A Tour of Reinforcement Learning
The View from Continuous Control

Benjamin Recht
University of California, Berkeley

trustable, scalable, predictable

Control Theory 🤔

~~Reinforcement Learning~~ is the study of how to use past data to enhance the future manipulation of a dynamical system

Disciplinary Biases

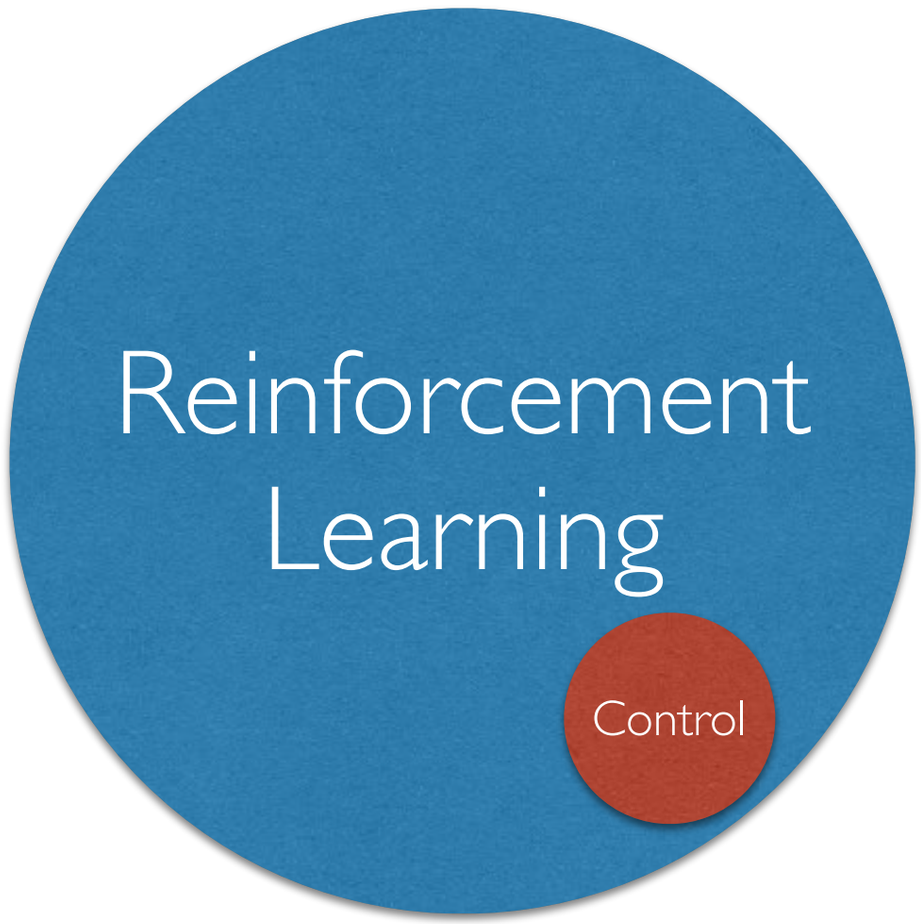
AE/CE/EE/ME

continuous

model ⟶ *action*

IEEE Transactions

CS



discrete

data ⟶ *action*

Science Magazine

Disciplinary Biases

AE/CE/EE/ME

CS

Today's talk will try to unify these camps and point out how to merge their perspectives.

continuous

model ⟶ *action*

IEEE Transactions

discrete

data ⟶ *action*

Science Magazine



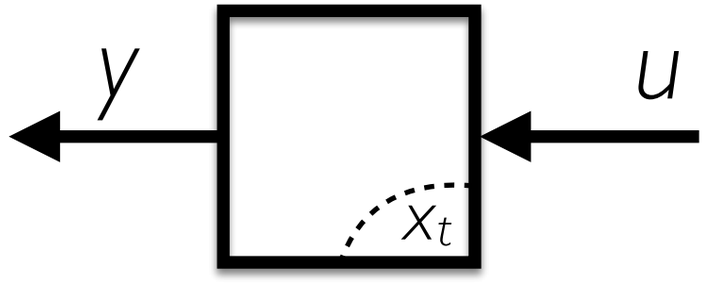
Main research challenge: What are the fundamental limits of learning systems that interact with the physical environment?

How well must we understand a system in order to control it?

theoretical foundations

- statistical learning theory
- robust control theory
- core optimization

Control theory is the study of dynamical systems with inputs



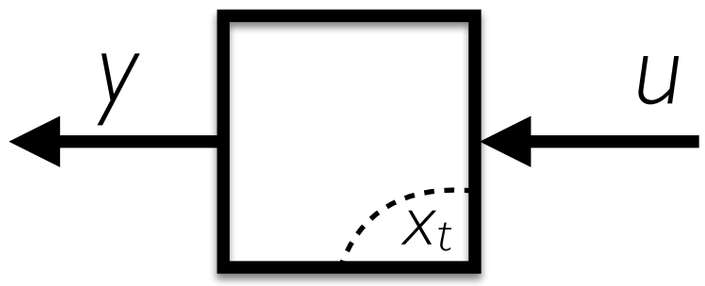
$$x_{t+1} = f(x_t, u_t)$$

x_t is called the *state*, and the dimension of the state is called the *degree*, *d*.

u_t is called the *input*, and the dimension is p.

Reinforcement Learning

~~Control theory~~ is the study of discrete dynamical systems with inputs



$$p(x_{t+1} \mid \text{past}) = p(x_{t+1} \mid x_t, u_t)$$

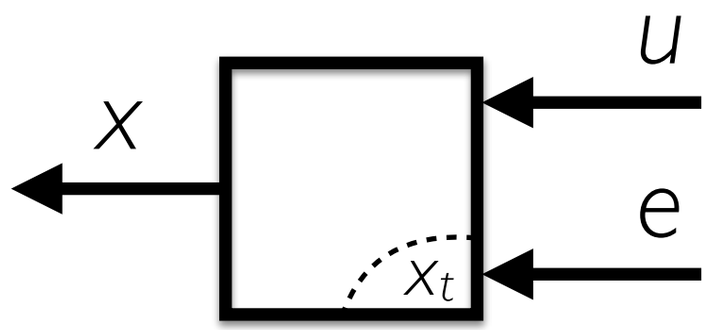
Markov Decision Process (MDP)

x_t is the *state*, and it takes values in [d]

u_t is called the *input*, and takes values in [p].

Optimal control

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$



C_t is the *cost.* If you maximize, it's called a *reward.*

e_t is a noise process

f_t is the state-transition function

$\tau_t = (u_1, \ldots, u_{t-1}, x_0, \ldots, x_t)$ is an observed *trajectory*

$\pi_t(\tau_t)$ is the *policy.* This is the optimization decision variable.

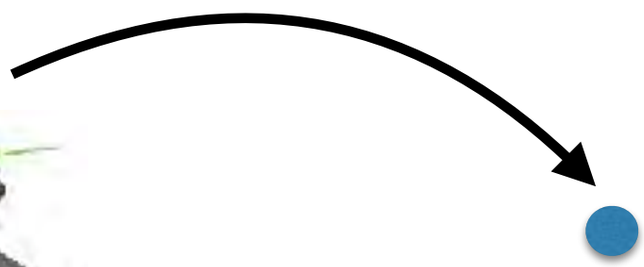
Newton's Laws

$$z_{t+1} = z_t + v_t$$
$$v_{t+1} = v_t + a_t$$
$$m\, a_t = u_t$$

$$\text{minimize} \quad \sum_{t=0}^{T} \mathbb{1}_{|(x_t)_1| > \epsilon}$$

$$\text{subject to} \quad x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t$$

$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$



Newton's Laws

$$z_{t+1} = z_t + v_t$$
$$v_{t+1} = v_t + a_t$$
$$m\, a_t = u_t$$

$$\text{minimize} \quad \sum_{t=0}^{T} (x_t)_1^2 + r u_t^2$$

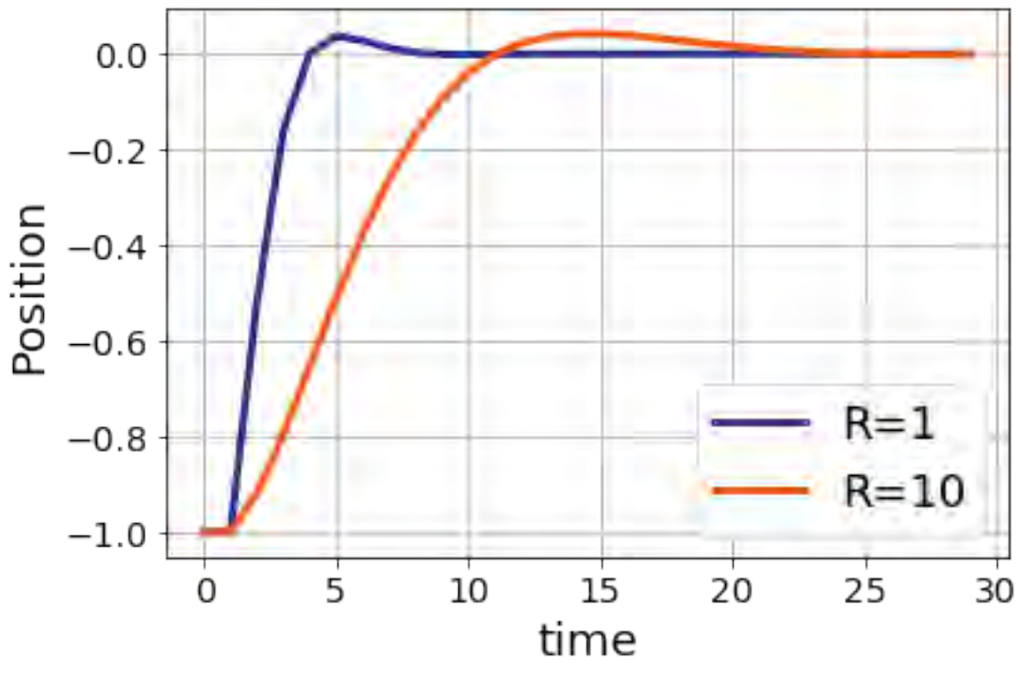
$$\text{subject to} \quad x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t$$

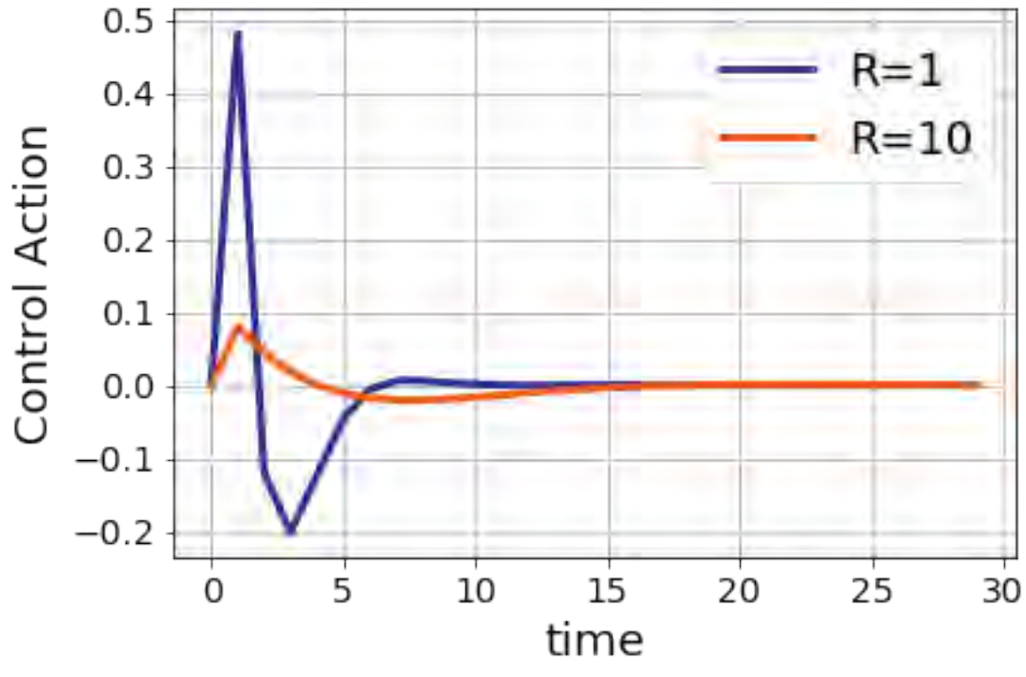
$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$

$$\text{minimize} \quad \sum_{t=0}^{T} (x_t)_1^2 + r u_t^2$$

$$\text{subject to} \quad x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t$$

$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$





"Simplest" Example: Linear Quadratic Regulator

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right] \leftarrow \textit{quadratic cost} \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t + e_t \leftarrow \textit{linear dynamics} \end{aligned}$$

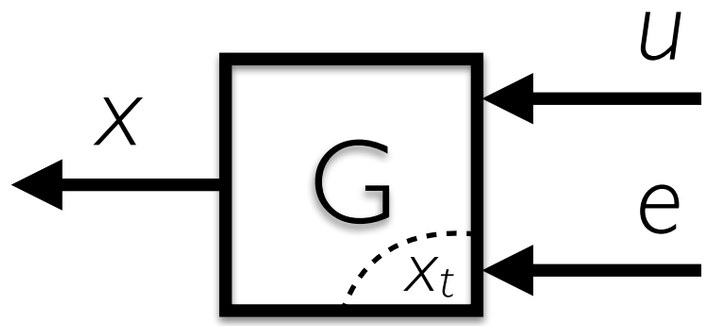
$$\text{minimize} \quad \sum_{t=0}^{T} (x_t)_1^2 + r u_t^2$$

$$\text{subject to} \quad x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t$$

$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$

Optimal control

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$



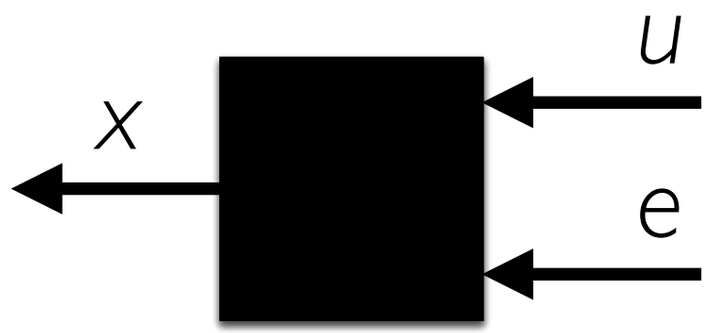
generic solutions with known dynamics:

Batch Optimization

Dynamic Programming

Learning to control

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$



C_t is the *cost*. If you maximize, it's called a *reward*.

e_t is a noise process

f_t is the state-transition function *unknown!*

$\tau_t = (u_1, \ldots, u_{t-1}, x_0, \ldots, x_t)$ is an observed *trajectory*

$\pi_t(\tau_t)$ is the *policy*. This is the optimization decision variable.

Major challenge: how to perform optimal control when the system is unknown?

