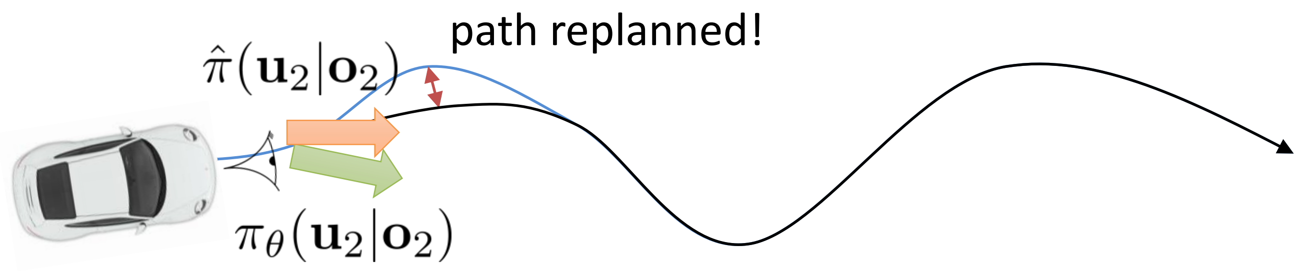
Kahn, Zhang, Levine, Abbeel '16

Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\text{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

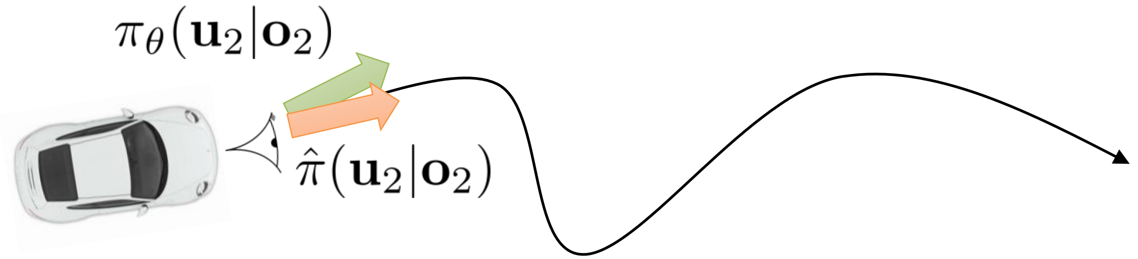


Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

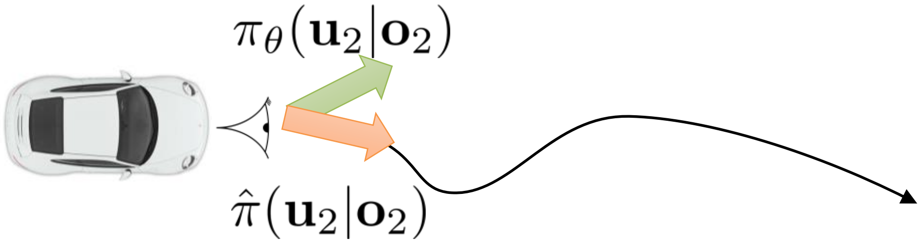


Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

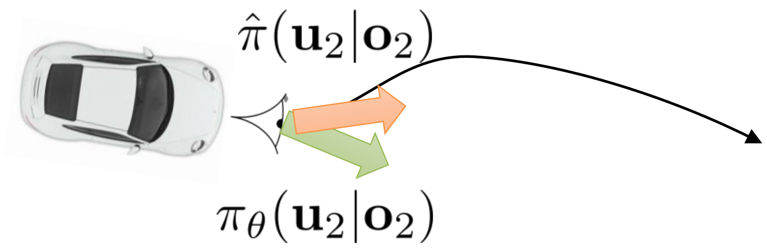


Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

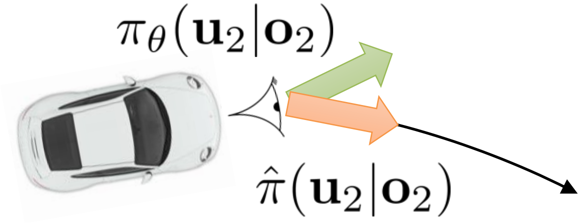


Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