Today: Reinvent RL attempting to answer this question

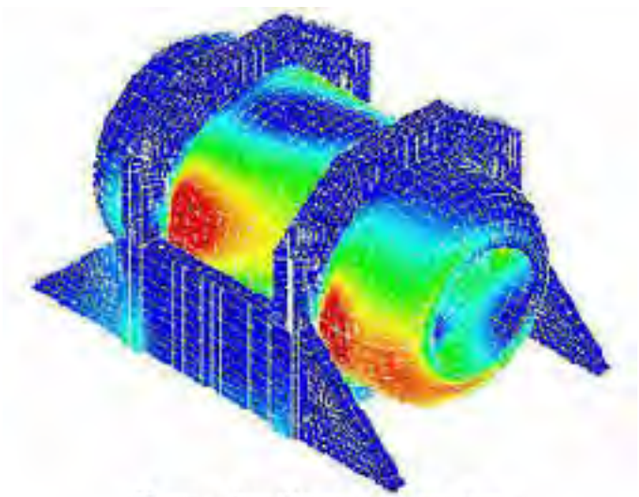
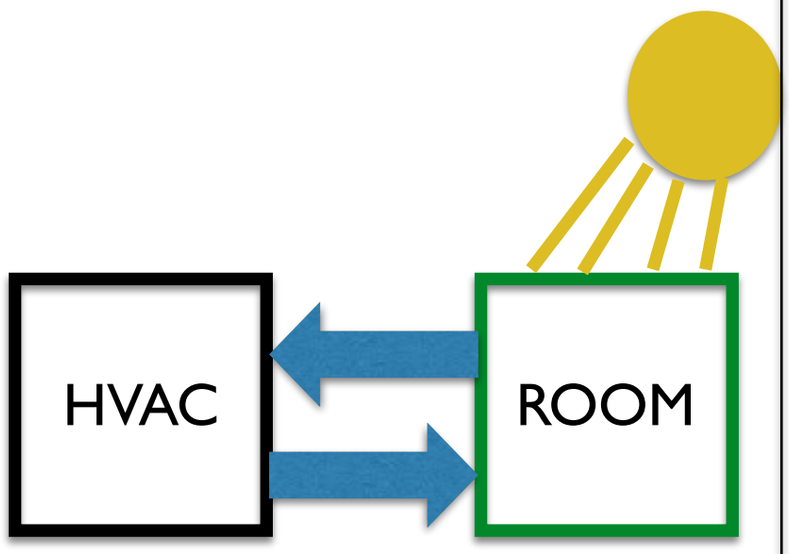
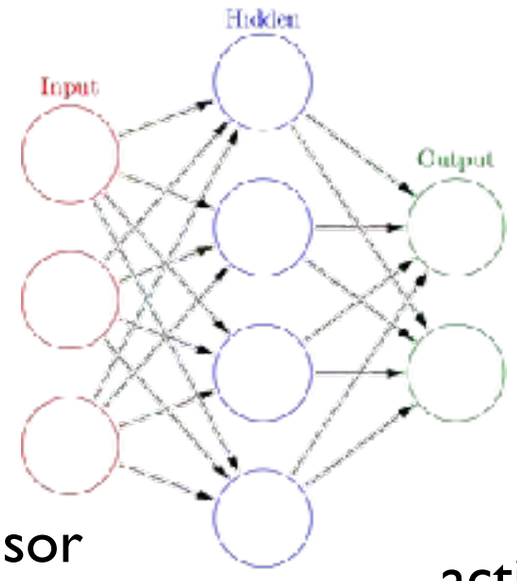


Figure 1 http://www.noranenz.com

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = \nabla \cdot \tau + \rho \mathbf{g}$$



$$M\dot{T} = \dot{Q} + \dot{m}_s c_p (T_s - T)$$



sensor
state

action

Identify everything

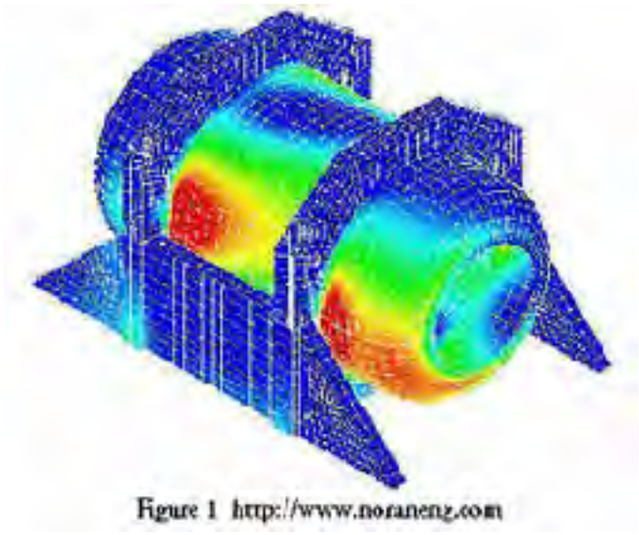
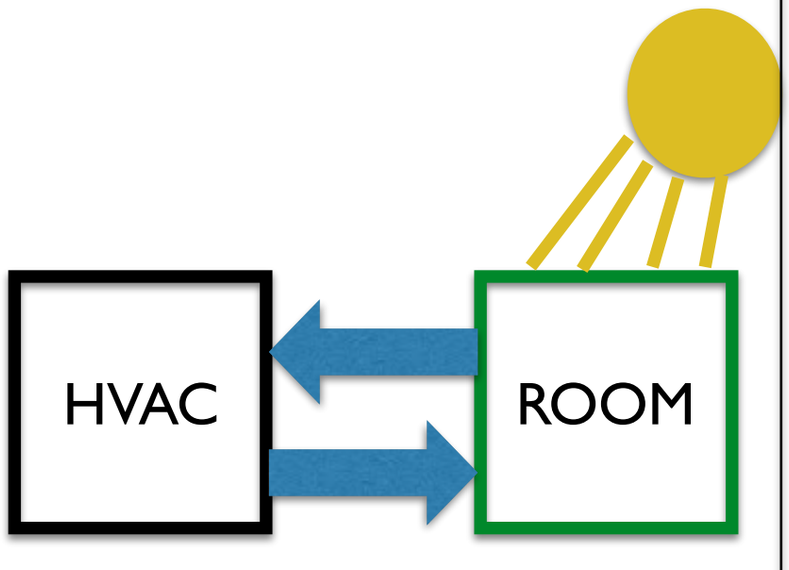


Figure 1 http://www.noranenz.com

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = \nabla \cdot \tau + \rho \mathbf{g}$$

- PDE control
- High performance aerodynamics

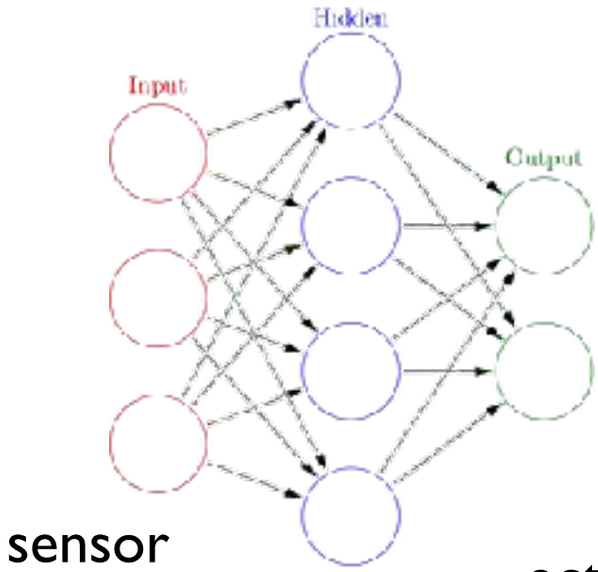
Identify a coarse model



$$M\dot{T} = \dot{Q} + \dot{m}_s c_p (T_s - T)$$

- model predictive control

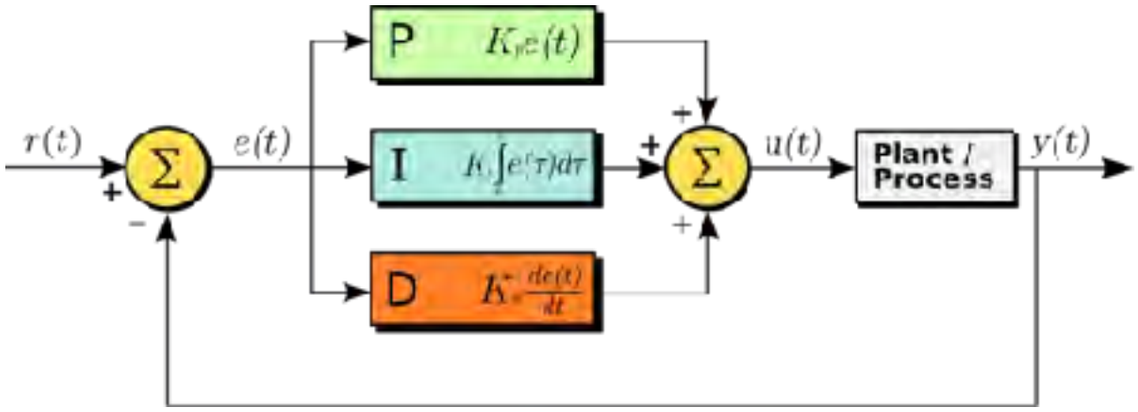
We don't need no stinking models!

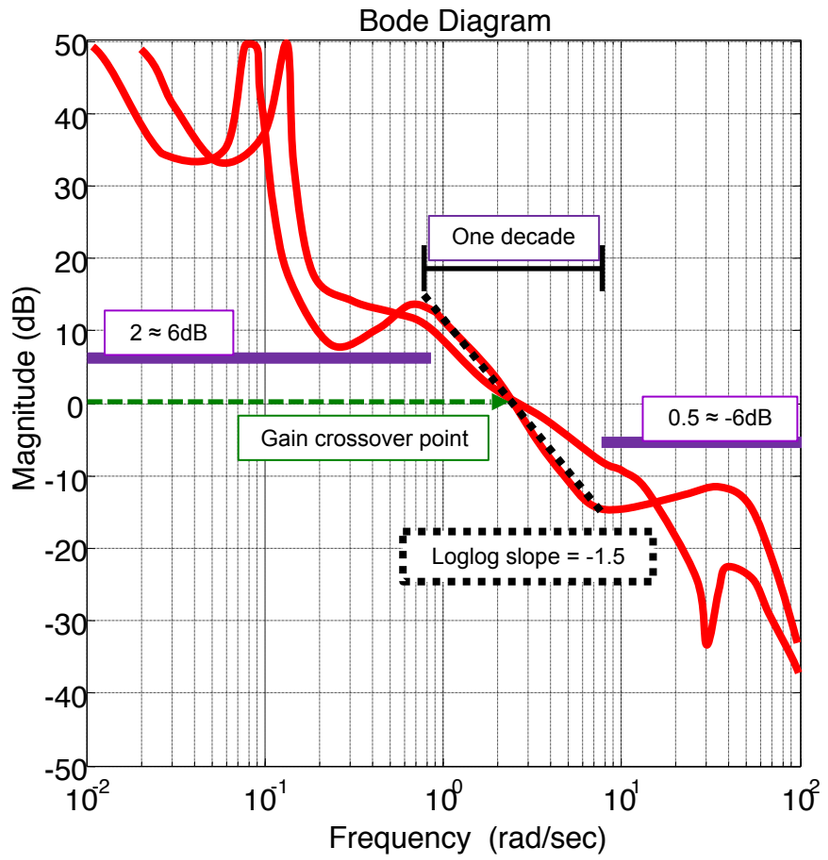


- reinforcement learning
- PID control?

We need robust fundamentals to distinguish these approaches

But PID control works…





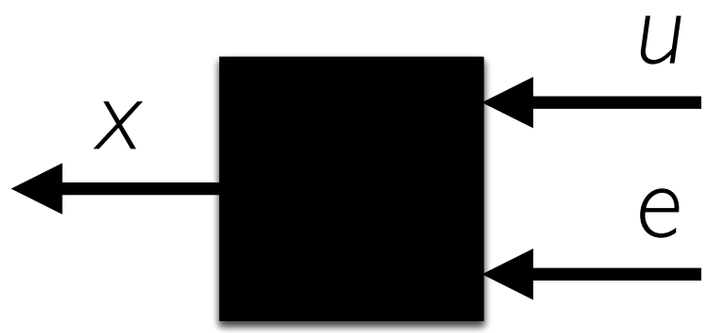
2 parameters suffice for 95% of all control applications.

How much needs to be modeled for more advanced control?

Can we learn to compensate for poor models, changing conditions?

Learning to control

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$



C_t is the *cost*. If you maximize, it's called a *reward*.

e_t is a noise process

f_t is the state-transition function *unknown!*

$\tau_t = (u_1, \ldots, u_{t-1}, x_0, \ldots, x_t)$ is an observed *trajectory*

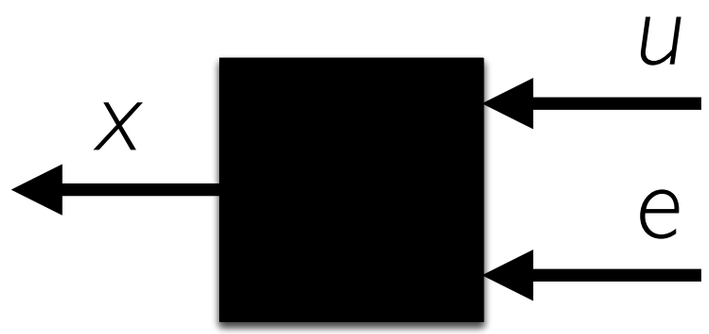
$\pi_t(\tau_t)$ is the *policy*. This is the optimization decision variable.

Major challenge: how to perform optimal control when the system is unknown?

Today: Reinvent RL attempting to answer this question

Learning to control

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$



Oracle: You can generate N trajectories of length T.

Challenge: Build a controller with smallest error with fixed sampling budget (N x T).

What is the optimal estimation/design scheme?

How many samples are needed for near optimal control?

The Linearization Principle

If a machine learning algorithm does crazy things when restricted to linear models, it's going to do crazy things on complex nonlinear models too.

Would you believe someone had a good SAT solver if it couldn't solve 2-SAT?

This has been a fruitful research direction:

- Recurrent neural networks (Hardt, Ma, R. 2016)
- Generalization and Margin in Neural Nets (Zhang *et al* 2017)
- Residual Networks (Hardt and Ma 2017)
- Bayesian Optimization (Jamieson *et al* 2017)
- Adaptive gradient methods (Wilson *et al* 2017)

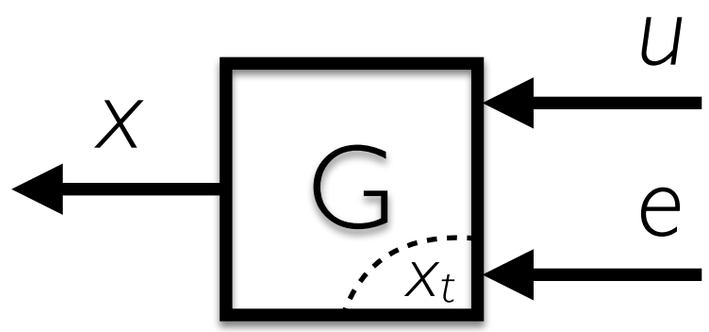
"Simplest" Example: LQR

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right] \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t + e_t \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \sum_{t=0}^{T} (x_t)_1^2 + r u_t^2 \\ \text{subject to} \quad & x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t \end{aligned}$$

$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$

RL Methods



minimize $\mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right]$ ← approximate dynamic programming

s.t. $x_{t+1} = f_t(x_t, u_t, e_t)$ ← model-based

$u_t = \pi_t(\tau_t)$ ← direct policy search

How to solve optimal control when the model *f* is unknown?

- **Model-based:** fit model from data
- **Model-free**
  - **Approximate dynamic programming:** estimate cost from data
  - **Direct policy search:** search for actions from data

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

Model-based RL

Collect some simulation data. Should have $x_{t+1} \approx \varphi(x_t, u_t) + \nu_t$

Fit dynamics with *supervised learning:* $\hat{\varphi} = \arg\min_{\varphi} \sum_{t=0}^{N-1} ||x_{t+1} - \varphi(x_t, u_t)||^2$

Solve approximate problem:

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_\omega\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = \varphi(x_t, u_t) + \omega_t \\ & u_t = \pi(\tau_t) \end{aligned}$$

minimize $\mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right]$

s.t. $x_{t+1} = f_t(x_t, u_t, e_t)$

$u_t = \pi_t(\tau_t)$

~~Approximate~~ Dynamic Programming

Both the methods and analyses are complicated, but this is the core of classical RL.

Sadly, if you don't already know it, this probably won't make a ton of sense until the sixth time you see it…

Dynamic Programming

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t) + C_f(x_{T+1}, u_{T+1})\right] =: \mathcal{Q}_1(x, u) \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t),\ x_1 = x \\ & u_t = \pi_t(\tau_t),\ u_1 = u \end{aligned}$$

"Q-function"

Terminal Q-function:

$$\mathcal{Q}_{T+1}(x, u) = C_f(x, u)$$

Recursive formula (recurse backwards):

$$\mathcal{Q}_k(x, u) = C_k(x, u) + \min_{u'} \mathbb{E}_e\left[\mathcal{Q}_{k+1}(f_k(x, u, e), u')\right]$$

Optimal Policy:

$$\pi_k(\tau_k) = \arg\min_{u} \mathcal{Q}_k(x_k, u)$$

"Simplest" Example: LQR

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}\left[\sum_{t=1}^{T-1} x_t^* Q x_t + u_t^* R u_t + x_T^* P_T x_T\right] \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t + \mathrm{e}_t \end{aligned}$$

Dynamic Programming: $\mathcal{Q}_T(x,u) = x^* P_T x$

$$\begin{aligned} \mathcal{Q}_t(x,u) &= C_t(x,u) + \min_{u'} \mathbb{E}_{\mathrm{e}}\left[\mathcal{Q}_{t+1}(f_t(x,u,\mathrm{e}),u')\right] \\ &= x^* Q x + u^* R u + (Ax+Bu)^* P_{t+1}(Ax+Bu) + c_t \end{aligned}$$

$$P_t = Q + A^* P_{t+1} A - A^* P_{t+1} B \left(R + B^* P_{t+1} B\right)^{-1} B^* P_{t+1} A$$

$$u_t = -(B^* P_{t+1} B + R)^{-1} B^* P_{t+1} A x_t =: K_t x_t$$

"Simplest" Example: LQR

$$\begin{aligned} \text{minimize} \quad & \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right] \\ \text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t + e_t \end{aligned}$$

When (A,B) known, optimal to build control $u_t = Kx_t$

$$u_t = -(B^*PB + R)^{-1} B^*PAx_t =: Kx_t$$

$$P = Q + A^*PA - A^*PB\left(R + B^*PB\right)^{-1} B^*PA$$

Discrete Algebraic Riccati Equation

- Dynamic programming has simple form because quadratics are miraculous.
- Solution is independent of noise variance.
- For finite time horizons, we could solve this with a variety of batch solvers.
- Note that the solution is only time invariant on the infinite time horizon.

Approximate Dynamic Programming

minimize $\mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right]$

s.t. $x_{t+1} = f_t(x_t, u_t, e_t)$

$u_t = \pi_t(\tau_t)$

Recursive formula: $\mathcal{Q}_k(x, u) = C_k(x, u) + \mathbb{E}_e\left[\min_{u'} \mathcal{Q}_{k+1}(f_k(x, u, e), u')\right]$

Optimal Policy: $\pi_k(\tau_k) = \arg\min_u \mathcal{Q}_k(x_k, u)$

Approximate Dynamic Programming

discount factor

minimize $\mathbb{E}_e\left[\sum_{t=1}^{\infty} \gamma^t C(x_t, u_t)\right]$

s.t. $x_{t+1} = f(x_t, u_t, e_t)$

$u_t = \pi(\tau_t)$

Bellman Equation: $\mathcal{Q}(x,u) = C(x,u) + \gamma \mathbb{E}_e\left[\min_{u'} \mathcal{Q}(f(x,u,e),u')\right]$

Optimal Policy: $\pi(x) = \arg\min_u \mathcal{Q}(x,u)$

Generate algorithms using the insight:

$$Q(x_k,u_k) \approx C(x_k,u_k) + \gamma \min_{u'} Q(x_{k+1},u') + \nu_k$$

Q-learning: *stochastic approximation*

$$Q_{\text{new}}(x_k,u_k) = (1-\eta)Q_{\text{old}}(x_k,u_k) - \eta\left(C(x_k,u_k) + \gamma \min_{u'} Q_{\text{old}}(x_{k+1},u')\right)$$

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

Direct Policy Search

Sampling to Search

$$\min_{z \in \mathbb{R}^d} \quad \Phi(z) \quad = \quad \min_{p(z)} \quad \mathbb{E}_p[\Phi(z)]$$

$$\leq \quad \min_{\vartheta} \quad \mathbb{E}_{p(z;\vartheta)}[\Phi(z)] \quad =: J(\vartheta)$$

- Search over probability distributions
- Use function approximations that might not capture optimal distribution
- Can build (incredibly high variance) stochastic gradient estimates by sampling:

$$\nabla J(\vartheta) = \mathbb{E}_{p(z;\vartheta)} \left[\Phi(z) \nabla_\vartheta \log p(z;\vartheta) \right]$$

Reinforce Algorithm

$$J(\vartheta) := \mathbb{E}_{p(z;\vartheta)}[\Phi(z)]$$

$$\begin{aligned}
\nabla_\vartheta J(\vartheta) &= \int \Phi(z) \nabla_\vartheta p(z;\vartheta) dz \\
&= \int \Phi(z) \left(\frac{\nabla_\vartheta p(z;\vartheta)}{p(z;\vartheta)} \right) p(z;\vartheta) dz \\
&= \int \left(\Phi(z) \nabla_\vartheta \log p(z;\vartheta) \right) p(z;\vartheta) dz \\
&= \mathbb{E}_{p(z;\vartheta)} \left[\Phi(z) \nabla_\vartheta \log p(z;\vartheta) \right]
\end{aligned}$$

Reinforce Algorithm

$$J(\vartheta) := \mathbb{E}_{p(z;\vartheta)}[\Phi(z)]$$

$$\nabla J(\vartheta) = \mathbb{E}_{p(z;\vartheta)}\left[\Phi(z)\nabla_\vartheta \log p(z;\vartheta)\right]$$

Sample	$z_k \sim p(z;\vartheta_k)$
Compute	$G(z_k,\vartheta_k) = \Phi(z_k)\nabla_{\vartheta_k} \log p(z_k;\vartheta_k)$
Update	$\vartheta_{k+1} = \vartheta_k - \alpha_k G(z_k,\vartheta_k)$

Reinforce Algorithm

$$J(\vartheta) := \mathbb{E}_{p(z;\vartheta)}[\Phi(z)]$$

Sample $z_k \sim p(z;\vartheta_k)$

Compute $G(z_k,\vartheta_k) = \Phi(z_k)\nabla_{\vartheta_k}\log p(z_k;\vartheta_k)$

Update $\vartheta_{k+1} = \vartheta_k - \alpha_k G(z_k,\vartheta_k)$

Generic algorithm for solving discrete optimization:

$$z \in \{-1,1\}^d \qquad p(z;\vartheta) = \prod_{i=1}^{d} \frac{\exp(z_i\vartheta_i)}{\exp(-\vartheta_i)+\exp(\vartheta_i)}$$

$$\vartheta_{k+1} = \vartheta_k - \alpha_k\Phi(z_k)(z_k + \tanh(\vartheta_k))$$

Does this "solve" *any* discrete problem?

Direct Policy Search

Random Search

minimize $\mathbb{E}_{e_t,\omega}\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right]$

s.t. $x_{t+1} = f_t(x_t, u_t, e_t)$

$u_t = \pi(\tau_t; \vartheta + \omega)$

parameter perturbation

$$G(\omega, \vartheta) = \left(\sum_{t=1}^{T} C(x_t, u_t)\right) \nabla \log p(\omega)$$

$$G^{(m)}(\omega, \vartheta) = \frac{1}{m} \sum_{i=1}^{m} \frac{C(\vartheta + \sigma\omega_i) - C(\vartheta - \sigma\omega_i)}{2\sigma} \omega_i$$

$$C(\vartheta) = \sum_{t=1}^{T} C(x_t, u_t)$$

$$\omega_i \sim \mathcal{N}(0, I)$$

random finite difference approximation to the gradient

aka… (μ,λ)-Evolution Strategies
SPSA
Bandit Convex Opt

Random Search for LQR

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right] \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t + e_t \end{aligned}$$

"Greedy strategy": Build control $u_t = K x_t$

- **Sample a random perturbation:** $\nu \sim \mathcal{N}(0, \sigma^2 I)$
- **Collect samples from control** $u_t = (K + \nu) x_t : \tau = \{x_1, \ldots, x_T\}$
- **Compute cost:** $J(\tau) = \sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t$
- **Update:** $K \leftarrow K - \alpha_t J(\tau)\ \nu$

Policy Gradient

minimize $\mathbb{E}_{e_t,u_t}\left[\sum_{t=1}^{T} C_t(x_t,u_t)\right]$

s.t. $x_{t+1} = f_t(x_t,u_t,e_t)$

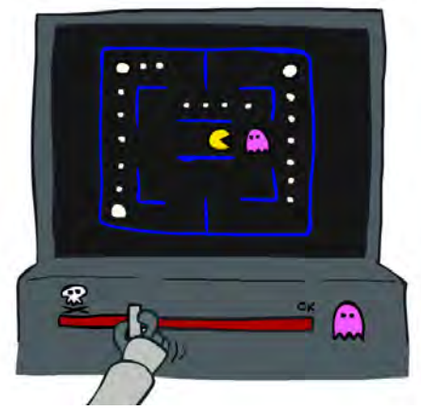
$u_t \sim p(u|x_t;\vartheta)$

probabilistic policy

Direct Policy Search

$$G(\tau,\vartheta) = \left(\sum_{t=1}^{T} C(x_t,u_t)\right) \cdot \left(\sum_{t=0}^{T-1} \nabla_\vartheta \log p_\vartheta(u_t|x_t;\vartheta)\right)$$

Policy Gradient for LQR



$$\begin{aligned} \text{minimize} \quad & \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right] \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t + e_t \end{aligned}$$

"Greedy strategy": Build control $u_t = K x_t$

- **Sample a bunch of random vectors:** $\nu_t \sim \mathcal{N}(0, \sigma^2 I)$
- **Collect samples from control** $u_t = K x_t + \nu_t$**:** $\tau = \{x_1, \ldots, x_T\}$
- **Compute cost:** $C(\tau) = \sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t$
- **Update:** $K_{\text{new}} \leftarrow K_{\text{old}} - \alpha_t \, C(\tau) \sum_{t=0}^{T-1} \nu_t x_t^*$

policy gradient only has access to 0-th order information!!!

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

Direct Policy Search

Policy Gradient

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_{e_t, u_t}\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t \sim p(u|x_t; \vartheta) \end{aligned}$$

probabilistic policy

Random Search

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_{e_t, \omega}\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi(\tau_t; \vartheta + \omega) \end{aligned}$$

parameter perturbation

- Reinforce applied to either problems does not depend on the dynamics.
- Both are Derivative-free algorithms!

Direct Policy Search

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e \left[\sum_{t=1}^{T} C_t(x_t, u_t) \right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

- Reinforce is NOT Magic
  - What is the variance?
  - What is the approximation error?
  - Necessarily becomes derivative free as you are accessing the decision variable by sampling
- But it's certainly super easy!

Sample Complexity?

Discrete MDPs:

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} \sim p(x|x_t, u_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

ADP

model-based

policy search

$x \in [d]$
$u \in [p]$

Algorithm Class	Samples per iteration	Parameters	"optimal" error after T
Model-based	1	d^2p	$\sqrt{\frac{d^2p}{T}}$
ADP	1	dp	$\sqrt{\frac{dp}{T}}$
Policy search	1	dp	$\sqrt{\frac{dp}{T}}$

Sample Complexity?

Continuous Control:

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

ADP

model-based

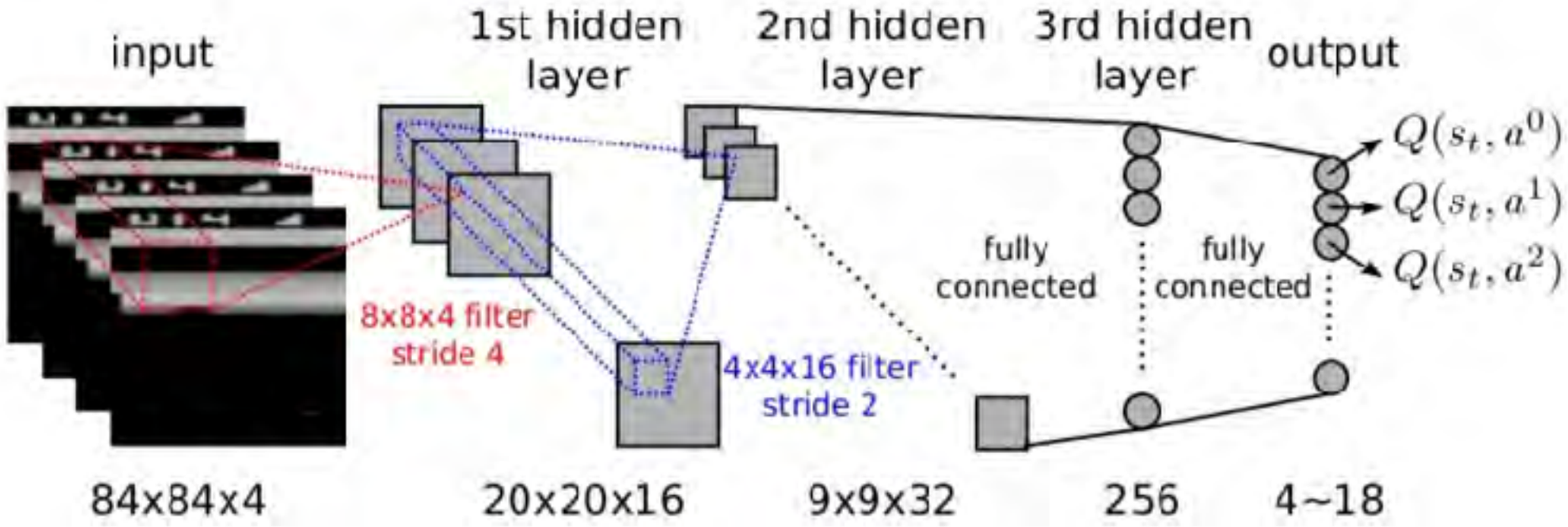
policy search

$x \in \mathbb{R}^d$

$u \in \mathbb{R}^p$

Algorithm	Samples per iteration	LQR parameters	"optimal" error after T
Model-based	d	d^2+dp	$C\sqrt{\frac{d+p}{T}}$
ADP	1	$\binom{d+p}{2}$	$C\frac{d+p}{\sqrt{T}}$
Policy search	1	dp	$C\sqrt{\frac{dp}{T}}$