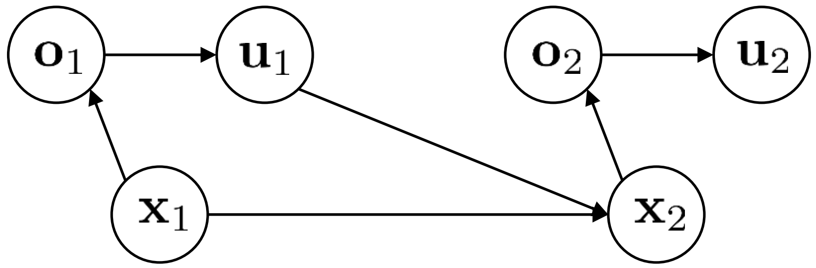
simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t)\|\pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

replanning = **M**odel **P**redictive **C**ontrol (MPC)

$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ – control from **images**

$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t)$ – control from **states**

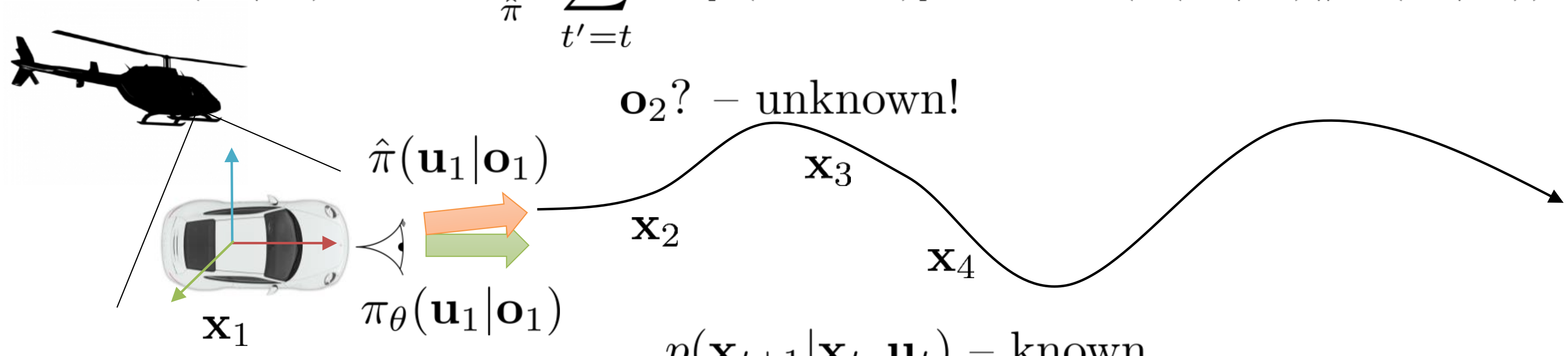


Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ – known

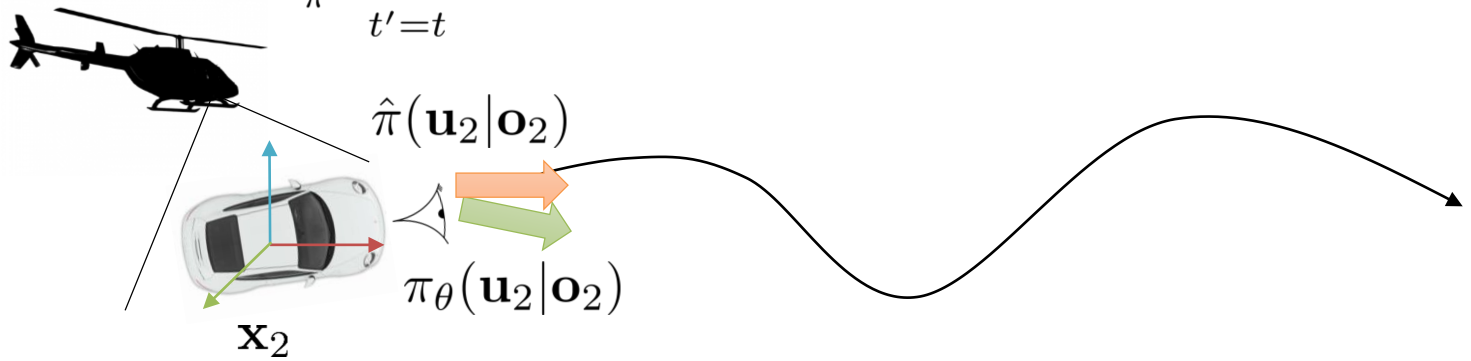
$p(\mathbf{o}_t|\mathbf{x}_t)$ – unknown