Deep Reinforcement Learning



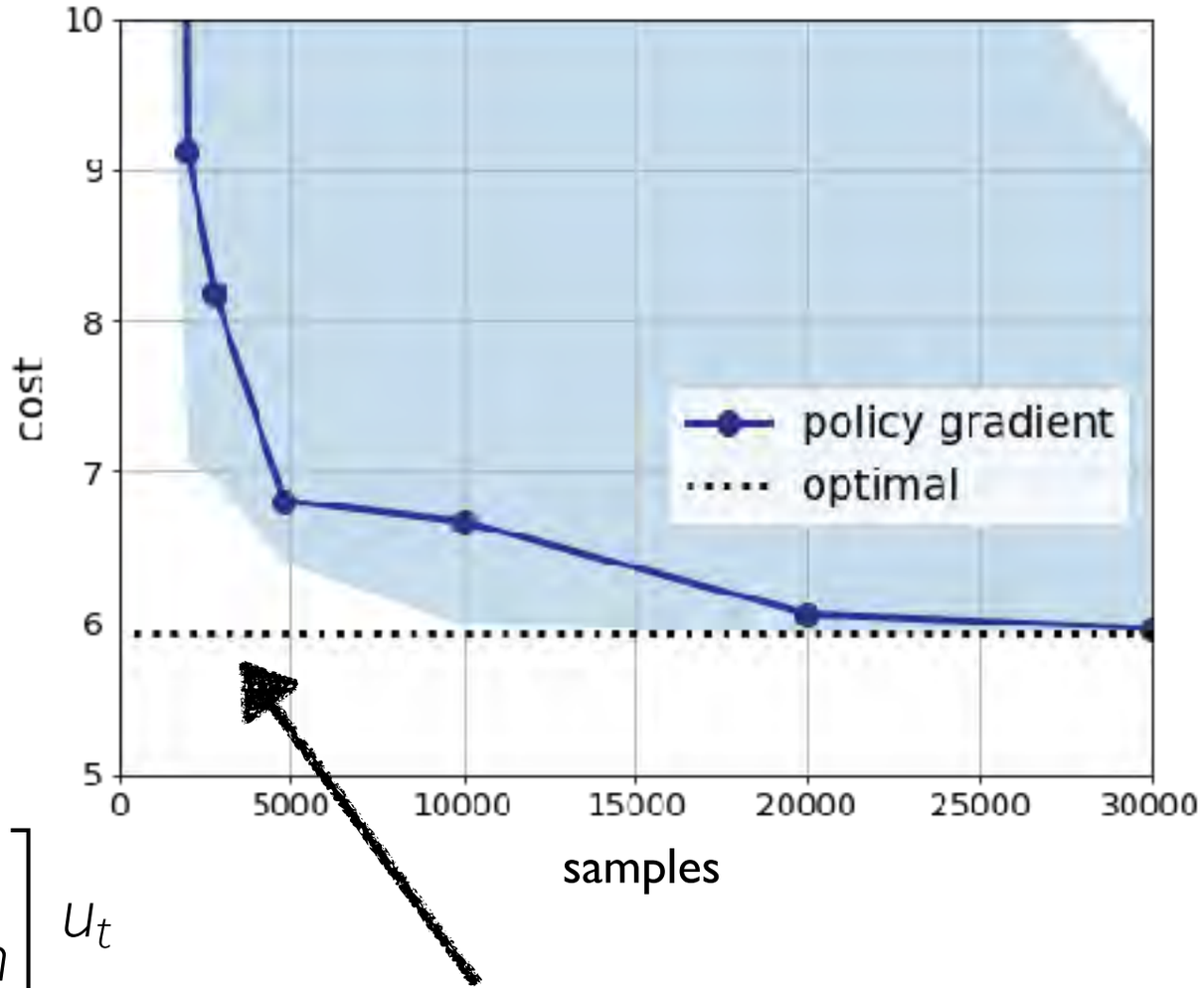
- Simply parameterize Q-function or policy as a deep net
- Note, ADP is tricky to analyze with function approximation
- Policy search is considerably more straightforward: make the log-prob a deep net.

"Simplest" Example: LQR

$$\text{minimize} \quad \sum_{t=0}^{T} (x_t)_1^2 + r u_t^2$$

$$\text{subject to} \quad x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t$$

$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$



Model-based and ADP with 10 samples

Extraordinary Claims Require Extraordinary Evidence*

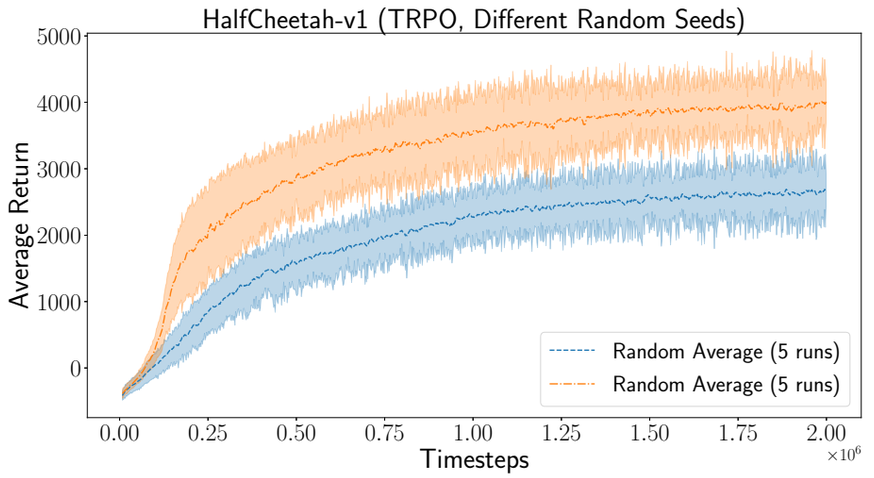
* only if your prior is correct

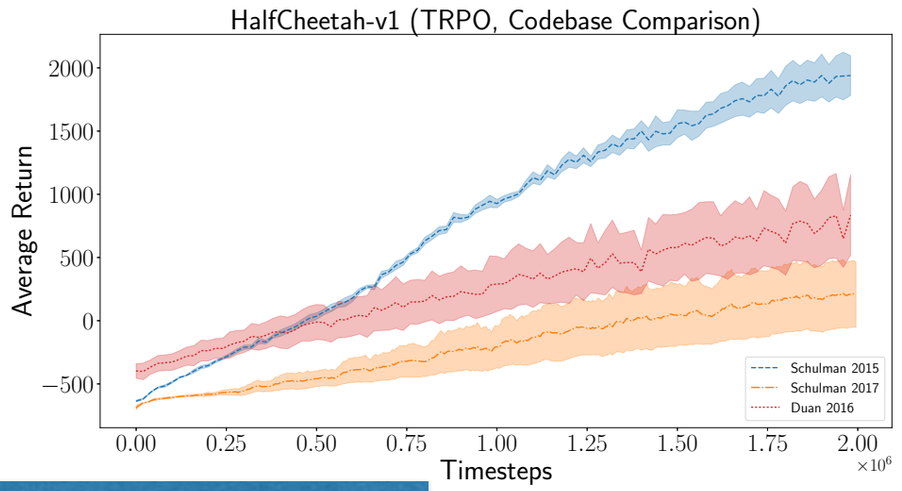
`blog.openai.com/openai-baselines-dqn/`

"Reinforcement learning results are tricky to reproduce: performance is very noisy, algorithms have many moving parts which allow for subtle bugs, and many papers don't report all the required tricks."

"RL algorithms are challenging to implement correctly; good results typically only come after fixing many seemingly-trivial bugs."

`arxiv:1709.06560`





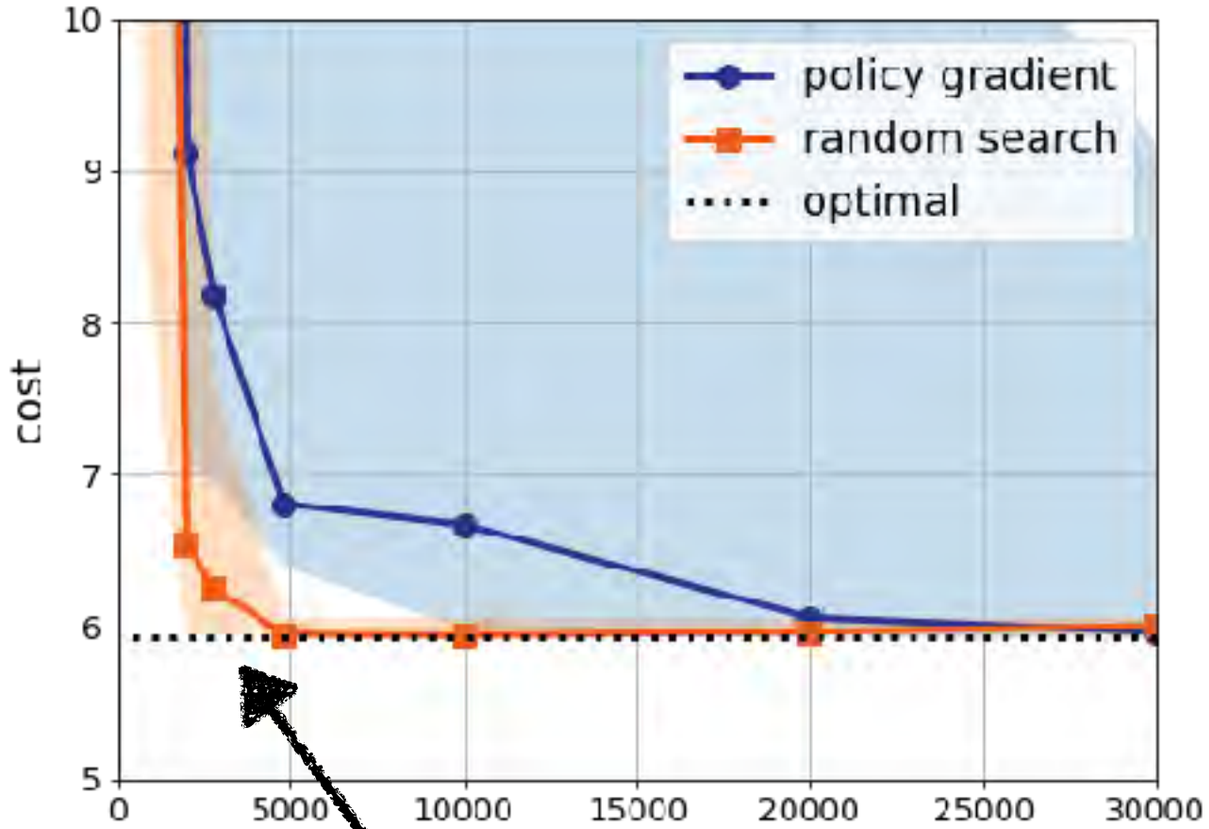
There has to be a better way!

"Simplest" Example: LQR

$$\text{minimize} \quad \sum_{t=0}^{T} (x_t)_1^2 + r u_t^2$$

$$\text{subject to} \quad x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_t$$

$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$



Model-based and ADP with 10 samples

Extraordinary Claims Require Extraordinary Evidence*

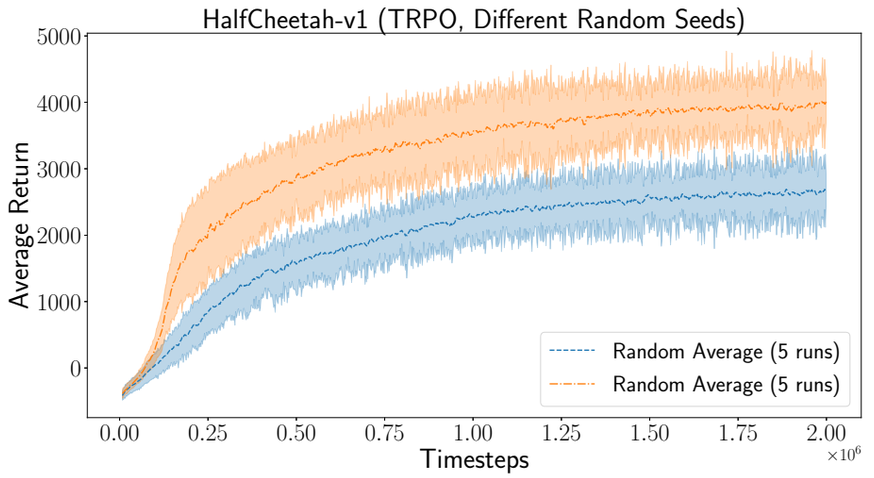
* only if your prior is correct

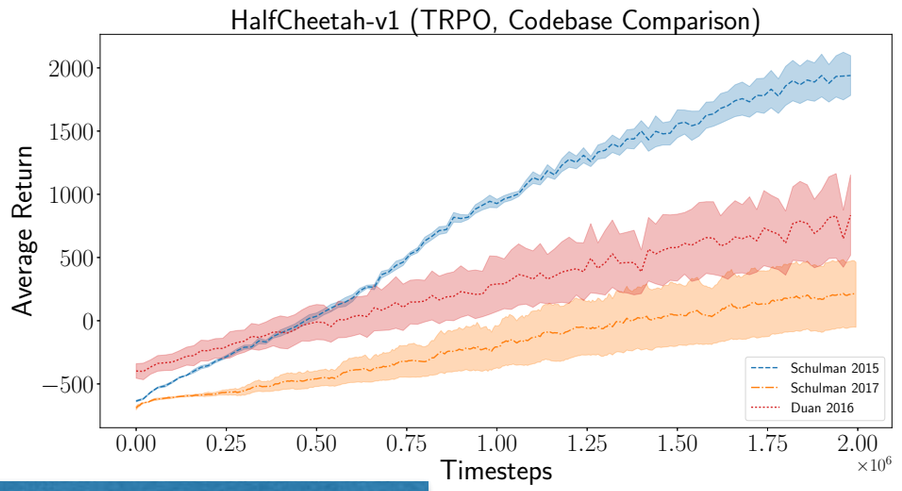
`blog.openai.com/openai-baselines-dqn/`

"Reinforcement learning results are tricky to reproduce: performance is very noisy, algorithms have many moving parts which allow for subtle bugs, and many papers don't report all the required tricks."

"RL algorithms are challenging to implement correctly; good results typically only come after fixing many seemingly-trivial bugs."

`arxiv:1709.06560`





There has to be a better way!

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t, e_t) \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

Model-based RL

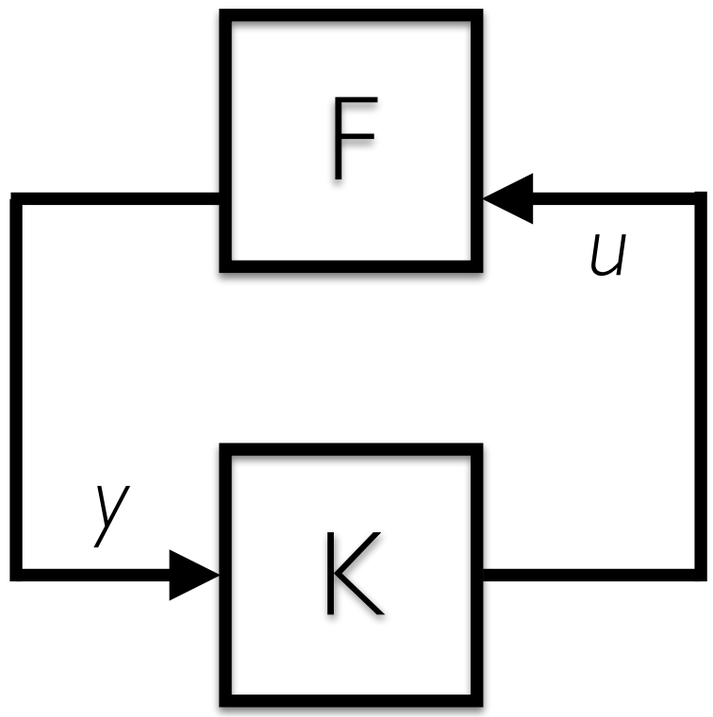
Collect some simulation data. Should have $x_{t+1} \approx \varphi(x_t, u_t) + \nu_t$

Fit dynamics with *supervised learning:* $\hat{\varphi} = \arg\min_{\varphi} \sum_{t=0}^{N-1} ||x_{t+1} - \varphi(x_t, u_t)||^2$

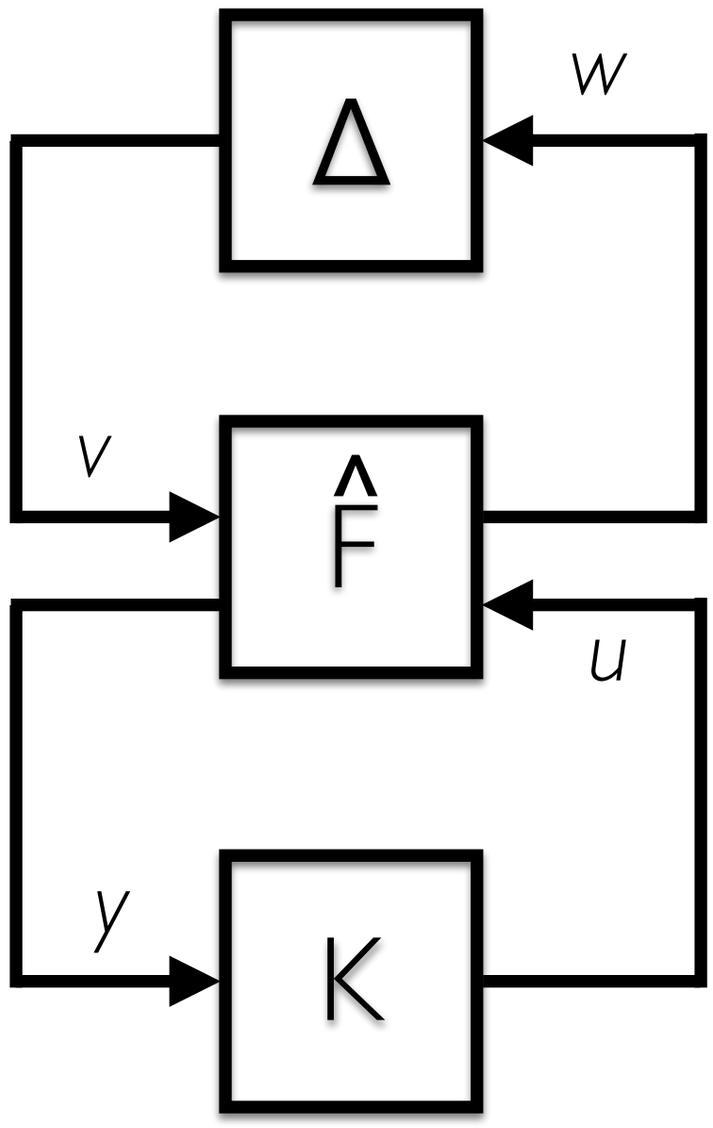
Solve approximate problem:

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_\omega\left[\sum_{t=1}^{T} C_t(x_t, u_t)\right] \\ \text{s.t.} \quad & x_{t+1} = \varphi(x_t, u_t) + \omega_t \\ & u_t = \pi(\tau_t) \end{aligned}$$

Coarse-ID control



Coarse-ID control



High dimensional stats bounds the error

Coarse-grained model is trivial to fit

Design robust control for feedback loop

Coarse-ID control (static case)

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & x^* Q x \\ \text{subject to} \quad & x = Bu + x_0 \end{aligned}$$

B unknown!

Collect data: $\{(x_i, u_i)\}$ $\qquad x_i = Bu_i + x_0 + e_i$

Estimate B: $\underset{B}{\text{minimize}} \quad \sum_{i=1}^{N} \|Bu_i + x_0 - x_i\|^2 \Rightarrow \hat{B}$

Guarantee: $\|B - \hat{B}\| \leq \epsilon$ with high probability

Note: $x = \hat{B}u + x_0 + \Delta_B u$

Robust optimization problem:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \sup_{\|\Delta_B\| \leq \epsilon} \|Q^{1/2}(x - \Delta_B u)\| \\ \text{subject to} \quad & x = \hat{B}u + x_0 \end{aligned}$$

Coarse-ID control (static case)

$$\begin{array}{ll} \underset{u}{\text{minimize}} & x^*Qx \\ \text{subject to} & x = Bu + x_0 \end{array}$$

B unknown!

Collect data: $\{(x_i, u_i)\}$ $\qquad x_i = Bu_i + x_0 + e_i$

Estimate B: $\underset{B}{\text{minimize}} \quad \sum_{i=1}^{N} \|Bu_i + x_0 - x_i\|^2 \Rightarrow \hat{B}$

Guarantee: $\|B - \hat{B}\| \leq \epsilon$ with high probability

Solve robust optimization problem:

$$\begin{array}{ll} \underset{u}{\text{minimize}} & \sup_{\|\Delta_B\| \leq \epsilon} \|Q^{1/2}(x - \Delta_B u)\| \\ \text{subject to} & x = \hat{B}u + x_0 \end{array}$$

Relaxation: (Triangle inequality!)

$$\begin{array}{ll} \underset{u}{\text{minimize}} & \|Q^{1/2}x\| + \epsilon\lambda\|u\| \\ \text{subject to} & x = \hat{B}u + x_0 \end{array}$$

Coarse-ID control (static case)

$$\begin{array}{ll} \underset{u}{\text{minimize}} & x^* Q x \\ \text{subject to} & x = Bu + x_0 \end{array}$$

B unknown!

Collect data: $\{(x_i, u_i)\}$ $\qquad x_i = Bu_i + x_0 + e_i$

Estimate B: $\underset{B}{\text{minimize}} \quad \sum_{i=1}^{N} \|Bu_i - x_i\|^2 \quad \Rightarrow \quad \hat{B}$

Guarantee: $\|B - \hat{B}\| \leq \epsilon$ with high probability

Relaxation:
(Triangle inequality!)

$$\begin{array}{ll} \underset{u}{\text{minimize}} & \|Q^{1/2}x\| + \epsilon\lambda\|u\| \\ \text{subject to} & x = \hat{B}u + x_0 \end{array}$$

Generalization bound

$$\text{cost}(\hat{u}) \leq \text{cost}(u_\star) + 4\epsilon\lambda\|u_\star\|\|Q^{1/2}x_\star\| + 4\epsilon^2\lambda^2\|u_\star\|^2$$

Coarse-ID control (static case)

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & x^* Q x \\ \text{subject to} \quad & x = Bu + x_0 \end{aligned}$$

B unknown!

Collect data: $\{(x_i, u_i)\}$ $\qquad x_i = Bu_i + x_0 + e_i$

Estimate B: $\underset{B}{\text{minimize}} \quad \sum_{i=1}^{N} \|Bu_i + x_0 - x_i\|^2 \Rightarrow \hat{B}$

Guarantee: $\|B - \hat{B}\| \leq \epsilon$ with high probability

Relaxation:
(Triangle inequality!)

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \|Q^{1/2}x\| + \epsilon\lambda\|u\| \\ \text{subject to} \quad & x = \hat{B}u + x_0 \end{aligned}$$

Generalization bound

$$\text{cost}(\hat{u}) = \text{cost}(u_\star) + \mathcal{O}(\epsilon)$$

Coarse-ID Control for LQR

$$\text{minimize} \quad \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right]$$

$$\text{s.t.} \quad x_{t+1} = Ax_t + Bu_t + e_t$$

Gaussian noise
Assume stable A

Run an experiment for T steps with random input. Then

$$\text{minimize}_{(A,B)} \quad \sum_{i=1}^{T} \|x_{i+1} - Ax_i - Bu_i\|^2$$

If $T \geq \tilde{O}\left(\frac{\sigma^2(d+p)}{\lambda_{\min}(\Lambda_c)\epsilon^2}\right)$ where $\Lambda_c = A\Lambda_c A^* + BB^*$

controllability Gramian

then $\|A - \hat{A}\| \leq \epsilon$ and $\|B - \hat{B}\| \leq \epsilon$ w.h.p.

[Dean, Mania, Matni, R., Tu, 2017]
[Mania, R., Simchowitz, Tu, 2018]

Coarse-ID Control for LQR

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \sup_{\|\Delta_A\|_2 \le \epsilon_A,\ \|\Delta_B\|_2 \le \epsilon_B} \lim_{T\to\infty} \tfrac{1}{T} \sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t \\ \text{s.t.} \quad & x_{t+1} = (\hat{A} + \Delta_A) x_t + (\hat{B} + \Delta_B) u_t \end{aligned}$$

Solving an SDP relaxation of this robust control problem yields

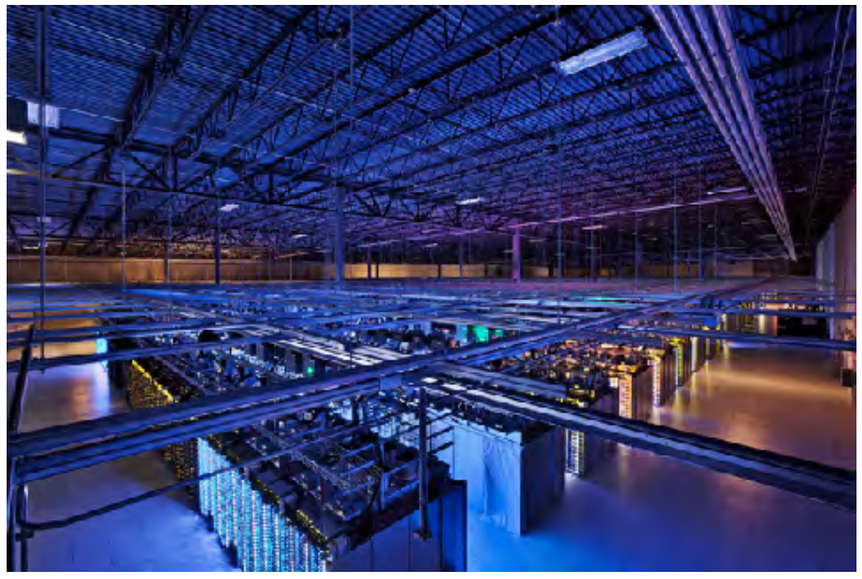
$$\frac{J(\hat{K}) - J_\star}{J_\star} \le C\, \Gamma_{\mathrm{cl}} \left(\lambda_{\min}(\Lambda_c)^{-1/2} + \|K_\star\|_2 \right) \sqrt{\frac{\sigma^2 (d+p)}{T}} \qquad \text{w.h.p.}$$

$$\Lambda_c = A \Lambda_c A^* + B B^*$$
controllability Gramian

$$\Gamma_{\mathrm{cl}} := \|(zI - A - BK_\star)^{-1}\|_{\mathcal{H}_\infty}$$
closed loop gain

This also tells you when your cost is finite!
Extends to unstable A case as well.

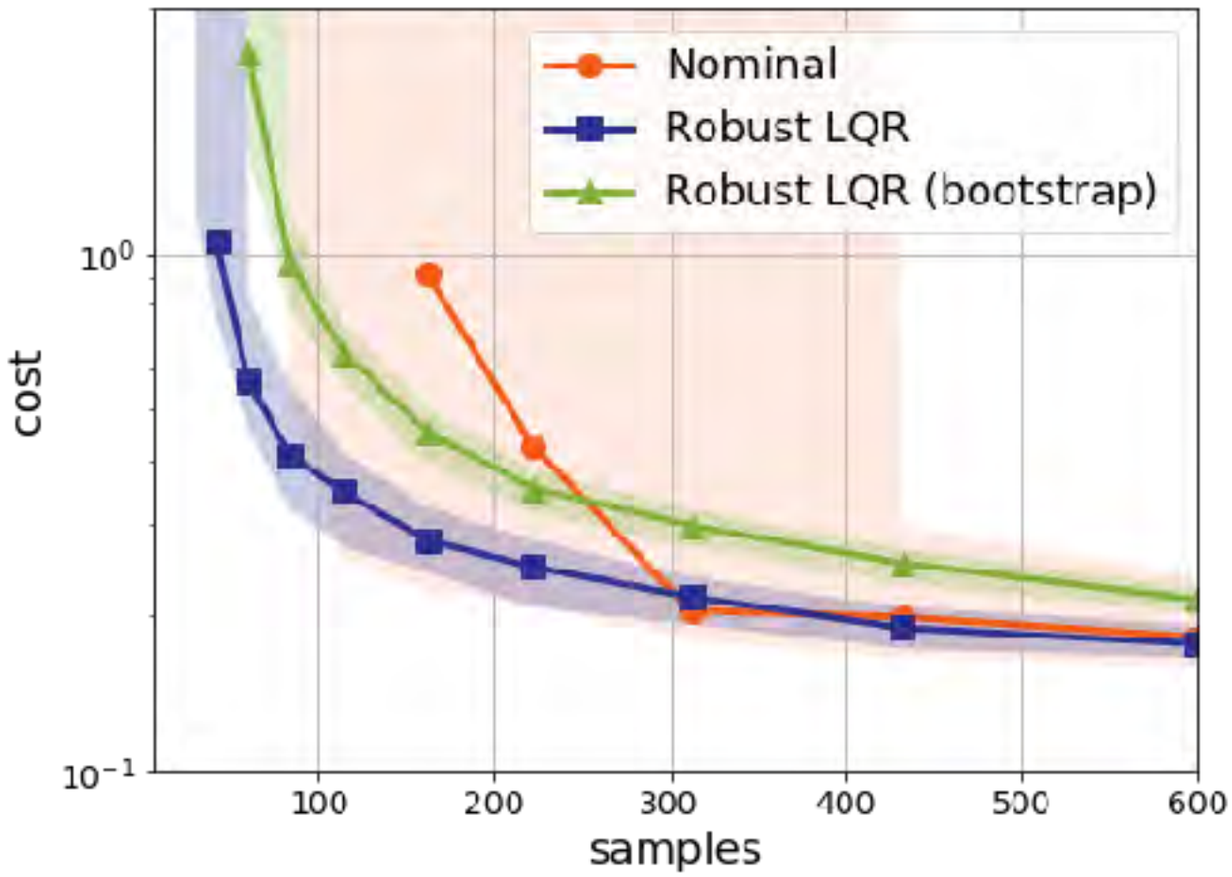
[Dean, Mania, Matni, R., Tu 2017]