Imitating MPC: PLATO algorithm

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

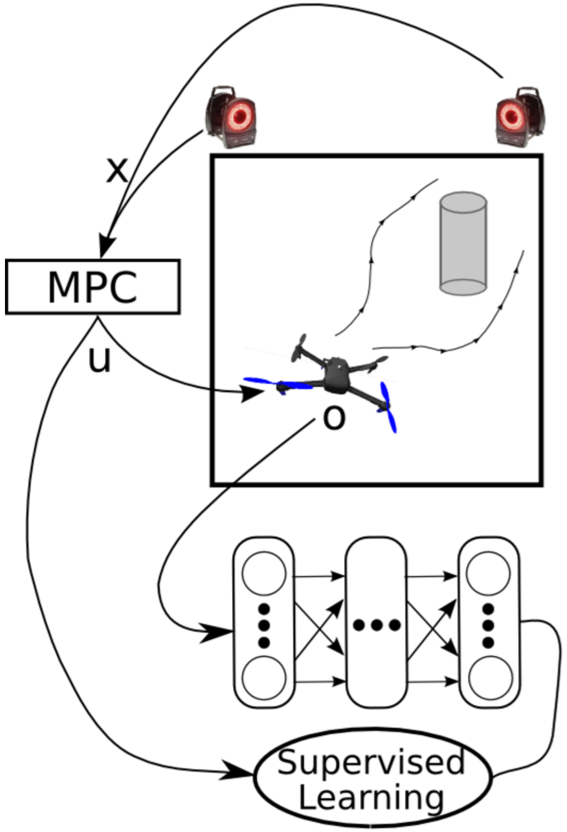
simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

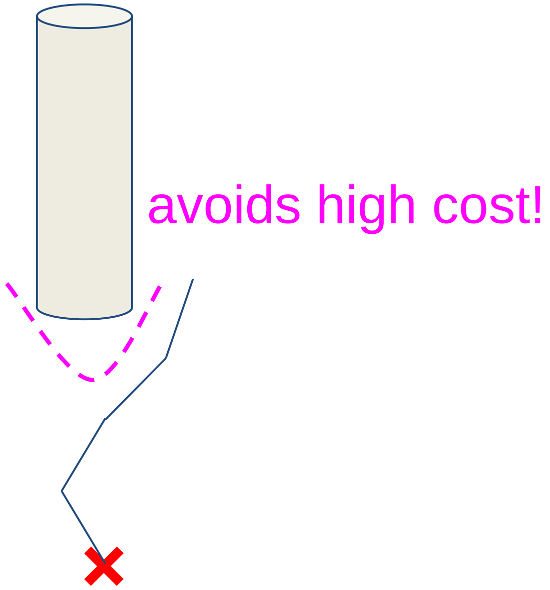
$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