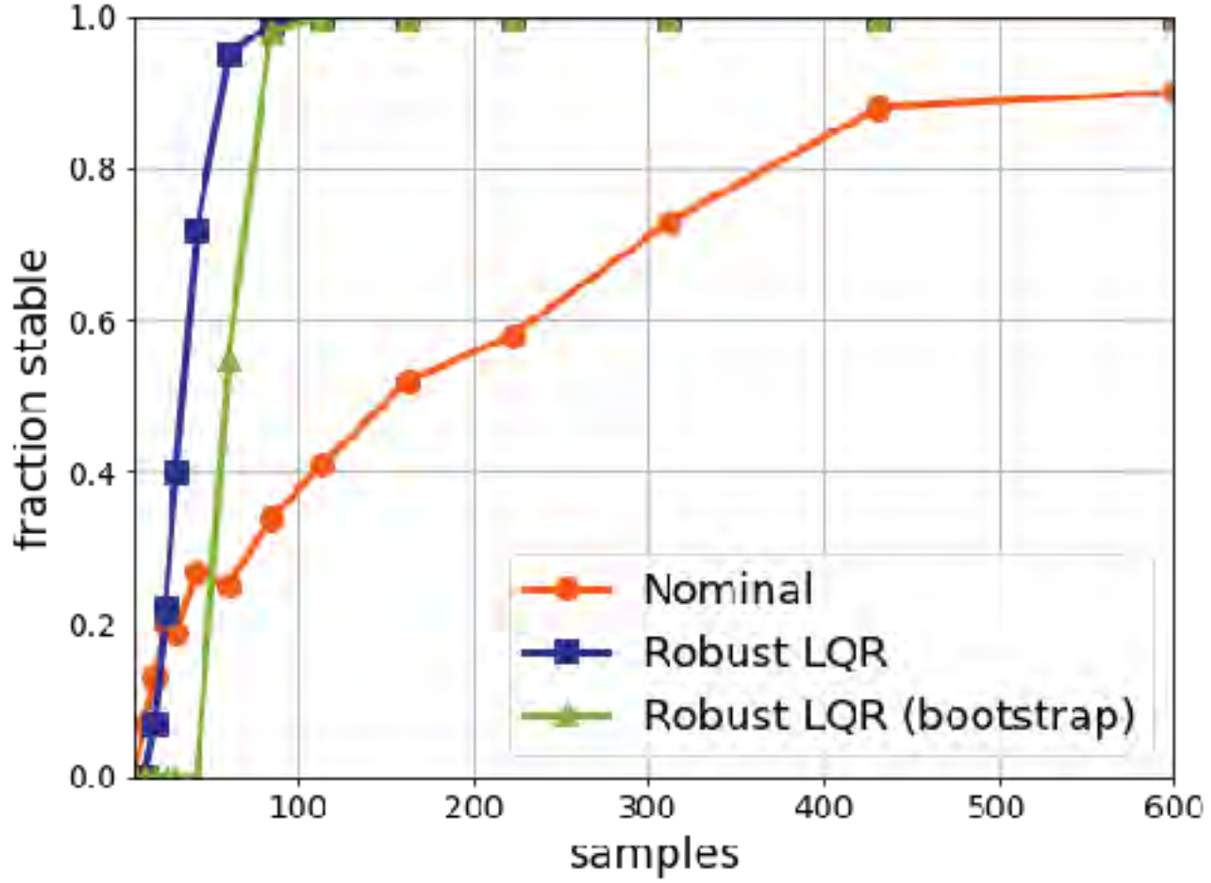


Why robust?

$$x_{t+1} = \begin{bmatrix} 1.01 & 0.01 & 0 \\ 0.01 & 1.01 & 0.01 \\ 0 & 0.01 & 1.01 \end{bmatrix} x_t + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u_t + e_t$$

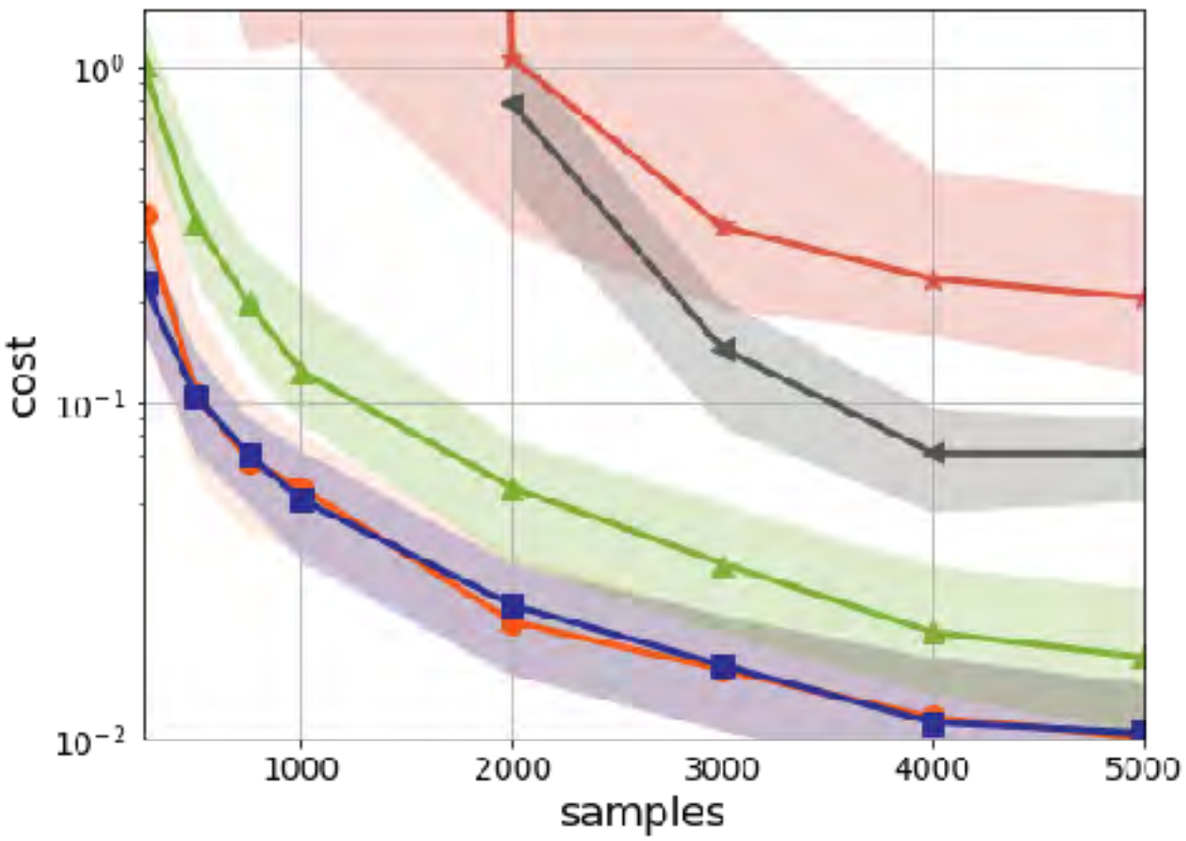
Slightly unstable system, system ID tends to think some nodes are stable

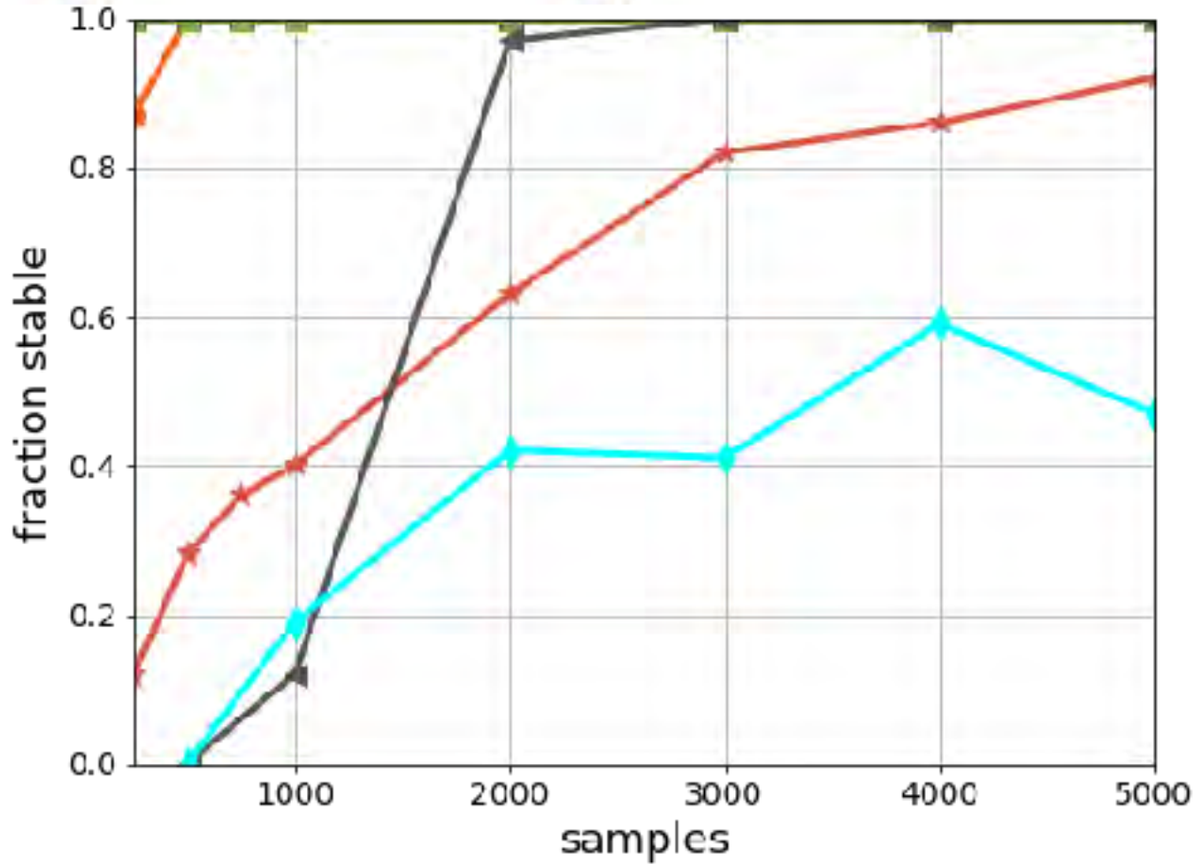




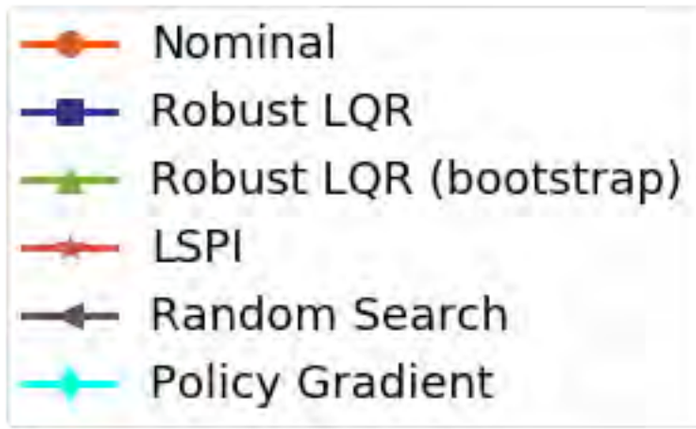
Least-squares estimate may yield unstable controller

Robust synthesis yields stable controller





Model-free performs worse than model-based



Why has no one done this before?

- Coarse-ID control is the first non-asymptotic bound for this oracle model.
- Our guarantees for least-squares estimation required some heavy machinery.
  - Indeed, best bounds building on very recent papers.
- Our SDP relaxation uses brand new techniques in controller parameterization *(Systems Level Synthesis* by Matni *et al.)*
  - **Key insight:** Robustness makes analysis tractable
- *The Singularity has arrived!*
- Lots of work in the last years, to be highlighted in extended bib.

Even LQR is not simple!!!

$$\text{minimize} \quad J := \sum_{t=1}^{\infty} x_t^T Q x_t + u_t^T R u_t$$

$$\text{s.t.} \quad x_{t+1} = A x_t + B u_t + e_t \qquad \textit{Gaussian noise}$$

$$\frac{J(\hat{K}) - J_\star}{J_\star} \leq C\, \Gamma_{\mathrm{cl}} \left(\lambda_{\min}(\Lambda_c)^{-1/2} + \|K_\star\|_2 \right) \sqrt{\frac{\sigma^2 (d+p) \log(1/\delta)}{n}}$$

where $\Lambda_c = A \Lambda_c A^* + B B^*$ *controllability Gramian*

$\Gamma_{\mathrm{cl}} := \|(zI - A - BK_\star)^{-1}\|_{\mathcal{H}_\infty}$ *closed loop gain*

Hard to estimate
Control insensitive to mismatch

Easy to estimate
Control very sensitive to mismatch

50 papers on Cosma Shalizi's blog say otherwise!

Need to fix learning theory for time series.

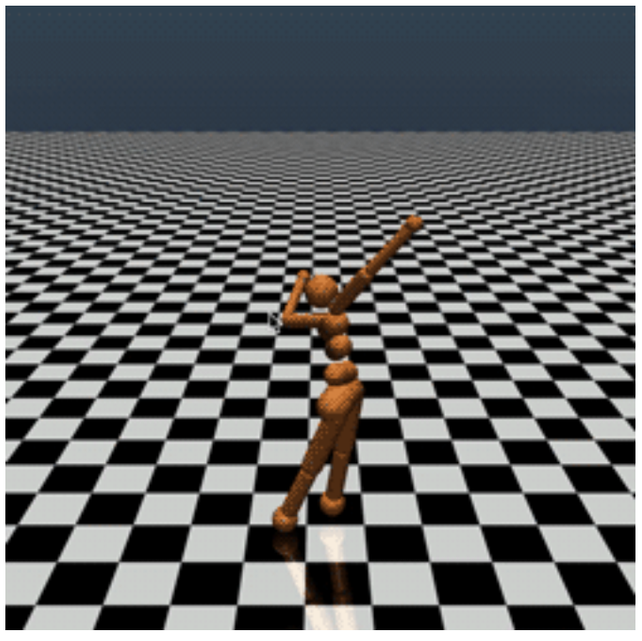
The Linearization Principle

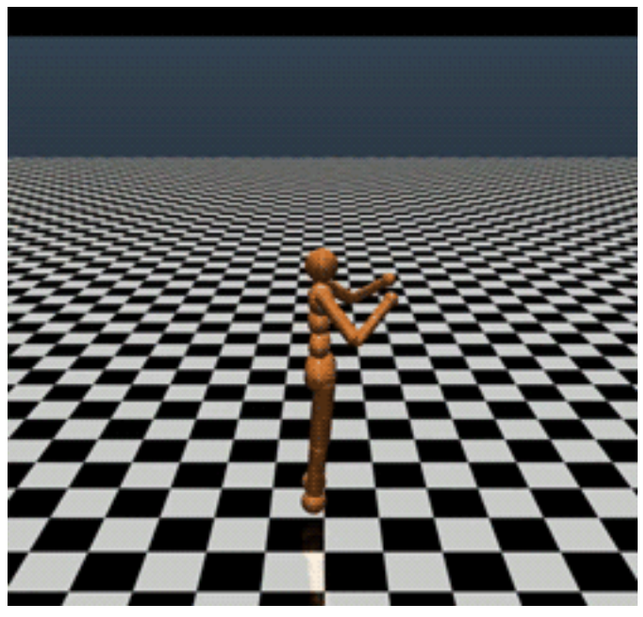
If a machine learning algorithm does crazy things when restricted to linear models, it's going to do crazy things on complex nonlinear models too.

What happens when we return to nonlinear models?

Random search of linear policies outperforms Deep Reinforcement Learning

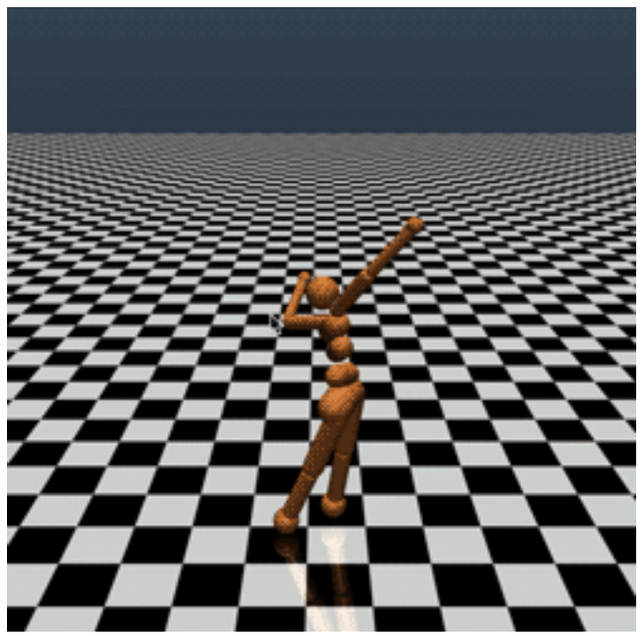
Larger is better		**Maximum average reward**		
Task	**RS**	**NG-lin**	**NG-rbf**	**TRPO-nn**
Swimmer-v1	365	366	365	131
Hopper-v1	3909	3651	3810	3668
HalfCheetah-v1	6722	4149	6620	4800
Walker	11389	5234	5867	5594
Ant	5146	4607	4816	5007
Humanoid	11600	6440	6849	6482

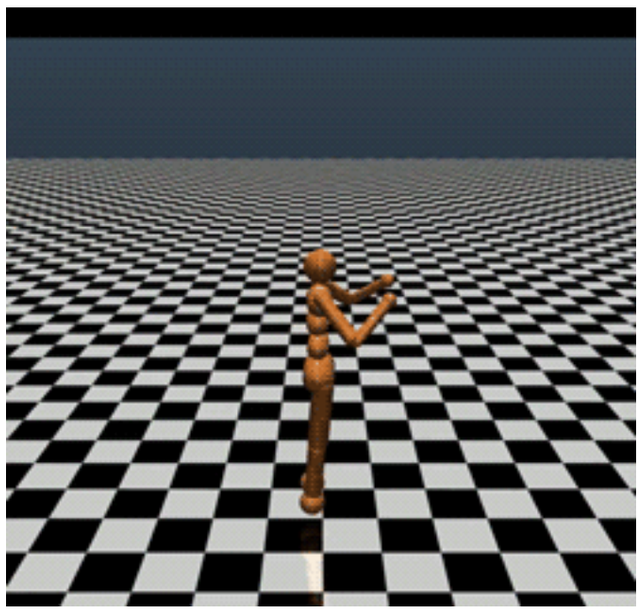


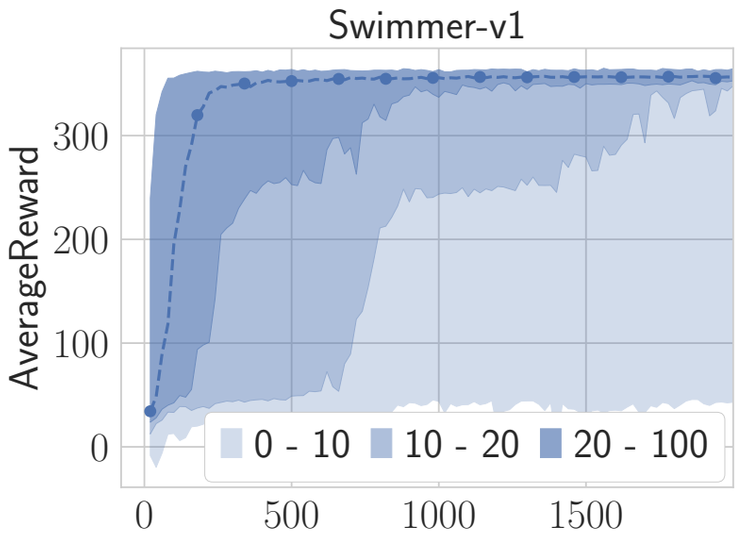


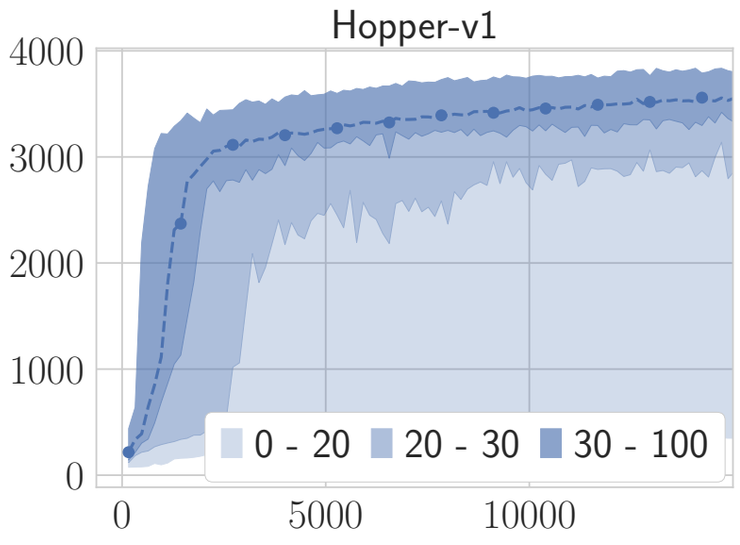
Larger is better

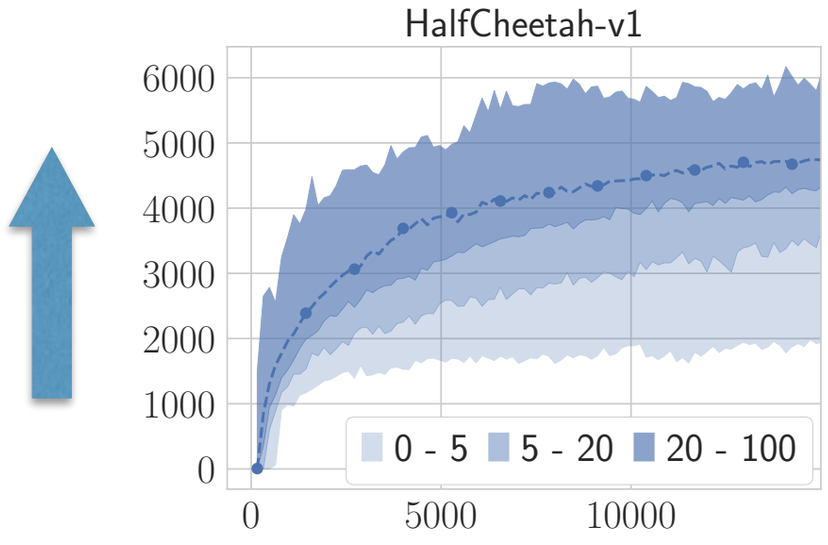
		Maximum average reward		
Task	**RS**	**NG-lin**	**NG-rbf**	**TRPO-nn**
Swimmer-v1	365	366	365	131
Hopper-v1	3909	3651	3810	3668
HalfCheetah-v1	6722	4149	6620	4800
Walker	11389	5234	5867	5594
Ant	5146	4607	4816	5007
Humanoid	11600	6440	6849	6482

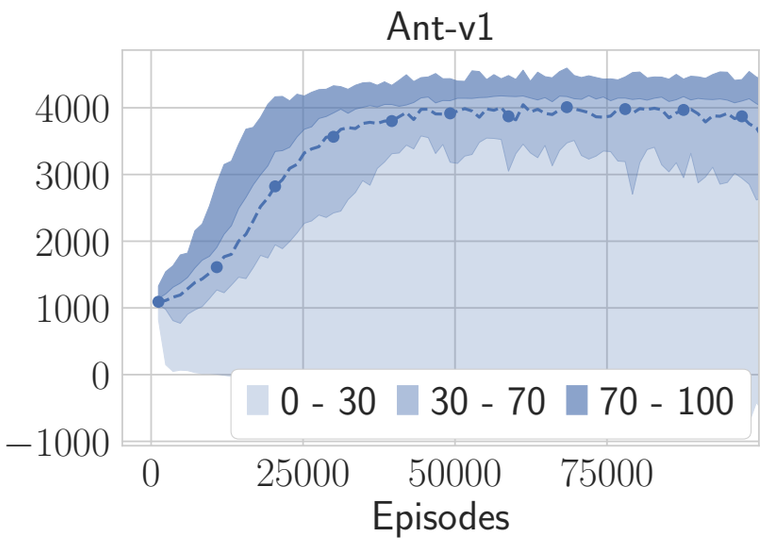


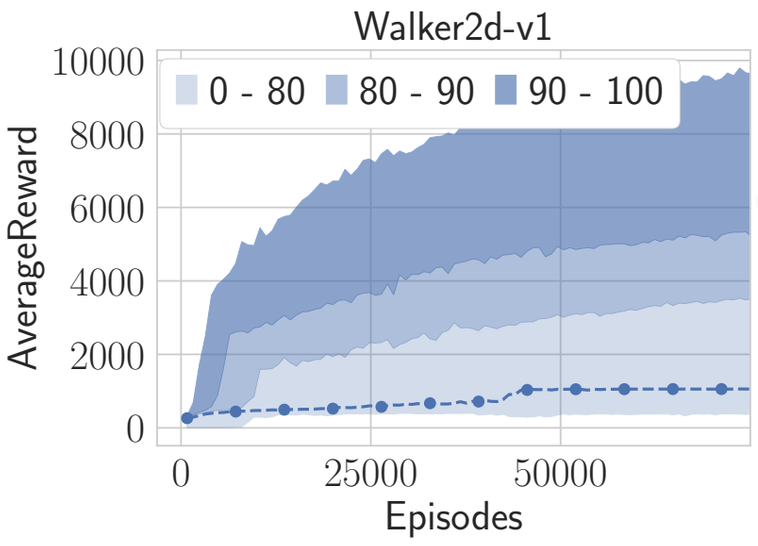


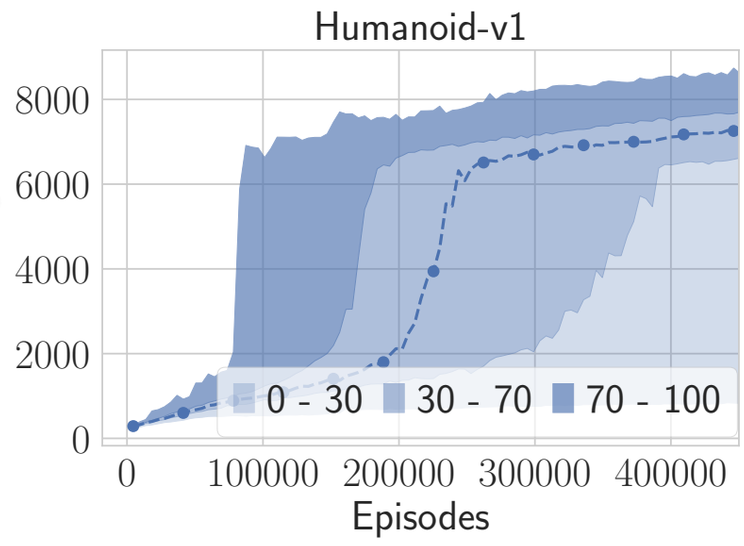












Larger is better

Model Predictive Control

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t) + C_f(x_{T+1})\right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t),\ x_1 = x \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

Optimal Policy: $\boxed{\pi(x) = \arg\min_u \mathcal{Q}_1(x, u)}$

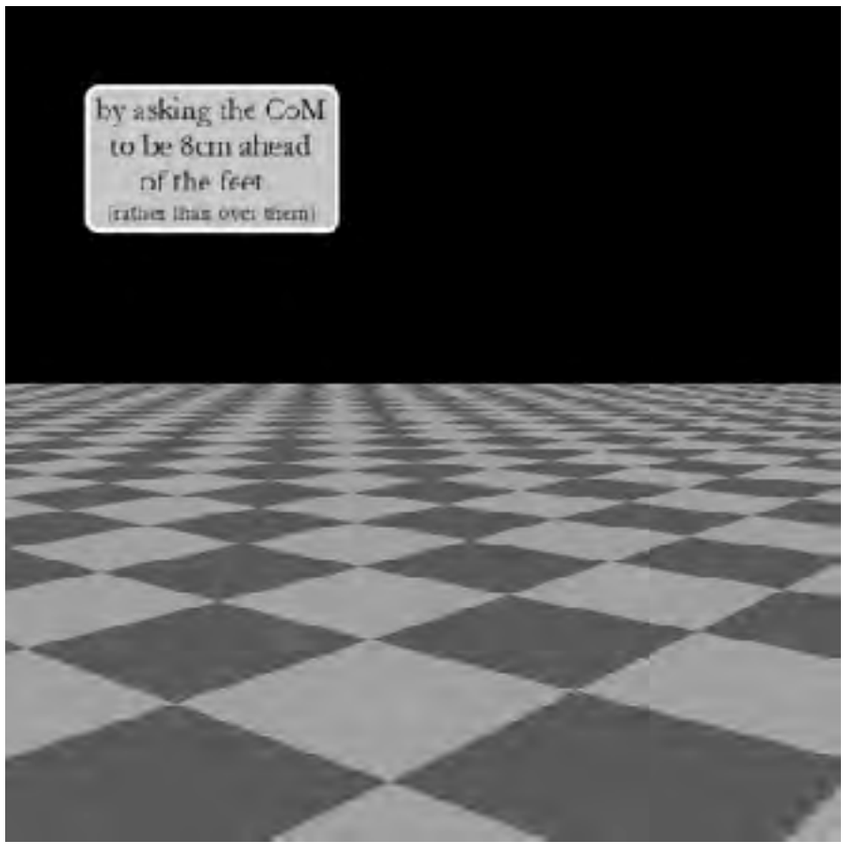
$$\mathcal{Q}_1(x, u) = C_1(x, u) + \mathbb{E}_e\left[\min_{u'} \mathcal{Q}_2(f_1(x, u, e), u')\right]$$

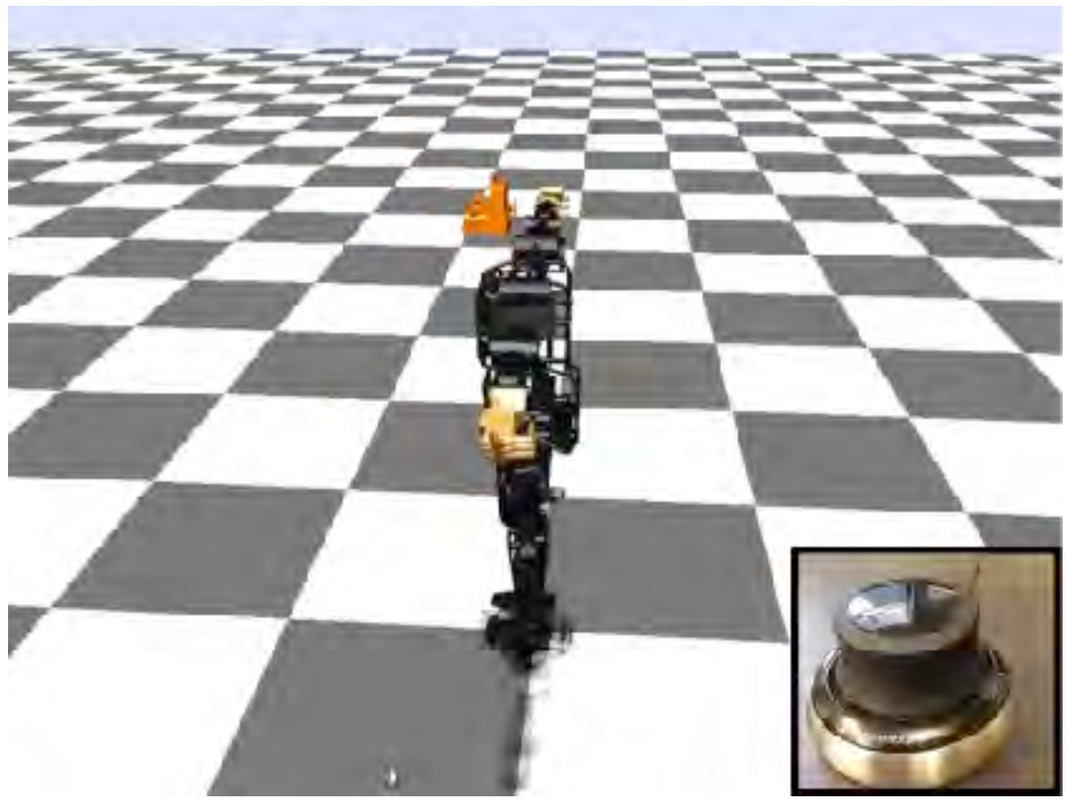
MPC: use the Q-function for *all* time steps

$$\mathcal{Q}_1(x, u) = \sum_{t=1}^{H} C_t(x, u) + \mathbb{E}_e\left[\min_{u'} \mathcal{Q}_{H+1}(f_H(x, u, e), u')\right]$$

MPC ethos: plan on short time horizons, use feedback to correct modeling error and disturbance.