Imitating MPC: PLATO algorithm

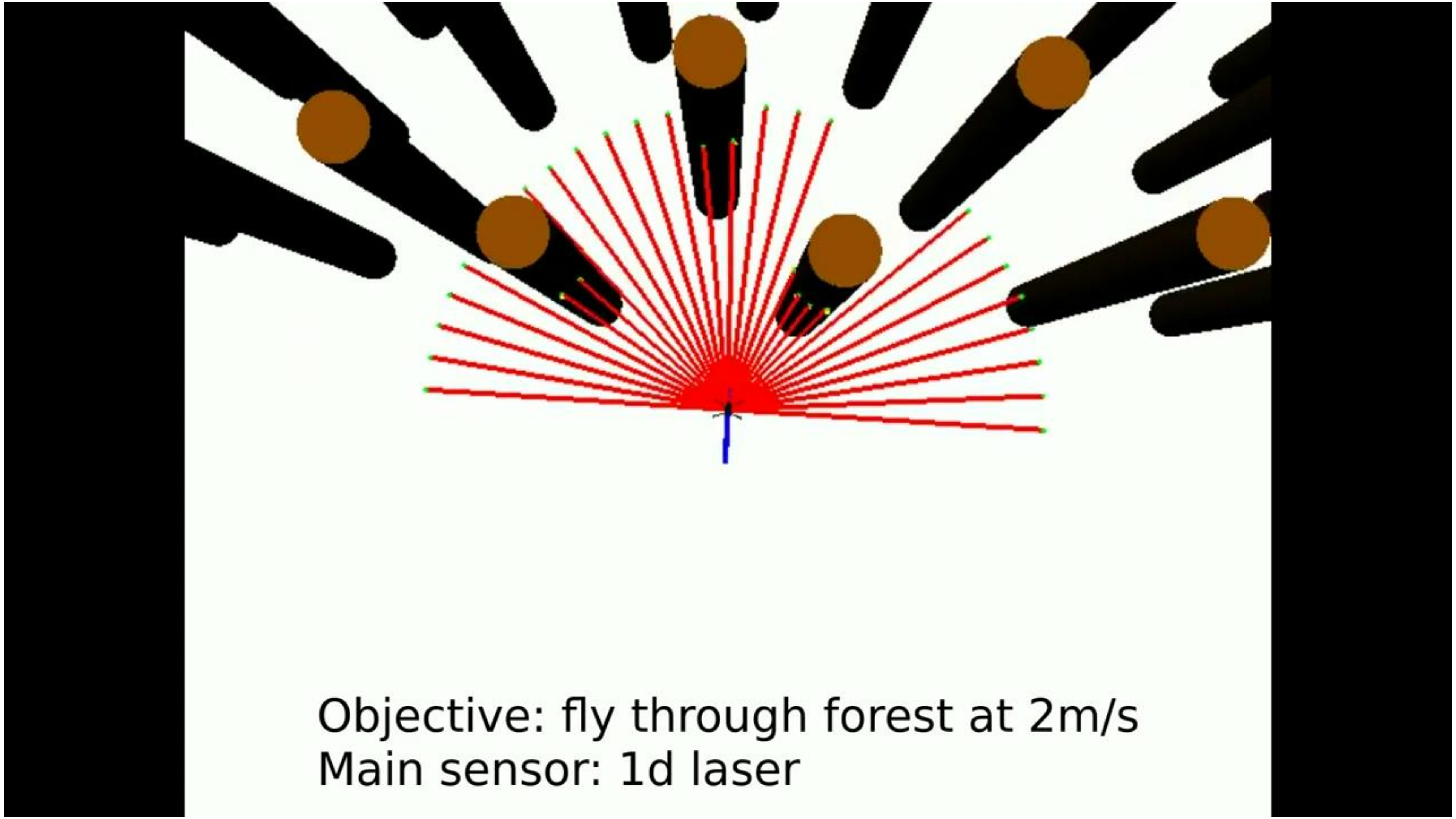
$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} \underline{E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})]} + \underline{\lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t)\|\pi_\theta(\mathbf{u}_t|\mathbf{o}_t))}$$





input substitution trick
need state at training time
but not at test time!

Imitating MPC: PLATO algorithm



DAgger vs GPS

- DAgger does not require an adaptive expert
  - Any expert will do, so long as states from learned policy can be labeled
  - Assumes it is possible to match expert's behavior up to bounded loss
    - Not always possible (e.g. partially observed domains)
- GPS adapts the "expert" behavior
  - Does not require bounded loss on initial expert (expert will change)

Why imitate optimal control?

- Relatively stable and easy to use
  - Supervised learning works very well
  - Optimal control (usually) works very well
  - The combination of the two (usually) works very well
- Input remapping trick: can exploit availability of additional information at training time to learn policy from raw observations
- Overcomes optimization challenges of backpropagating into policy directly
- Usually sample-efficient and viable for real physical systems

Limitations of model-based RL

- Need some kind of model
  - Not always available
  - Sometimes harder to learn than the policy
- Learning the model takes time & data
  - Sometimes expressive model classes (neural nets) are not fast
  - Sometimes fast model classes (linear models) are not expressive
- Some kind of additional assumptions
  - Linearizability/continuity
  - Ability to reset the system (for local linear models)
  - Smoothness (for GP-style global models)
  - Etc.

Model-free RL: trial and error learning

- What if we didn't need a model?
- Intuition: trial and error learning
- Much slower
- Often more general
- Coming up next!