Model Predictive Control





Videos from Todorov Lab

`https://homes.cs.washington.edu/~todorov/`

Learning in MPC

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t) + C_f(x_{T+1})\right] \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t),\ x_1 = x \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

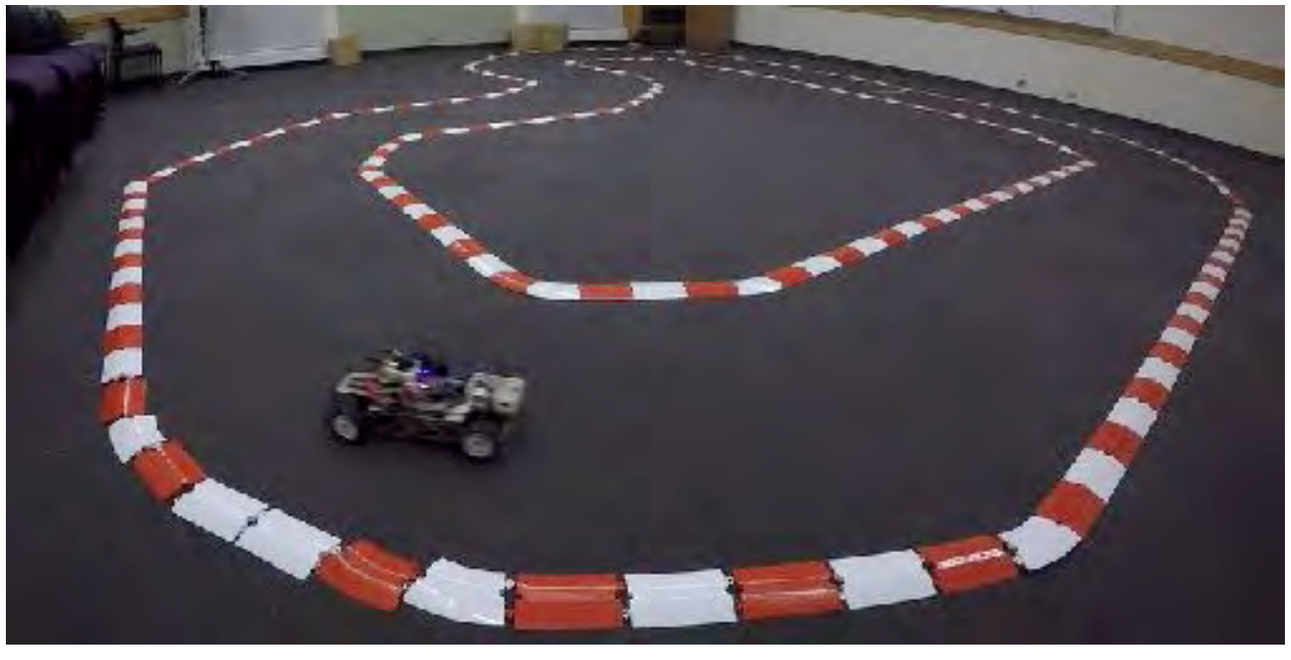
Optimal Policy:

$$\pi(x) = \arg\min_u \mathcal{Q}_1(x, u)$$

MPC: use the Q-function for *all* time steps

$$\mathcal{Q}_1(x, u) = \sum_{t=1}^{H} C_t(x, u) + \mathbb{E}_e\left[\min_{u'} \mathcal{Q}_{H+1}(f_H(x, u, e), u')\right]$$

Use past data to learn the terminal Q-function:
The value of a state is the minimum value seen for the remainder of the episode from that state.



[Rosolia et al., 2016]

So many things left to do…

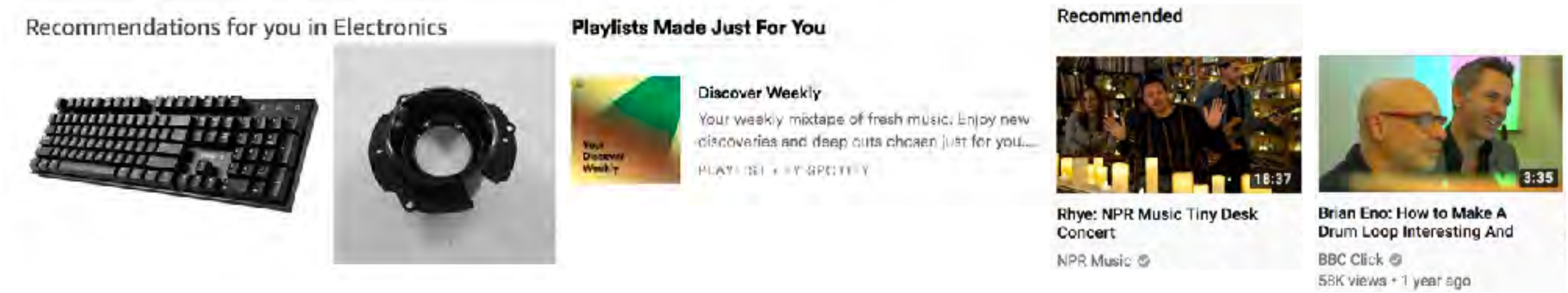
- Are the coarse-ID results optimal, even with respect to the parameters?
- Tight upper and lower sample complexities for LQR. (Is the optimal error scaling $T^{-1/2}$ or T^{-1}?)
- Finite analysis of learning in MPC.
- Adaptive control.
- Iterative learning control.
- Nonlinear models, constraints, and improper learning.
- Safe exploration, earning about uncertain environments.
- Implementing in test-beds.

Actionable Intelligence

~~Control Theory~~

~~Reinforcement Learning~~ is the study of how to use past data to enhance the future manipulation of a dynamical system

Actionable Intelligence is the study of how to use past data to enhance the future manipulation of a dynamical system



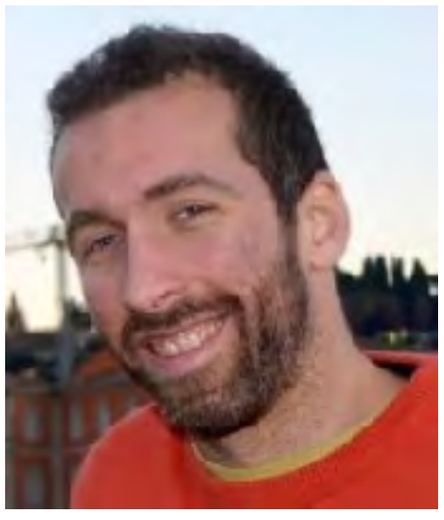
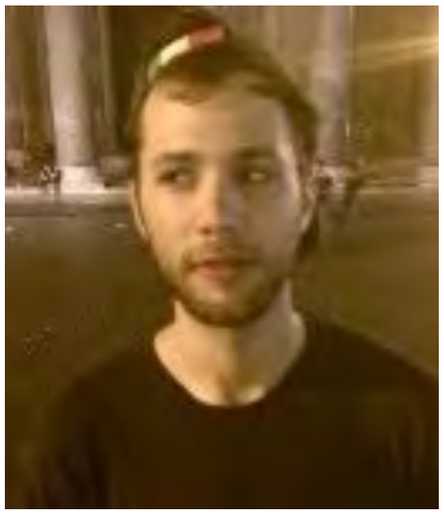
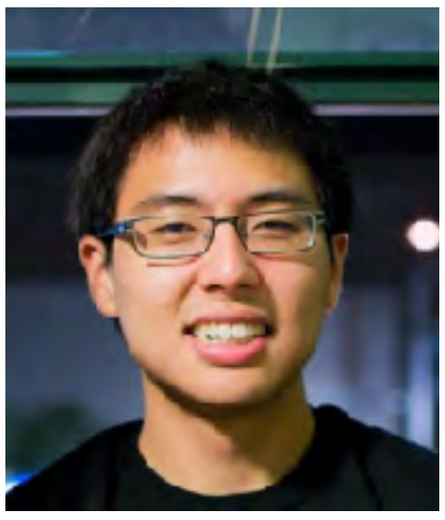
As soon as a machine learning system is unleashed in feedback with humans, that system is an actionable intelligence system, not a machine learning system.

Actionable Intelligence

trustable, scalable, predictable

Collaborators



Joint work with Sarah Dean, Aurelia Guy, Horia Mania, Nikolai Matni, Max Simchowitz, and Stephen Tu.

Recommended Texts

- D. Bertsekas. *Dynamic Programming and Optimal Control.* 4th edition, volumes 1 (2017) and 2 (2012). Athena Scientific.
- D. Bertsekas. and J. Tsitsiklis. *Neuro-dynamic Programming.* Athena Scientific, 1996.
- F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems.* Cambridge, 2017.
- B. Recht "A Tour of Reinforcement Learning: The View from Continuous Control." **`arXiv:1806.09460`**

References from the Actionable Intelligence Lab

- **`argmin.net`**
- "On the Sample Complexity of the Linear Quadratic Regulator." S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. **`arXiv:1710.01688`**
- "Non-asymptotic Analysis of Robust Control from Coarse-grained Identification." S. Tu, R. Boczar, A. Packard, and B. Recht. **`arXiv:1707.04791`**
- "Least-squares Temporal Differencing for the Linear Quadratic Regulator" S. Tu and B. Recht. In submission to ICML 2018. **`arXiv:1712.08642`**
- "Learning without Mixing." H. Mania, B. Recht, M. Simchowitz, and S. Tu. In submission to COLT 2018. **`arXiv:1802.08334`**
- "Simple random search provides a competitive approach to reinforcement learning." H. Mania, A. Guy, and B. Recht. **`arXiv:1803.07055`**
- "Regret Bounds for Robust Adaptive Control of the Linear Quadratic Regulator." S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. **`arXiv:1805.09388`**

`https://people.eecs.berkeley.edu/~brecht/publications.html`

$$\underset{u}{\text{minimize}} \quad \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right]$$
$$\text{s.t.} \quad x_{t+1} = Ax_t + Bu_t + e_t$$

Key to formulation:
Write (*x*,*u*) as linear function of disturbance

$$\begin{bmatrix} x_t \\ u_t \end{bmatrix} = \sum_{k=1}^{t} \begin{bmatrix} \Phi_x[k] \\ \Phi_u[k] \end{bmatrix} e_{t-k}$$

$$\mathbb{E}\left[x_t^* Q x_t\right] = \sigma^2 \sum_{k=1}^{t} \mathrm{Tr}\left(\Phi_x[k]^* Q \Phi_x[k]\right)$$

$$\mathbb{E}\left[u_t^* R u_t\right] = \sigma^2 \sum_{k=1}^{t} \mathrm{Tr}\left(\Phi_u[k]^* R \Phi_u[k]\right)$$

$$\underset{\Phi}{\text{minimize}} \quad \left\| \begin{bmatrix} Q^{\frac{1}{2}} & 0 \\ 0 & R^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \Phi_x \\ \Phi_u \end{bmatrix} \right\|_F^2$$
$$\text{s.t.} \quad \Phi_x[t+1] = A\Phi_x[t] + B\Phi_u[t]$$
$$\Phi_x[0] = I$$

Key to formulation:
Write (x,u) as linear function of disturbance

$$\begin{bmatrix} x_t \\ u_t \end{bmatrix} = \sum_{k=1}^{t} \begin{bmatrix} \Phi_x[k] \\ \Phi_u[k] \end{bmatrix} e_{t-k}$$

$$\begin{aligned} \underset{\Phi}{\text{minimize}} \quad & \sup_{\|\Delta_A\|_2 \le \epsilon_A,\ \|\Delta_B\|_2 \le \epsilon_B} \left\| \begin{bmatrix} Q^{\frac{1}{2}} & 0 \\ 0 & R^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \Phi_x \\ \Phi_u \end{bmatrix} \right\|_F^2 \\ \text{s.t.} \quad & \Phi_x[t+1] = (\hat{A} + \Delta_A)\Phi_x[t] + (\hat{B} + \Delta_B)\Phi_u[t] \\ & \Phi_x[0] = I \end{aligned}$$

As in the static case, push robustness into cost.

$$\begin{aligned} \underset{\Phi}{\text{minimize}} \quad & \sup_{\|\Delta_A\|_2 \le \epsilon_A,\ \|\Delta_B\|_2 \le \epsilon_B} \left\| \begin{bmatrix} Q^{\frac{1}{2}} & 0 \\ 0 & R^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \Phi_x \\ \Phi_u \end{bmatrix} (I + \Delta)^{-1} \right\|_F^2 \\ \text{s.t.} \quad & \Phi_x[t+1] = \hat{A}\Phi_x[t] + \hat{B}\Phi_u[t] \\ & \Phi_x[0] = I \end{aligned}$$

SLS Formulation of Robust LQR

$$\underset{\gamma\in[0,1)}{\text{minimize}}\ \frac{1}{1-\gamma} \qquad \min_{\Phi_x,\Phi_u} \left\| \begin{bmatrix} Q^{\frac{1}{2}} & 0 \\ 0 & R^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \Phi_x \\ \Phi_u \end{bmatrix} \right\|_F$$

$$\text{s.t.} \quad \Phi_x[t+1] = \hat{A}\Phi_x[t] + \hat{B}\Phi_u[t]$$

$$\Phi_x[0] = I$$

$$\sup_{\theta\in[0,2\pi)} \left\| \sum_{k=1}^{T_{\max}} \begin{bmatrix} \frac{\epsilon_A}{\sqrt{\alpha}}\Phi_x[k] \\ \frac{\epsilon_B}{\sqrt{1-\alpha}}\Phi_u[k] \end{bmatrix} e^{ik\theta} \right\| \leq \gamma$$

- Solvable by SDP for fixed γ
- Binary search over γ to find optimal solution

SLS Formulation of Robust LQR

$$\begin{aligned}
\text{minimize}_{X,Z,W,\gamma} \quad & \frac{1}{(1-\gamma)^2}\{\text{Trace}(QW_{11}) + \text{Trace}(RW_{22})\} \\
\text{subject to} \quad & \begin{bmatrix} X & X & Z^* \\ X & W_{11} & W_{12} \\ Z & W_{21} & W_{22} \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} X - I & \hat{A}X + \hat{B}Z & 0 & 0 \\ (\hat{A}X + \hat{B}Z)^* & X & \epsilon_A X & \epsilon_B Z^* \\ 0 & \epsilon_A X & \alpha\gamma^2 I & 0 \\ 0 & \epsilon_B Z & 0 & (1-\alpha)\gamma^2 I \end{bmatrix} \succeq 0 .
\end{aligned}$$

- Solvable by SDP for fixed γ
- Binary search over γ to find optimal solution
- Optimal controller is $K = -ZX^{-1}$